Document de travail contenant les TD et préparations de TP Assembleur

JMT

1. Initiation

Rappels :

Les registres, la mémoire (adressage segment + offset => indiquer qu'en 8086 seuls les registres BX, SI, DI et BP (segment implicite SS) permettent d'adresser la mémoire), l'architecture d'un programme assembleur :

```
DOSSEG
.MODEL SMALL
.STACK 100H
.386
.DATA
.CODE
mov ax,@data
mov ds,ax
...
mov ah,4ch
int 21h
end
```

l'instruction MOV, les interruptions DOS (INT 21H) n° 1 (saisie d'un caractère), 2 (affichage d'un caractère), 09h (affichage d'une chaîne de caractères), 0ah (saisie d'une chaîne de caractères)
les instructions CMP, les sauts (poly distribué en cours), les procédures CALL-RET

```
DOSSEG
.386
.MODEL SMALL
.STACK 100H
.DATA
.CODE
mov ax,@data
mov ds,ax
...
call toto   ; appel d'une procédure toto
...
mov ah,4ch
int 21h           ; fin effective du programme principal

toto PROC   ; début de la procédure toto
...
ret
endp        ; fin de la procédure

end               ; fin du module contenant pgm principal et procédures
```

Exemple de programme avec procédure
Contenu de PROC.ASM (cf. cours)

```
DOSSEG
.MODEL SMALL
.STACK 200H
.DATA
      ch1   db'salut',0dh,0ah,0
finsp:
      ret
      endp
debut:
      mov   ax,@data
```

```
      ch2   db'mon',0dh,0ah,0
      ch3   db'vieux',0dh,0ah,0
.CODE
imp   PROC
boucle:
      mov   dl,[bx]
      and   dl,dl
      jz    finsp
      inc   bx
      mov   ah,2
      int   21h
      jmp   boucle
      mov   ds,ax
      mov   bx,offset ch1
      call  imp
      mov   bx,offset ch2
      call  imp
      mov   bx,offset ch3
      call  imp
fin:
      mov   ah,4ch
      int   21h
      end   debut
```

(voir également le listing distribué en cours contenant une procédure dans un fichier différent de celui contenant le programme principal)

Exercices de préparation au TP n°1 :

Ex 1 : Ecrire un programme qui affiche à l'écran 'bonjour tout le monde'.

```
DOSSEG
.MODEL SMALL
.STACK 200H
.DATA
messagebonjour db 'Bonjour, tout le monde',13,10,'$'
mov ax,@data
mov ds,ax
mov ah,9
mov dx,offset messagebonjour
int 01h
mov ah,4ch
int 21h
end
```

Ex 2 : cf. TP : affichage de bonjour, 'prénom' après avoir saisi le prénom
Contenu de PRENOM.ASM

```
dosseg
.model small
.stack 100h
.data
premiermessage    db 'entrez votre prénom :',13,10,'$'
entree                 db 13,14 dup (?)
messagebonjour    db 'bonjour $'
commentva         db ', comment vas-tu ?$'
.code
mov   ax,@data
mov   ds,ax
mov   ah,9
mov   dx,offset premiermessage
int   21h
mov   ah,0ah
mov   dx,offset entree
int   21h
mov   bx,offset entree
inc   bx
mov   cx,0
mov   cl,[bx]
inc   cl
add   bx,cx
```

```
mov   byte ptr [bx],'$'
mov   ah,9
mov   dx,offset messagebonjour
int   21h
mov   dx,offset entree+2
int   21h
mov   dx,offset commentva
int   21h
mov   ah,4ch
int   21h
end
```

Autres exercices :

Ex 1 : Ecrire un programme qui range dans AL le dernier caractère d'une chaîne terminée par un délimiteur 0.

Ex 2 : Ecrire un programme qui inverse une chaîne et l'affiche à l'écran.

Contenu de INVCHAIN.ASM

```
dosseg
.model small
.stack      200h
.data
chaine      db    'ABCDEF '
.code
debut:      mov   ax,@data
            mov   ds,ax
            mov   bx,offset chaine
dernierk: mov     al,[bx]
            cmp   al,20h
            je    inverse
            inc   bx
            jmp   dernierk
inverse:    dec   bx
            mov   dl,[bx]
            mov   ah,2
            int   21h
            cmp   bx,offset chaine
            je    fin
            jmp   inverse
fin:  mov   ah,4ch
            int   21h
            end
```

Contenu de INVCHAIB.ASM

```
dosseg
.model small
.stack      200h
.data
taille      equ   6
chaine1     db    'ABCDEF '
chaine2     db    taille dup (?),'$'
.code
debut:      mov   ax,@data
            mov   ds,ax
            mov   cx,6
            mov   bx,offset chaine1
            add   bx,cx
            dec   bx
            mov   bp,offset chaine2
boucle:     mov   dl,[bx]
            mov   [ds:bp],dl
            inc   bp
            dec   bx
            dec   cx
            cmp   cx,0
            je    fin
            jmp   boucle
fin:  mov   dx,offset chaine2
            mov   ah,9
            int   21h
            mov   ah,4ch
            int   21h
            end
```

2. Fonctions mathématiques

Rappels :

ADD, ADC, SUB, SBB, INC, DEC
MUL, IMUL, DIV, IDIV

Rotations :

