ROYAUME DU MAROC
UNIVERSITE ABDELMALEK ESSAADI
Ecole Nationale des Sciences Appliquées
Tanger

Année universitaire 2019-2020

TD-TP N°1 : Programmation en langage d'assemblage

Exercice 1:

1- Charger les registres AX, BX, CX, DX avec une information de 16 bits en utilisant l'adressage immédiat, l'adressage registre, l'adressage direct et l'adressage basé.
2- Charger dans les registres BX et DX respectivement par le mot faible et le mot fort d'une information de 32 bits. Ensuite charger les registres CL et DL respectivement par le 1ère octet et le 3ème octet de cette information.

Exercice 2:

Soit un tableau constitué de 10 éléments de 8 bits. Faire un programme qui fait les taches suivantes :

1- Chargement dans AL du 1er élément du tableau en utilisant l'adressage direct.
2- Chargement dans BL du 5ème élément du tableau en utilisant l'adressage direct.
3- Chargement dans CL du 1er élément du tableau en utilisant l'adressage basé.
4- Chargement dans DL du dernier élément du tableau en utilisant l'adressage basé relatif.
5- Chargement dans AL du 4ème élément du tableau en utilisant l'adressage indexé.

Exercice 3

Ecrire un programme permettant l'addition de deux informations de 16 bits en utilisant l'adressage registre.

Exercice 4:

Soient deux informations de 32 bits, en utilisant l'adressage direct, écrire un programme permettant :
L'addition et stockage du résultat en mémoire
La soustraction et stockage du résultat en mémoire.

Exercice 5:

Charger les registres AX, BX, CX, DX avec une information de 16 bits. Faire les opérations suivantes : Empilement, remise à zéro, dépilement des quatre registres.

Exercice 6 :

Faire l'addition de 10 éléments de 8 bits en stockant le résultat en mémoire (résultat ≤255).
En utilisant l'adressage indexé de la table
En utilisant l'adressage indexé et l'adressage basé.
Déterminer les éléments d'une table C obtenus par la relation suivante :

$$C_i = (A_i + B_i)$$

Où A_i et B_i représentent respectivement les éléments de deux tables A et B.

ROYAUME DU MAROC
UNIVERSITE ABDELMALEK ESSAADI
Ecole Nationale des Sciences Appliquées
Tanger

Année universitaire 2019-2020

TD-TP N°2 : Programmation en langage d'assemblage

Exercice 1:

Ecrire un programme qui Compte le nombre de zéro dans un tableau constitué de 10 éléments de 8 bits. Ce nombre sera stocker dans une variable mémoire de 8 bits.

Exercice 2 :

Faire le transfert de 10 éléments de 8 bits d'un bloc A vers un bloc B.

Exercice 3 :

Construire un tableau B à partir de l'inversion d'un tableau A contenant 5 informations de 16 bits.

Exercice 4 :

Faire le calcul suivant :

$$P = (A1 \times A2 + A3) \times (A4 + 10)$$

Où $A1$, $A2$, $A3$ et $A4$ sont des données de 8 bits.

Exercice 5 :

Donner le programme qui fait la remise à zéro des cases mémoires situées entre 00F0H et 0FFFH

Exercice 6 :

Déterminer la somme s définit par :

$$S = \sum (a_i - b_i)$$

Où a_i et b_i représentent respectivement les éléments de deux tables A et B. on considère que : $0 \leq a_i - b_i$, $S \leq 255$ et $i \leq 10$.

Exercice 7 :

1) Déterminer le minimum d'un bloc constitué de 10 éléments de 8 bits. Le minimum sera stocké dans le registre AL.
2) Déterminer le maximum d'un bloc constitué de 10 éléments de 8 bits. Le maximum sera stocké dans le registre BL.

Exercice 8 :

Faire le transfert d'un tableau de 5 éléments de 32 bits. Ce tableau sera stocké entre les adresses de déplacements suivantes : 1000H et 1013H.

Année universitaire 2019-2020

TD-TP N°3 : Programmation en langage d'assemblage

Interruption logicielles

Exercice 1:

1- Saisir à partir du clavier, avec écho, deux informations numériques X et Y appartenant au pavet numérique 0-9 dont les codes ASCII sont entre 30H et 39H.