ROR, ROL, RCR, RCL

Décalages :
SHL, SHR

Sélection de la taille de données (BYTE PTR)

Exercices de préparation au TP n°2 :

Ex 1 : Multiplication
Contenu du fichier PRODTABL.ASM (à modifier)

```
dosseg
.model small
.stack 200h
.data
tab     db      1,2,3,4,5,6,7
resul   dw      8 dup (?)
.code
debut:  eov     ax,@data
        mov     ds,ax
        mov     cx,6
        mov     bx,offset tab
        mov     di,offset resul
        mov     ah,0
        mov     al,[bx]
boucle: inc     bx
        mov     dx,0
        mov     dl,[bx]
        mul     dx
        dec     cx
        inc     di
        inc     di
        mov     [di],ax
        cmp     cx,0
        je      fin
        jmp     boucle
fin:    mov     ah,4ch
        int     21h
        end
```

Contenu de PRODTA32.ASM

```
dosseg
.model small
.stack 200h
.386
.data
tab     dw      1000,2000,30
resul   dd      3 dup (?)
.code
debut:  mov     ax,@data
        mov     ds,ax
        mov     cx,2
        mov     si,offset tab
        lea     di,resul
        mov     eax,0
        mov     ax,[si]
boucle: inc     si
        inc     si
        mov     ebx,0
        mov     bx,[si]
        mul     ebx
        dec     cx
        mov     [di],eax
        add     di,4
        jcxz    fin
        jmp     boucle
fin:    mov     ah,4ch
        int     21h
        end
```

Ex 2 : Multiplication-division
Contenu du fichier OPERA.ASM

```
      DOSSEG
      .MODEL small
      .STACK 100h
      .DATA
GLOBAL NOMBRE:Word
A   dw    102
B   dw    43
C   dw    40
D   dw    52
reste     dw ?
```

Ex 3 : Affichage des résultats à l'écran
Contenu du fichier ASCII.ASM

```
      DOSSEG
      .MODEL small
      .STACK 100h
      .DATA
GLOBAL nombre:Word
nombre  dw      ?
blanc   db      ' '
dmille  db      ?
mille   db      ?
cent    db      ?
```

```
resultat   dw ?
reponse1   db 'le quotient est : $'
reponse2   db 13,10,'le reste est : $'
      .CODE
debut:
      EXTRN   affiche:PROC
      mov ax,@data
      mov ds,ax
      mov ax,D
      sub ax,3
      push ax
      mov ax,2
      imul A
      push dx
      push ax
      mov ax,B
      imul C
      pop bx
      pop cx
      add ax,bx
      adc dx,cx
      pop bx
      idiv bx
      mov resultat,ax
      mov reste,dx
      mov dx,offset reponse1
      mov ah,09h
      int 21h
      mov ax,resultat
      mov nombre,ax
      call affiche
      mov dx,offset reponse2
      mov ah,09h
      int 21h
      mov ax,reste
      mov nombre,ax
      call affiche
      mov ah,4ch
      int 21h
end
```

Autres exercices :

Ex 1 : Multiplication d'un mot placé en mémoire par 8 en utilisant les fonctions décalage.

Contenu du fichier MULTI8.ASM

```
DOSSEG
.model small
.stack 100h
.data
nombre dw 3
resul dw ?
.code
       mov ax,@data
       mov ds,ax
       mov bx,offset nombre
       mov ax,[bx]
       shl ax,3
```

```
dizaine db       ?
unite   db       ?
clair   dw       ' '
fin     db       '$'
      .CODE
      PUBLIC affiche
affiche PROC
boucle:
      mov ax,' 0'
      mov dmille,al
      mov mille,al
      mov cent,al
      nop
      mov dizaine,al
      mov unite,al
      mov ax,nombre
sup_dmille:
      cmp ax,10000
      jb inf_dmille
      sub ax,10000
      inc dmille
      jmp sup_dmille
inf_dmille:
      cmp ax,1000
      jb positif
      sub ax,1000
      inc mille
      jmp inf_dmille
positif:
      cmp al,100
      jb inf_cent
      sub al,100
      inc cent
      jmp positif
inf_cent:
      cmp al,10
      jnge inf_dix
      sub al,10
      inc dizaine
      jmp inf_cent
inf_dix:
      cmp dmille,'0'
      jne pasnul
      mov dmille,' '
      cmp mille,'0'
      jne pasnul
      cmp dmille,' '
      jne pasnul
      mov mille,' '
      cmp cent,'0'
      jne pasnul
      cmp mille,' '
      jne pasnul
      mov cent,' '
      cmp dizaine,'0'
      jnz pasnul
      cmp cent,' '
      jnz pasnul
      mov dizaine,' '
pasnul:
      add al,30h
      mov unite,al
      mov clair,' '
      mov dx,offset dmille
      mov ah,09h
      int 21h
```

```
        mov resul,ax                        ret
        mov ah,4ch                  affiche ENDP
        int 21h                     end
end
```

3. Traitements de chaînes de données

Rappels :

LODS?, STOS?, MOVS?, SCAS?, CMPS?
CLD, STD
REP, REPNE/REPNZ, REPE/REPZ

Exercices

Ecrire un programme qui :
- saisit le prénom ;
- concatène 'bonjour ' et le prénom dans une nouvelle chaîne en utilisant les fonctions de traitement de données ;
- affiche la chaîne concaténée.
- donne la position du premier 'a' rencontré dans le prénom.