Entrée : AH=00H
Sortie : AL=Code ASCII de la touche
Interruption : 16H du BIOS
Stocker ces informations en mémoire. La conversion ASCII-Binaire : AL=AL-30H
Effectuer la somme Z=X+Y avec Z<9, ensuite afficher Z en utilisant :
Entrée : AH=0EH
AL=Code ASCII de l'information
Sortie : aucune
Interruption : 10H du BIOS
La conversion binaire-ASCII :AL=AL+30H

2- Saisir sans écho puis réafficher une information alphanumérique en utilisant les interruptions 16H et 10H. refaire la saisie avec écho (int 21H du DOS).

Exercice 2: affichage d'une chaine de caractères

Déclarer et initialiser un tableau de chaines de caractères (le dernier élément de la chaine doit être le symbole $), ensuite donner le programme qui permet d'afficher sur écran cette chaine de caractères :
AH=09H
DX=offset de la chaine de caractères
Sortie : aucune
Interruption : 21H du dos

Exercice 3:

Refaire l'exercice 2 en saisissant la chaîne de caractères à partir du clavier

Exercice 4:

Déterminer la version du DOS de votre machine
Entrée : AH=30H
Sortie : AL=Numéro de version principal (XX)
AH= Numéro de version complémentaire (YY)
Interruption : 21H du DOS
Faire un affichage sous la forme : XX.YY

Année universitaire 2019-2020

TD-TP N°4 : Programmation en langage d'assemblage

Exercice 1:

Ecrire un programme permettant l'effacement de l'écran, en utilisant l'interruption 10H du BIOS, équivalente à la commande CLS du DOS.

Exercice 2:

Ecrire un programme permettant l'affichage sur écran un message à (ligne 10, colonne 40) à l'aide de l'interruption 10H du BIOS.

Exercice 3:

1- Visualiser une page contenant le même caractère répété 80×25 fois.
2- Refaire la question 1- en accédant directement a mémoire vidéo en mode texte.

On donne :

Adresse de la première case mémoire de la RAM vidéo : B800H :0000H

Instruction **STOSB** : permettant le transfert de (AL) vers (ES :DI).

Il est à noter que l'exécution de cette instruction incrémente automatiquement le registre index DI et que chaque caractère est représenté dans la RAM vidéo par deux octets : 1ère octet pour l'ASCII du caractère, le 2ème octet pour son attribut.

Exercice 4:

Dessiner sur écran, en mode graphique, un quadrillage formé de 20 lignes et de 32 colonnes. On choisira pour les lignes la couleur violette (attribut=2) et la couleur bleue pour les colonnes (attribut=1).

Les lignes et les colonnes seront tracées à l'aide d'un ensemble de pixels (points).

Exercice 5:

Ajouter à l'exercice 7 de la série 2 (recherche d'un minimum de 8 bits) deux procédure ; la première pour la conversion Binaire pur-ASCII, la 2ème pour afficher sur écran la valeur de ce minimum.

Exercice 6:

Donner le programme qui permet de tester la zone mémoire RAM situé entre les adresses de déplacement 0200H et 02FFH. Ceci en utilisant la valeur actuelle du registre segment DS.

Stocker la valeur 00H dans les différentes cases mémoires

Lire les contenus de ces cases en les comparant avec 00H

Comparer le nombre de cases défectueuses (variable DEFEC) et non défectueuses (BOB).

Afficher sur écran les valeurs des variables BON et DEFEC ainsi que les messages associés à ces variables.

On donne :

Instruction STOSB voir (exercice 3)

Instruction **LODSB :** permet de charger (AL) par le contenu de (DS :SI).

Il est à noter que l'exécution de LODSB incrémente automatiquement SI.