On supposera que la chaîne concaténée ne contiendra pas de caractères au départ.

Contenu de PRECONC.ASM

```
;programme qui effectue la concaténation d'une chaîne bonjour avec
;une chaîne dans laquelle on a entré le prénom
;la chaîne concaténée est ensuite affichée à l'écran
dosseg
.model small
.stack 100h
.data
premiermessage  db 'entrez votre prénom :',13,10,'$'
entree          db 13,14 dup (?)
;il est donc supposé ci-dessus que la chaîne de caractères
; comprend 13 caractères
;en incluant le retour chariot
messagebonjour  db 'bonjour '
concat db 21 dup ('$')
;Ci-dessus, il est intéressant de préparer une zone mémoire
;constituée de $
;
;adresse de la zone mémoire contenant les data dans ds et dans es
;ce qui est indispensable pour utiliser la fonction movs?
```

```
;
.code
mov     ax,@data
mov     ds,ax
mov es,ax
;
;demande d'affichage 'entrez votre prénom ?'
mov     ah,9
mov     dx,offset premiermessage
int     21h
;
;Entrée du prénom
mov     ah,0ah
mov     dx,offset entree
int     21h
;
;initialisation de si, di, df pour utiliser movs?
mov si,offset messagebonjour
mov di,offset concat
cld
;
;copie des 8 caractères de 'bonjour ' dans la chaîne concat
mov cx,8
rep movsb
;
;si contient l'offset du début de la chaîne prénom
mov si,offset entree+2
;
;recherche du nombre de caractères effectifs de la chaîne prénom
;pour initialiser le compteur afin de copier cette chaîne prénom
;dans la chaîne concat
mov bx,offset entree+1
mov cx,0
mov cl,[bx]
rep movsb
mov byte ptr [di],'$'
;ligne ci-dessus facultative puisque l'espace mémoire
; est initialisé avec des '$'
;
;affichage de la chaîne concat
mov ah,09h
mov dx,offset concat
int 21h
;
;recherche du premier 'a' trouvé dans le prénom
mov al,'a'
mov di,offset concat
mov cx,0
mov cl,[bx]
add cx,8
repne scasb
;
;on soustrait à di l'offset de concat pour donner la position
;du 'a', ce qui nous donne la position du 'a' dans la chaîne concat
;puis on soustrait 8 (correspondant à 'bonjour ') pour avoir la
;position dans le prénom proprement dit
mov ax,offset concat
sub di,ax
sub di,8
;
;fin du programme
mov     ah,4ch
int     21h
end
;
```

```
;COMPLEMENT
;Il faudra modifier ce programme pour pouvoir traiter les cas
;où le prénom ne comprend pas de 'a'.
;Si le cas se présente avec le programme actuel, que se passe-t-il ?
```

4. Copies de fichiers

Rappels :

Les interruptions DOS 3ch (création et ouverture d'un nouveau fichier), 3dh (ouverture d'un fichier existant), 3eh (fermeture d'un fichier ouvert), 3fh (lecture d'un fichier ouvert), 40h (écriture dans un fic`ier ouvert).

Exercices de préparation au TP n°3 :

Contenu de PARAM.ASM

```
;listing permettant de lire des paramètres d'entrée de programme
;à l'aide du PSP (Préfixe de Segment de Programme)

;Le PSP est une zone de 256 octets qui se place toujours devant le programme,
;juste avant une limite de segment. Il est installé automatiquement par DOS au
;lancement du programme et précède le programme proprement dit.
;Au moment du chargement d'un programme EXE, DOS remplit certains registres avec
;des valeurs bien particulières. On peut en déduire la valeur du PSP. Voici
le ;contenu des registres avant l'exécution de la première instruction d'un
;fichier EXE :
;CS contient le segment du code courant
;IP l'offset du point de départ du programme
;DS l'adresse complète du PSP. En fait il s'agit du segment du PSP qui démarre
;toujours à l'offset 0 à partir de ce segment.
;ES contient la même chose que DS
;SS le segment de la pile
;SP la taille (fin) du segment de pile
;
;Pour connaître les paramètres éventuellement transmis par le PSP, on a besoin
;d'aller lire les informations suivantes :
;A l'offset 80h, il y a un octet qui contient le nombre de caractères qui ont été
;écrits dans la ligne de commande ;
;A partir de l'offset 81h et pour 127 octets, il y a la chaîne des paramètres du
;programme.
;
dosseg
.model small
.stack 100h
.data
text1 db 'Aucun paramètre n''a été transmis ',13,10,'$'
text2 db 'Les paramètres suivants ont été transmis :',13,10,'$'
.code
mov ax,@data
mov ds,ax
mov ah,09h
lea dx,text1      ; on prépare dx pour l'affichage de text1
mov bl,es:[80h]   ; on stocke le nombre de caractères lus dans bl
cmp bl,0          ; pour savoir si des paramètres ont été transmis
```

```
je dehors         ; on continue à dehors s'il n'y a pas de paramètres
lea dx,text2      ; on prépare l'affichage de text2
int 21h
mov bh,0          ; on met les bits de poids fort de bx à 0
mov byte ptr es:[81h+bx],'$'  ; on met '$' à la fin de la chaîne
                                    ; du PSP pour pouvoir l'afficher
mov dx,81h  ; pointe sur la chaîne des paramètres du programme
```

```
mov ah,09h
push es
pop dq      ; on met dans ds l'adresse du segment du PSP pour
            ; pouvoir l'afficher à l'aide de la fonction 09h du DOS
dehors:
int 21h
mov ah,4ch
int 21h
end
```

- Programme de copie de fichiers

Contenu de TP9.ASM

```
nombre_octets equ 1000
dosseg
.model small
.stack 200h
.data
questionsource db 'nom du fichier source : ',13,10,'$'
questiondestination db 13,10,'nom du fichier destination :',13,10,'$'
nomsource db 15,17 dup (0)
nomdestination db 15,17 dup (0)
fichier db nombre_octets dup ('$')
.code
mov ax,@data
mov ds,ax
;Saisie des noms des fichiers source et destination
mov ah,09h
lea dx,questionsource
int 21h
mov ah,0ah
lea dx,nomsource
int 21h
lea bx,nomsource+1
mov dl,[bx]
mov dh,0
add bx,dx
inc bx
mov byte ptr [bx],0
mov ah,09h
lea dx,questiondestination
int 21h
mov ah,0ah
lea dx,nomdestination
int 21h
lea bx,nomdestination+1
mov dl,[bx]
mov dh,0
add bx,dx
inc bx
mov byte ptr [bx],0
;ouverture du fichier source
mov ax,3d00h
lea dx,nomsource+2
int 21h
mov bx,ax       ;bx contient le handle
mov ah,3fh
mov cx,nombre_octets
lea dx,fichier
int 21h
push ax      ;sauvegarde du nombre d'octets lus dans la pile
mov ah,3eh
int 21h
;création du fichier destination et écriture à l'intérieur
mov ah,3ch
lea dx,nomdestination+2
```

```
mov cx,0
int 21h
mov bx,ax       ;sauvegarde du handle
mov ah,40h
pop cx          ;récupération du nombre d'octets lus
lea dx,fichier
int 21h
mov ah,3ch
int 21h
;fin du programme
mov ah,4ch
int 21h
end
```

5. Entrées-sorties

Rappels :

IN, OUT
Les registres de configuration du 8250

Exercices de préparation au TP n°4 :

cf. TP n°4 question sur le 8250

Contenu de EMIS.ASM

```
com EQU 3f8h
comp1 equ 3f9h
comp2 equ 3fah
comp3 equ 3fbh
comp4 equ 3fch
comp5 equ 3fdh
dosseg
.model small
.stack 200h
.data
discours db 'Emission en assembleur'
fin_chaine db 13,10,'$'
echap db 27
car_emis db ?
.code
mov ax,@data
mov ds,ax
lea dx,discours
mov ah,09h
int 21h
mov al,80h
mov dx,comp3
out dx,al
mov al,0
mov dx,com
out dx,al
mov dx,comp1
out dx,al
mov al,3
```

Contenu de REC.ASM

```
com EQU 3f8h
comp1 equ 3f9h
comp2 equ 3fah
comp3 equ 3fbh
comp4 equ 3fch
comp5 equ 3fdh
dosseg
.model small
.stack 200h
.data
discours db 'Reception en assembleur'
fin_chaine db 13,10,'$'
echap db 27
.code
mov ax,@data
mov ds,ax
lea dx,discours
mov ah,09h
int 21h
mov al,80h
mov dx,comp3
out dx,al
mov al,0
mov dx,com
out dx,al
mov dx,comp1
out dx,al
mov al,3
mov dx,comp3
```

```
mov dx,comp3
out dx,al
mov al,'a'
boucle:cmp al,echapje fin
call attente
mov ah,01
int 21h
mov dx,com
out dx,al
cmp al,13
jne boucle
lea dx,fin_chaine
mov ah,09h
int 21h
jmp boucle
fin:
mov ah,4ch
int 21h
attente PROC
push ax
boucle_attente:
mov dx,comp5
in al,dx
and al,20h
cmp al,20h
jne boucle_attente
pop ax
ret
endp
end
```

```
out dx,al
mov al,'a'
boucle:cmp al,echap
je fin
call attente
mov dx,com
in al,dx
mov dl,al
mov ah,02h
int 21h
cmp al,13
jne boucle
lea dx,fin_chaine
mov ah,09h
int 21h
jmp boucle
fin:
mov ah,4ch
int 21h
attente PROC
push ax
boucle_attente:
mov dx,comp5
in al,dx
and al,1
cmp al,1
jne boucle_attente
pop ax
ret
endp
end
```