Le résumé des fonctions vidéo

(AH)	fonction	Registres charges supplémentaire à l'entrée	Résultats obtenues en registres à la sortie
0	Etablir le mode vidéo	(AL)=0 écran 40×25 caractères noir et blanc (par (défaut) (AL)=1 écran 40×25 caractères couleur (AL)=2 écran 80×25 caractères noir et blanc (AL)=3 écran 80×25 caractères couleur (AL)=4 écran graphique 320×200 pixels couleur (AL)=5 écran graphique 320×200 pixels noir et blanc (AL)=0 écran graphique 640×200 pixels noir et blanc	
1	Établir l'espace de mouvement pour le curseur	(CH) bits 0-4 la ligne du début du curseur (CH) bits 5-7 mis à zéro (CL) bits 0-4 la ligne de la fin du curseur (CL) bits 5-7 mis à zéro	
2	Établir la position du curseur	(DH,DL) ligne, colonne : (0,0) le point supérieur gauche (BH) le numéro de la page doit avoir la valeur zéro pour le mode graphique	
3	Lire la position du curseur lumineux	(BH) le numéro de la page doit avoir la valeur zéro pour le mode graphique	(DH,DL) ligne, colonne du curseur (CH,CL) l'espace de mouvement, établit en avance pour le curseur
4	Lire la position du crayon lumineux		(AH)=0 le commutateur pour le crayon lumineux n'est pas en marche (AH)=1 les valeurs retournées dans les registres sont valides (DH,DL) ligne, colonne du curseur (CH) la ligne de la trame (0 à 199) (BX) la colonne du pixel (0 à 319636)
5	La sélection de la page active, seulement en mode caractère	(AL) le numéro de la page (0 à 7 pour modes 0 et 1 0 à 3 pour modes 2 et 3)	
6	Défilement de la ma page de bas en haut	(AL) le nombre de lignes (AL)=0 blanchir la fenêtre entière (CH,CL) la ligne et la colonne du coin supérieur gauche (DH,DL) la ligne et la colonne du coin inférieur droit (BH) attribut utilisé pour une ligne blanche	
7	Défilement de la page active de haut en bas	(AL) le nombre de lignes (AL)=0 blanchir la fenêtre entière (CH,CL) la ligne et la colonne du coin supérieur gauche (DH,DL) la ligne et la colonne du coin inférieur droit (BH) attribut utilisé pour une ligne blanche	
8	Lire l'attribut et le caractère qui se trouve à	(BH) la page à afficher (seulement en mode caractère)	(AL) le caractère lu

	la position courante du curseur		(AH) attribut du caractère (seulement pour le mode caractère)
9	Ecrire l'attribut et le caractère qui se trouve à la position courante du curseur (chaque page a un curseur)	(BH) la page à afficher (seulement pour le mode caractère) (BL) - Attribut du caractère (mode caractère) - Couleur du caractère mode graphique) (CX) Le nombre de caractère à écrire (AL) le caractère à écrire	
10	Ecrire le caractère seulement, à la position du curseur	(BH) la page à afficher (CX) le nombre de caractère à écrire (AL) le caractère à écrire	
11	Etablir la palette des couleurs (seulement pour le mode graphique 320×200 pixels	(BH) ID de la palette couleur (voir la documentation IBM-PC) (BL) la valeur de la couleur dans ID	
12	Ecrire un point (mode graphique)	(DX) le numéro e la ligne (CX) le numéro de la colonne (AL) la valeur de la couleur (0, 1, 2, 3)	
13	Lire un point (mode graphique)	(DX) le numéro e la ligne (CX) le numéro de la colonne	(AL) le point lu
14	Mode télétype	(AL) le caractère à écrire (BL) la couleur d fond (mode graphique) (BH) la page mode caractère	
15	Lire l'état vidéo		(AL) le mode courant (AH) le nombre de caractères colonnes sur l'écran

Année universitaire 2019-2020

Corrigé :TD-TP N°1 : Programmation en langage d'assemblage

Exercice 1:

Exercice 1 :

1) Charger AX,BX,CX etDX

```
.model small
.stack
.data
val1 word 34
.code
.startup
        mov ax,4567   ;A. immédiat
        mov bx, ax    ;A. registre
        mov cx,val1   ;A. direct
        mov bx,1000h
        mov dx,[bx]   ;A. basé
 .exit
end
```

2) Charger BX, DX, CL et DL

```
.model small
.stack
.data
val1 dword   BF1FF1E2h
.code
.startup
        mov bx,offset val1
        mov dx,word ptr [bx+2]
        push bx
        mov bx,word ptr [bx]
        pop bx
        mov cl,byte ptr [bx]
        mov dl,byte ptr [bx+2]
.exit
end
```

Exercice 2:

```
.model small
.stack
.data
tab db 4,5,6,7,12,3,5,9,34,56
 .code
.startup
      mov al,tab[0]
      mov bl,tab[4]
      mov bx, offset tab[0]
      mov cl,tab[bx]
      mov dl,tab[bx+9]
      mov di,3
      mov al,tab[di]
.exit
end
```

Exercice 3 :

```
;Exercice 3 serie 1:Adressage basé
.model small
.stack
.data
val1 dd          F111h
val2 dd          9222h
Result dd 0h
.code
.startup
   mov bx,offset val1
   mov ax,[bx]
   add ax,val2
   mov bx,offset Result
   mov [bx],ax
   adc [bx+2],0
 .exit
 end
```

Exercice 4:

```
.model small
.stack
.data
val1 dd          FFFFFFFFh
val2 dd          11111111h
Result dd 0h

.code
.startup
   mov bx,offset val1
   mov ax,[bx]
   push bx
   mov bx,offset val2
   add ax,[bx]
   push bx
   mov bx,offset result
   mov [bx],ax
   mov cx,bx
   pop bx
   mov ax,[bx+2]
   pop bx
   adc ax,[bx+2]
   mov bx,cx
   mov [bx+2],ax
   mov [bx+4],0
   adc [bx+4],0
   .exit
 end
```

Exercice 5 :

```
;Exercice 5 serie 1
.model small
.stack
.data
A dw 54
B dw 67
C dw 88
D dw 255
.code
.startup
mov ax,A
mov bx,B
mov cx,C
mov dx,D

push ax
push bx
push cx
push dx

and ax,0
and bx,0
and cx,0
and dx,0

pop dx
pop cx
pop bx
pop ax
.exit
end
```

Exercice 6:

```
;Exercice 6 serie 1
.model small
.stack
.data
Result db 0
tab db 4,5,6,7,12,3,5,9,34,56
.code
.startup
mov al,0
mov di,0
mov cx, 10
etq:    add al,tab[di]
        inc di
        loop etq
        mov Result,al
.exit
end
```

Exercice 7 :

```
;Exercice 7 serie 1
.model small
.stack
.data
tabA db 4,5,6,7,12,3,5,9,34,56
tabB db 4,5,6,7,12,3,5,9,34,56
tabC db 10 dup(0)
.code
.startup
mov al,0
mov di,0
mov cx, 10
etq:    mov al,tabA[di]
        add al,tabB[di]
        mov tabC[di],al
        inc di
        loop etq
.exit
end
```

Année universitaire 2019-2020

Corrigé :TD-TP N°2 : Programmation en langage d'assemblage

Exercice 1 :

```
.model small
.stack
.data
Val1 dd 1,0,5,6,0,4,0,0,23,0
Nzero db 0
.code
.startup
mov dx,0
repete:     cmp tab[di],0
            jz compte
            jmp near ptr compare
compte:     inc Nzero
compare:    inc di
            cmp di,10
            jnz repete
.exit
end
```

Exercice 2:

```
.model small
.stack
.data
tabA db 4,5,6,7,12,3,5,9,34,56
tabB db 10dup(0)
.code
.startup
            mov di,0
etq:        mov al,tabA[di]
            mov tabB[di],al
            inc di
            cmp di,10
            jnz etq
.exit
end
```

Exercice 3 :

```
.model small
.stack
.data
tabA word 3,5,9,34,56
tabB word 5dup(0)
.code
.startup
        mov di,0
        mov si,4
etq:    mov ax,tabA[di]
        mov tabB[si],ax
        inc di
        inc di
        dec si
        dec si
        cmp di,10
        jnz etq
.exit
end
```

Exercice 4:

```
.model small
.stack
.data
        a1 db 7
        a2 db 5
        a3 db 6
        a4 db 5
        r dw 0
.code
.startup
        mov al,a1
        mul a2
        add al,a3
        adc ah,0
        mov bl,a4
        add bl,10
        adc bh,0
        mul bx
        mov r,ax
.exit
end
```