- Aspect 8255

contenu de PARA.ASM (chenillard)

```
porta equ 02e0h
portb equ 02e1h
portc equ 02e2h
ctrlw equ 02e3h
dosseg
.model small
.stack 200h
.code
init:   mov al,91h
mov     dx,ctrlw
out     dx,al
debut:  mov dx,portc
        in  al,dx
        and     al,07h
        cmp     al,0
        je      debut
        cmp     al,1h
        je      copieasurb
        cmp     al,3h
        je      rotbd
        cmp     al,2
        je      rotbg
        and     al,4h
        cmp     al,4
        jne     debut
        mov     ah,4ch
        int     21h
        int     03h
copieasurb: mov dx,porta
        in      al,dx
        mov     dx,portb
        out     dx,al
        jmp     debut
rotbd:  mov     dx,portb
        in      al,dx
        ror     al,1
        jmp     tempo_1s
eti1:   mov     dx,portb
        out     dx,al
        jmp     debut
```

contenu de PARA1.ASM (1ère question)

```
portb equ 02e1h
portc equ 02e2h
ctrlw equ 02e3h

dosseg
.model small
.stack 200h
.code
init:   mov al,91h
mov     dx,ctrlw

out     dx,al
debut:  mov dx,portc
        in  al,dx
        and     al,0fh
        cmp     al,0
        je      copieasurb
        mov     ah,4ch
        int     21h
copieasurb: mov dx,porta
        in      al,dx
        mov     dx,portb
        out     dx,al
        jmp     debut
        end
```

```
rotbg:  mov     dx,portb
        in      al,dx
        rol     al,1
        jmp     tempo_1s
tempo_1s:mov    bx,2
eti2:   mov     cx,0ffffh
compt:  loop    compt
        dec     bx
        cmp     bx,0
        jne     eti2
        jmp     eti1

        end
```

6. Retour sur les adresses

Révision de ce que sont :
- nom et contenu pour les registres ;
- adresse et contenu pour les cases mémoire ;
- numéro et contenu pour les ports d'entrée-sortie.

Instructions permettant de faire tout cela
MOV
MOV[ ]
IN/OUT

Exercices :

1) Stocker 03H dans BX
Rép : MOV BX,3

2) Stocker 03H à l'adresse physique 32157H
Rép : mov byte ptr 3000:2157,3

3) Stocker 03h dans le port n° 278H

Rép : MOV DX, 278H
MOV AL,3
OUT DX,AL

4) Lire le contenu de CX et le stocker dans l'accumulateur (AX ou AL)
Rép : MOV CX,AX

5) idem pour CH
Rép : MOV CH,AL

6) Lire le contenu de la mémoire d'adresse physique 32157H

MOV AX,3215H
MOV DS,AX
MOV BX,7
MOV AL,DS:[BX]

7) Lire le contenu du port n° 29BH
Rép : MOV DX,29BH
IN AL,DX

Références bibliographiques

- Programmer en assembleur sur PC - H. Schakel - Micro Applications, Paris, 1992.

- Guide de l'utilisateur Turbo-Assembleur - Borland, 1990.

- Guide de l'utilisateur Turbo-Debugger - Borland, 1990.

- 8086-8088 - J.W. Coffron - Sybex, Paris, 1984.

- Le livre d'or PC - M. Althaus - Sybex, Paris, 1992.

Petites colles

Ce sont les sujets de colles qui ont pu être posés en TD en fonction de l'avancement de l'enseignement

Colle n°1 :

Sujet :
On suppose connu l'état actuel du contenu du registre BX (0000H) et des cases mémoire d'adresse :
<DS:0000> contient 10h
<DS:0001> contient 25h
<DS:0002> contient 30h

Qu'est-ce qui aura changé après l'exécution de chacune de ces instructions indépendantes les unes des autres :
1) MOV BX,'$'

2) MOV [BX], '$'

3) MOV BYTE PTP [BX], '$'

Rem : le code ASCII de '$' est 24h.

Ecrire un programme assembleur qui :
- permet de saisir le nom de famille (20 caractères maxi) ;
- permet de saisir le prénom (10 caractères maxi) ;
- affiche à l'écran 'Enchanté de faire votre connaissance, *prénom nom_de_famille*'.

Variante de la seconde partie :
Ecrire un programme qui lit une chaîne de caractères au clavier (maxi 32 caractères) et la stocke dans des cases mémoire à partir de l'offset chaine1, puis la recopie au label chaine2 à l'aide de l'instruction MOVSB.

Réponses :

1) BX=0024H, les mémoires sont inchangées.

2) BX est inchangé, les cases mémoire contiennent respectivement 24h, 00h, 30h.

3) BX est inchangé, les cases mémoire contiennent respectivement 24h, 25h et 30h.