Exercice 5 :

```
;Exercice 5 serie 2
.model small
.stack
.data
.code
.startup
mov bx,00F0h
repete: mov [bx],0
        inc bx
        cmp bx,2000h
        jnz repete
.exit
end.exit
end
```

Exercice 6:

```
;Exercice 6 serie 2
.model small
.stack
.data
Result db 0
tabA db
4,5,26,77,12,53,25,19,34,56
tabB db 4,5,6,7,12,3,5,9,24,36
tabC db 10 dup(0)
.code
.startup
mov al,0
mov di,0
mov cx, 10
etq:    mov al,tabA[di]
        sub al,tabB[di]
        mov tabC[di],al
        inc di
        loop etq
.exit
end
```

Exercice 7 :

1)

```
.model small
.stack
.data
tab db 123,129,234,129,222,234,100,101,99,33
y db 10
z db 3dup(0)
.code
.startup
        mov cx,9
        mov si,1
        mov al, tab[0]
etq:    cmp al,tab[si]
        jnae etq1
        mov al,tab[si]
etq1:   inc si
loop etq
exit
end
```

Exercice 7:

2)

```
.model small
.stack
.data
tab db 123,129,234,129,222,234,100,101,99,33
y db 10
z db 3dup(0)
.code
.startup
        mov cx,9
        mov si,1
        mov al, tab[0]
etq:    cmp al,tab[si]
        jae etq1
        mov al,tab[si]
etq1:   inc si
loop etq
exit
end
```

Exercice 8 :

```
;Exercice 8 serie 2
.model small
.stack
.data
tabA dd
FFFFFFFFh,EEEEEEEEh,22662266h,77777
777h,12121212h
.code
.startup
mov bx,1000h
mov di,0
mov cx, 5
etq:    mov ax,word ptr tabA[di]
        mov dx,word ptr tabA[di+2]
        mov [bx],ax
        mov [bx+2],dx
        add di,4
        add bx,4
        loop etq
.exit
end
```

Année universitaire 2019-2020

Corrigé :TD-TP N°3 : Programmation en langage d'assemblage

Exercice 1: sans écho

```
.model small
.stack
.data
x db 0
y db 0
.code
.startup
mov ah,00h
int 16h
sub al,30h
mov x, al
mov ah,00h
int 16h
sub al,30h
mov y,al
add al,x
add al,30h
mov ah,0eh
int 10h
.exit
end
```

Exercice 2:

```
;exercice 2 serie 3
.model small
.stack
.data
msg db "Hello GSEA1 $",10,13
.code
.startup
etq:    mov ah,09h
        mov dx,offset msg
        int 21h
.exit
endend
```

Exercice 3:

```
;exercice 3 serie 3
.model small
.stack
.data
X db 10 dup (0)
;y wd 0
.code
.startup
mov di,0
etq:    mov ah,01h
        int 21h
        mov X[di],al
        inc di
        cmp al,'$'
jnz etq:
        mov ah,09h
        mov dx,offset X
        int 21h
.exit
end
```

Exercice 4:

```
;Exercice 4 serie 3

.model small
.stack
.data
X db ?
Y db 10
Z db 4 dup(0)
.code
.startup
mov ah,30h
int 21h
add al,30h
mov ah,0Eh
int 10h
mov ah,0Eh
mov al,'.'
int 10h
mov al,X
mov di,0
etq:    mov ah,0
div Y
add ah,30h
mov z[di],ah
inc di
cmp al,0
jnz etq
affiche:    dec di
mov ah,0Eh
mov al,Z[di]
int 10h
cmp di,0
jnz affiche
mov ah,0Eh
mov al,0Ah
int 10h
.exit
end
```

Année universitaire 2019-2020

TD-TP N°4 : Programmation en langage d'assemblage

Exercice 1 :

```
Exercice 1 serie 4
.model small
.stack
.data
.code
.startup
mov dh,0
lig:  mov dl,0
col:      mov ah,0eh
          mov al,' '
          int 10h
          inc dl
          cmp dl,80
          jnz col
          inc dh
          cmp dh,25
          jnz lig
.exit
end.exit
end
```