Contenu du fichier COLLEA.ASM ; les fichiers COLLEB et COLLEC.ASM contiennent des variantes.

```
;*****************************************************************
;*Programme permettant de :
;*- rentrer le nom de famille (20 caractères max)
;*- rentrer le prénom (10 caractères max)
;*- afficher 'Enchanté de faire votre connaissance, "prénom" "nom"
;*****************************************************************
dosseg
.model small
.stack 200h
.data
msg_nom_famille db 'Veuillez donner votre nom de famille :',13,10,'$'
nom_famille db 21,22 dup (?) ; signifie que l'on initialise un
;caractère à 21 (nombre de caractère +1 pour le retour chariot)
;suivi de 22 caractères initialisés à "rien". Le total est donc
;de 23 caractères (20 pour le nom + 2 pour la fonction 0ah + 1
;pour le retour chariot)
msg_prenom db 13,10,'Veuillez donner votre prénom : ',13,10,'$'
prenom db 11,12 dup (?)
msg_fin db 13,10,'Enchanté de faire votre connaissance, $'
.code
```

```
mov ax,@data
mov ds,ax
;
;Affichage de 'veuillez donner votre nom de famille'
lea dx,msg_nom_famille ; mov dx, offset msg_nom_famille
mov ah,09h
int 21h
;
;Entrée du nom de famille
mov ah,0ah
lea dx,nom_famille
int 21h
;
;appel d'une procédure (cf fin du programme) permettant de
;placer le caractère '$' à la fin de la chaîne de caractère
;qui vient d'être rentrée par la fonction DOS 0ah
call pos_dollar ;nécessite l'offset de la chaîne rentré au format
            ;0ah dans dx, les registres ne sont pas modifiés
;
;Affichage de 'Veuillez donner votre prénom'
lea dx,msg_prenom
mov ah,09h
int 21h
;
;Entrée du prénom
mov ah,0ah
lea dx,prenom
int 21h
;
;Appel de la procédure permettant de placer le '$'
call pos_dollar
;
;Affichage de 'Enchanté de faire votre connaissance :'
lea dx,msg_fin
mov ah,09h
int 21h
;
;Affichage du prénom
lea dx,prenom+2
mov ah,09h
int 21h
;
;Positionnement d'un espace devant la chaîne nom_de_famille
;Cet espace sera écrit sur le second octet de la chaîne
;C'est-à-dire qu'il remplacera l'octet contenant le nombre de
;caractères réellement lus par la fonction DOS 0ah.
lea bx,nom_famille+1 ;positionnement d'un espace
mov byte ptr [bx],20h
;
;Affichage du nom de famille à partir du second octet (contenant
;l'espace) de la chaîne
lea dx,nom_famille+1
mov ah,09h
int 21h
;
;Fin du programme principal
mov ah,4ch
int 21h
;
;PROCEDURE PERMETTANT DE POSITIONNER UN DOLLAR $ A LA FIN
;D'UNE CHAINE DE CARACTERES
;
pos_dollar PROC
;Sauvegarde du contenu des registres ax et bx dans la pile pour
;pouvoir les utiliser dans la procédure sans que cela n'influe
```

```
; sur le contenu des registres que l'on utilise dans le programme
;principal (cf variables locales)
push ax
push bx
mov bx,dx
inc bx
mov al,[bx]
mov ah,0
add bx,ax
inc bx
mov byte ptr [bx],'$'
;restauration des registres à la fin de la procédure avant de
;redonner la main au programme principal
pop bx
pop ax
ret
endp
;ret et endp indiquent la fin de la procédure
;end indique la fin du programme
end
```

Réponse de la variante
Contenu de COLMOVSB.ASM

```
dosseg
.model small
.stack 200h
.data
chaine1 db 35 dup (?)
chaine2 db 32 dup (?)
mess db 'Entrez une chaîne, limitée à 32 caractères : ',13,10,'$'
.code
mov ax,@data
mov ds,ax
mov es,ax
;
;affichage du message du début
mov ah,09h
lea dx,mess
int 21h
;
;entrée de la chaîne
mov ah,0ah
lea bx,chaine1
mov byte ptr [bx],33; placement du nbr maxi de caractères
mov dx,bx
int 21h
;
;copie de la chaîne
inc bx
mov cl,[bx]
mov ch,0
lea si,chaine1+2
lea di,chaine2
rep movsb
mov ah,4ch
int 21h
end
```

Colle n°2 :

1) Je dois aller écrire une valeur dans l'octet mémoire d'adresse physique 184B7H/223C7H/1D58CH. Le contenu du registre de

segment est 1472H/211BH/1B72H, quelle doit être la valeur contenue dans le registre d'offset pour me permettre d'adresser cette mémoire ?

Rép : 3D97H/1217H/1E6CH.

2) Le contenu du registre de segment est 3000H/4000H/D000H. Puis-je adresser la case mémoire d'adresse physique 42787H/57285H/D118CH. Pourquoi ?

Rép : non car dépassement des 64 ko maxi autorisés pour un segment/non/oui.

3) Je dois stocker dans l'accumulateur l'octet contenu dans le port d'entrée n° 35H/1B8H/1D5H.

Rép : IN AL,35H / MOV DX,1B8H / IN AL,DX MOV DX,1D5H IN AL,DX

4) Je veux écrire 05H/07H/0BH dans le registre dl/bh/ah en écrasant le moins de valeurs possibles.

Rép : MOV DL,05H MOV BH,7 MOV AH,0BH

5) Je veux lire le contenu de l'octet mémoire dont l'adresse physique est 5B18CH/4C174H/125BCH et le stocker dans l'accumulateur en écrasant le moins de valeurs possibles.

6) Explicitez ces lignes de programme et dites lesquelles sont impossibles et pourquoi. L'objectif est d'écrire 28H dans l'octet mémoire d'adresse physique 2CB45H.

a)
```
mov ax,2B83H
mov ds,ax
mov bx,1315H
mov byte ptr ds:[bx],28H
```

b)
```
mov ax,2B83H
mov es,ax
mov cx,1305H
mov es:[cx],28
```

c)
```
mov ax,2CB45H
mov [ds],28H
```

Rép : a) correct, le ds: est facultatif ; b) offset incorrect, cx ne peut être mis entre crochets, 28 n'est pas 28H ; c) impossible

7) Ecrire 25H/38H dans le contenu du port n° 5D8H/27H/1CH.

GTR 1è Année Assembleur juin 94

N. B. : L'usage des calculatrices est interdit
Aucun document n'est autorisé

1) Vous proposerez un classement par catégories des registres du **8086**. Puis vous listerez les registres du 8086 et expliquerez leur(s) utilisation(s) possible(s).

2) On veut adresser la case mémoire d'adresse physique 7BC2CH. Sachant que le registre d'offset contient F28CH, combien doit contenir le registre de segment ?

3) Quelle différence y a-t-il entre les deux instructions suivantes :

```
MOV BX,AX     et     MOV [BX],AX
```

4) Donnez l'équivalent Assembleur des instructions Pascal suivantes :

```
dx := bx;
al := al + bl;
bx := bx + 1;
si := si - 10h;
ax := ax * bx;
```

5) Que réalisent les instructions suivantes ?

```
.DATA
tab dw 20 dup (?)
lec db 17,35 dup (?)
valeur db 8,15
```

6) Dites comment vous réaliseriez en assembleur : la lecture de l'octet situé à l'adresse physique 23B5CH et le stockage dans l'accumulateur. Vous vous assurerez de ne pas écraser de valeurs inutilement.

7) Parmi les lignes de programme suivantes, dites lesquelles sont impossibles :

```
1) mov ds,25787h    2) in al,35ch    3)       out
```

```
al,dx
4) mov dx,257h          5) mov al,37ch       6)
in ax,dx
7) mov [cx], 38h    8) mov ax,ds:[bx]    9)   mov
cx,byte ptr [bx]
10) mov byte ptr [bx], 18h                11)   in
al,36h
```

8) Commentez les lignes suivantes en mettant en évidence les erreurs. L'objectif est d'écrire 37h dans l'octet mémoire d'adresse physique 35783H

```
mov ds,35781h
mov bx,2
mov [bx],37
```

Vous proposerez la solution la plus simple possible qui fonctionne.

9) Interprétation d'un programme assembleur. Vous indiquerez les évolutions des contenus des registres en déroulant pas à pas ce programme.

val1 est à l'offset 1fh
val2 est à l'offset 1dh
val3 est à l'offset 4

MODULE	DUMP							
mov di,offset val2	0000:0000	18	25	74	87	98	96	AF	D5
mov al,[di]	0000:0008	54	A1	B5	00	00	DE	D6	D6
mov cx,val1	.								
add cx,ax	.								
mov bx,val3	.								
mov ah,-18	0020:0000	25	00	68	03	25	18	AA	36
mov si,offset val3	0020:0008	74	14	63	1A	73	D8	3B	1E
mov dl,[si+8]	0021:0000	B3	9F	C9	14	25	12	31	02
mov dh,[di+4]	0021:0008	AB	17	FF	66	6A	2E	00	36
	0022:0000	18	9F	C9	00	00	00	00	00
	0022:0008	00	17	FF	66	6A	2E	00	00

AX 12C5	SI DF36	DS 0020
BX BCDE	DI 456B	ES 0100
CX 456B	BP 0010	SS 76E8
DX 457E	SP 0200	CS 76E0

10) Ecrire un programme qui :
- saisit un prénom ;
- place '$' à la fin du prénom ;
- teste si l'initiale est en majuscule ;
- si oui, pas de modification ;
- si non, l'initiale doit être transformée en majuscule ;
- affiche à l'écran la chaîne modifiée.

Rappels :les codes ASCII des minuscules commencent à 61H et se suivent dans l'ordre alphabétique ;

les codes ASCII des majuscules commencent à 41H et se suivent dans l'ordre alphabétique.

Quelques sauts

JCXZ : saut si CX == 0
JE/JZ : saut si égal
JL/JNGE : saut si plus petit
JLE/JNG : saut si plus petit ou égal
JNE/JNZ : saut si différent
JNL/JGE : saut si plus grand ou égal
JNLE/JG : saut si plus grand

Interruptions DOS

09H : Affichage d'une chaîne de caractères :

Affiche une chaîne de caractères terminée par '$' et située à l'adresse physique exprimée par DS:DX.

02H : Affichage d'un caractère :

Affiche le caractère dont le code ASCII est stocké dans DL.

01H : Saisie d'un caractère :

Le code ASCII du caractère lu est stocké dans AL.

0AH : Saisie d'une chaîne de caractères :

Saisit une chaîne au clavier et la stocke dans un tampon d'adresse physique exprimée par DS:DX. La saisie s'arrête après avoir frappé un retour chariot au clavier.

Le tampon destiné à stocker la chaîne saisie a le format suivant :

1er octet : nombre maxi de caractères à saisir donné par l'utilisateur ;

2nd octet: nombre de caractères effectivement saisis renvoyé par la fonction ;

A partir du 3ème octet : zone de stockage de la chaîne saisie proprement dite.

Fonctions de traitement des données

LODS?; MOVS?, STOS?, SCAS?, CMP? : ? peut être remplacé par B, W, ou D selon le format des données traitées.

LODS? :

charge dans l'accumulateur le contenu de la mémoire d'adresse pointée par DS:SI. Ensuite SI est incrémenté si DF est à zéro (DF mis à zéro à l'aide de CLD) ou décrémenté si DF est à un (DF mis à 1 par STD).