Exercice 2:

```
Exercice 2 serie 4
.model small
.stack
.data
msg db "Hello, GSEA1!", 0Dh,0Ah, 24h
.code
.startup
mov ah,0
mov al,3
int 10h
mov dh,10
mov dl,40
mov ah,2
int 10h
mov ah,09h
mov dx, offset msg
int 21h
mov dh,11
mov dl,0
mov ah,2
int 10h
mov ah,09h
mov dx, offset msg
int 21h
.exit
end
```

Exercice 3 :

```
;Exercice 3-1 serie 4
.model small
.stack
.data
.code
.startup
mov ah,0h
mov al,3
int 10h
mov dh,0
lig:    mov dl,0
col:    mov ah,0eh
        mov al,'a'
        int 10h
        inc dl
        cmp dl,80
        jnz col
        inc dh
        cmp dh,25
        jnz lig
.exit
end
```

Exercice 3:

```
;Exercice 3-2 serie 4
.model small
.stack
.data
.code
.startup
mov ax,B800h
mov es,ax
mov di,0000h
mov al,'J'
etq:    stosb
        inc di
        mp di,100
        jnz etq
.exit
end
```

Exercice 4 :

```
;Exercice 4 serie 4
.model small
.stack
.data
.code
.startup




.exit
end
```

Exercice 5:

```
;Exercice 5 serie 4
.model small
.stack
.data
.code
.startup
.model small
.stack
.data
tab db
123,129,234,129,222,234,100,101,102,213
 y db 10
 z db 3dup(0)
 .code
 .startup
 mov cx,9
 mov si,1
 mov al, tab[0]
 etq:   cmp al,tab[si]
        jae etq1
        mov al,tab[si]
 etq1:  inc si
 loop etq
 call conversion
 call affiche
 .exit
 conversion proc near
    mov di,0
 etq2:  mov ah,0
        div y
        add ah,30h
        mov z[di],ah
        inc di
        cmp al,0
        jnz etq2
        ret
conversion endp
 affiche proc near
  etq3: dec di
  mov ah,0eh
  mov al,z[di]
  int 10h
  cmp di,0
  jnz etq3
  ret
affiche endp
end
```

Exercice 6:

```
;Exerice 6 serie 4
.model small
.stack
.data
x db 00H
msg1 db "le nombre de cases memoires
bonnes est:$",10,13
msg2 db "le nombre de cases memoires
defectueuses est:$",10,13
bon dw 0
defec dw 0
y db 10
z db 3 dup(0)
.code
.startup
mov ah,0   ;Etablir le mode video
mov al,3   ;Ecran 80x25
int 10h
mov di,0200h
mov bx,di
mov es,bx
etq:    mov al,x
stosb
lodsb
cmp al,x
je etq1
inc defec
jmp short etq2
etq1:   inc bon
etq2:   cmp di,020Fh     ;0300h
        jne etq
        mov ah,09h
        mov dx,offset msg1
        int 21h
        mov ax,bon
call conversion
call affiche
mov dh,12
mov dl,9
mov ah,2
int 10h
mov ah,09h
mov dx,offset msg2
int 21h
mov ax,defec
call conversion
call affiche
.exit
conversion proc near
    mov di,0
 etq4:  mov ah,0
        div y
        add ah,30h
        mov z[di],ah
        inc di
        cmp al,0
        jnz etq4
        ret
conversion endp
affiche proc near
  etq5: dec di
        mov ah,0eh
        mov al,z[di]
        int 10h
        cmp di,0
        jnz etq5
  ret
  affiche endp
.exit
end
```

Exercice supplémentaire : Tri croissant d'un tableau

```
.model small
.stack
.data
tab db 2,45,6,8,1,0,55,123,253,9
.code
.startup
mov cx,9
etq2:   push cx
        mov di,9
        sub di,cx
        push di
        mov al,tab[di]
etq1:   cmp al,tab[di+1]
        jnae etq
        xchg al,tab[di+1]
etq:    inc di
        loop etq1
        pop di
        pop cx
        mov tab[di],al
        loop etq2
.exit
end
```