STOS? :

stocke le contenu de l'accumulateur dans la mémoire d'adresse ES:DI. Ensuite DI est incrémenté si DF est à zéro ou décrémenté si DF est à un.

MOVS? :

transfère le contenu de la mémoire d'adresse physique DS:SI dans la mémoire d'adresse ES:DI. Ensuite SI et DI sont incrémentés si DF est à zéro ou décrémentés si DF est à un.

SCAS? :

compare le contenu de l'accumulateur avec le contenu de la mémoire d'adresse ES:DI.Ensuite DI est incrémenté si DF est à zéro ou décrémenté si DF est à un.

CMPS? :

compare deux chaînes stockées respectivement en DS:SI et ES:DI. Ensuite SI et DI sont incrémentés si DF est à zéro ou décrémentés si DF est à un.

Correction de l'examen de juin 1994

1) Registres à usage généraux
AX : accumulateur (MUL, DIV, IN, OUT)
BX : base (possible d'adresser la mémoire)
CX : compteur (REP, LOOP)
DX : données (INT DOS)
SI : index source (traitement de chaînes de caractères, adressage mémoire)
DI : index destination (traitement de chaînes de caractères, adressage mémoire)
BP : pointeur de base (adressage mémoire possible)
SP : pointeur de pile
Les registres AX, BX, CX, DX peuvent être coupés en deux registres 8 bits AH, AL, BH, BL, CH, CL, DH, DL. Le registre de segment associé est DS sauf pour BP et SP qui utilisent SS.
Registres de segment
CS : code ; DS: données ; SS: pile ; ES : extra segment.
Registres de contrôle du microprocesseur :
IP : pointeur d'instruction
FLAGS : registre des indicateurs.

2) Contenu de DS : 6C9A.

3) On stocke le contenu de AX dans BX ; on stocke le contenu de AX dans les deux octets mémoire pointés par BX (poids faible) et par BX+1 (poids fort).

4)
```
mov dx,bx
add al,bl
inc bx
sub si,10h
mul bx
```

5) Réservation de 20 mots de 16 bits dans la mémoire, vides, appelés tab.

Réservation, à l'étiquette lec, d'un octet initialisé à 17, suivi de 35 octets vides.

Réservation, à l'étiquette valeur, d'un octet contenant 8 et d'un octet contenant 15.

6) mov al,25B5H:000CH est une solution simple

alternative :
```
mov ax,23B5h
mov ds,ax
mov bx,000ch
mov al,[bx] ou mov al,ds:[bx]
```

7) lignes impossibles : 1-2-3-5-7-9

8) mov ds,35781h; impossible car valeur sur 20 bits, l'adresse physique doit être séparée en segment et offset, mov ds,valeur est impossible.
mov bx,2 ; pas de problème
mov [`x],37; il faut mettre 37h et byte ptr
Solutions possibles : mov byte ptr 3578H:0003H,37H ou
```
mov ax,3578h
mov ds,ax
mov bx,3
mov byte ptr [bx],37h
```

9) AX : 122E puis EE2E ; BX : 1825 ; CX : 1836 puis 2A64 ; DX : 4525 puis 9F25 ;
SI : 0004 ; DI : 001D

Contenu de COLLE94.ASM

```
dosseg
.model small
.stack 200h
.data
toto dw 25h,368h
val3 dw 1825h
toto1 dw 036aah,1474h,1a63h
totofin db 73h,0d8h,3bh,1eh,0b3h,9fh
totofins db 0c9h,14h,25h,12h,30h,02h
totofint db 0abh,17h,0ffh,66h,6ah
val2 dw 002eh
val1 dw 1836h,0c99fh,00h
.code
;initialisation   des   variables   commemov si,offset val3
demandé
mov ax,@data
mov ds,ax
mov ax,0100h
mov es,ax
mov ax,12c5h
mov bx,0bcdeh
mov cx,456bh
mov dx,457eh
mov bp,0010h
mov si,0df36h
mov di,456bh

;début des instructions de l'examen
mov di,offset val2
mov al,[di]
mov cx,val1
add cx,ax
mov bx,val3
mov ah,-18
mov dl,[si+10h]
mov dh,[di+4]

;fin du programme
mov ah,4ch
int 21h
end
```

10) contenu de col9410.asm

```
dosseg
.model small
.stack 200h
.data
accueil  db  'bonjour,  veuillez  entrer
votre prénom : ',13,10,'$'
rc db 13,10,'$'
chaine db 13,14 dup (?)
.code
mov ax,@data
mov ds,ax

mov ah,09h; affichage du message
;d'accueil
mov dx,offset accueil
int 21h

mov ah,0ah      ;saisie du prénom
lea dx,chaine
int 21h

lea bx,chaine+1 ;placement d'un dollar à
la fin de la chaîne
mov cl,[bx]
mov ch,0
add bx,cx
inc bx
mov byte ptr[bx],'$'
lea bx,chaine+2
mov al,[bx]
cmp al,61h
jl majuscule   ;saut si initiale déjà en
majuscule
sub al,20h
mov byte ptr[bx],al
majuscule:
lea dx,rc
mov ah,09h
int 21h
lea dx,chaine+2
int 21h

mov ah,4ch      ;fin du programme
int 21h
end
```