Solution TD 5 : Architecture des ordinateurs
2020/2021

Question 1 : Expliquez les déclarations assembleur suivantes ?

1.1
```
ASSUME DS:Data
Data SEGMENT
    ch_in    DB " C'est quoi tous ces blancs ?",0
    ch_out   DB 64 DUP(?)
Data ENDS
```

- **Bloc de données (segment)**
- **Réservation d'espace mémoire pour stocker des valeurs, d'initialisation,**
- **Localisation de ces espaces en mémoire (adresse) par des identificateurs (#variables).**
- **DB = unité octet (Byte),**
- **" " définit la valeur d'une chaîne de caractères, taille déterminée automatiquement 0 = un octet de plus, de valeur 0 (Sert à localiser la fin)**
- **64 dup = donne une taille (64 octets), et la valeur (? = non initialisée)**

1.2
```
ASSUME SS:Pile
  Pile SEGMENT STACK
       DW 64 DUP(?)
  vide EQU THIS WORD
  Pile ENDS
```

Bloc de pile (segment)
Réserver de la place mémoire pour la pile
Localiser le bas (sous la pile)

1.3 A quoi servent les instructions suivantes ?

```
        ASSUME CS:Code
         Code SEGMENT
debut:     MOV AX,Data
           MOV DS,AX
           MOV AX,Pile
           MOV SS,AX
           MOV SP,vide
           MOV BP,SP
           ...
fin:       MOV AH,4CH
           INT 21H
          Code ENDS
END debut
```

- **Bloc d'instructions (segment)**
- **Les premières lignes sont l'initialisation du registre DS,**
- **DS est la partie poids fort des adresses des données en bloc données**
- **Les suivantes sont l'initialisation du registre SS,**
- **CS est la partie poids fort des adresses des données en pile**
- **Ensuite initialisation des pointeurs de pile**
- **Les dernières instructions sont l'appel à une fonction du DOS, qui lui rend la main (fin de programme)**
- **END début définit le point d'entrée du programme (la 1ère instruction)**

Question 2 : Ecrire un programme qui supprime les espaces au début d'une chaîne de caractère :le programme remplit ch_out à partir de ch_in en ayant supprimé les espaces au début. On suppose que les déclarations utilisée dans la question 1.1 ont été faites et que l'on peut les utiliser. Attention : la chaîne fournie pourrait être tout autre (mais se terminerait par 0).

```
        MOV BX,offset ch_in
espc:   MOV AL,[BX]          ; on passe tous les espace
        CMP AL,0             ; si la chaîne est vide finir
        JE copy
        CMP AL," "
        JNE memo
        INC BX
        JMP espc

memo:   MOV CX, BX           ; une sauvegarde de l'@ source
        MOV BX,offset ch_out
        MOV DX, BX           ; une sauvegarde de l'@ destination

copy:   MOV BX,CX ; copie des caractères
        MOV AL,[BX]
        MOV BX,DX
        MOV [BX],AL
        CMP AL,0
        JE fin
        INC CX
        INC DX
        JMP copy
        …
fin: …
```

Question 3

3.1 Donnez l'ensemble des directives et des instructions à l'écriture d'un programme assembleur. On ne demande que le squelette d'un programme : ne pas détailler les données du segment de données (marquer juste un commentaire **; ici déclaration des données)** pas plus les instructions du programme (marquer juste un commentaire **; ici déclaration du programme)**

```
ASSUME  DS :data, CS :code
data   SEGMENT
        ; ici déclaration des données
data ENDS
code SEGMENT
début :  MOV    AX, data
         MOV   DS, AX
            ; ici instructions du programme
        MOV    AH, 4C
        INT  21H
Code   ENDS
END   début
```

3.2 Donnez les directives pour la déclaration de CHAIN initialisée à ''Tableau de !$'', d'un tableau TAB de 1236 caractères (non prédéfinis), de la variable N initialisée à la taille de TAB, de la variable I qui va de 0 à N. Les directives sont les lignes du programme que vous placeriez à la place du commentaire précédant dans le segment de données.

```
CHAIN  DB 'Tableau de   !$'
TAB      DB   1236  dup( ?)
 N         DW  1236
 I          DW  0
```

3.3 Ecrire un programme qui détermine la taille de la chaine CHAIN, et range cette taille dans N. la chaine CHAIN se termine par le caractère '$'. On suppose ici que l'on a fait les déclarations des questions 3.1, 3.2 Et que l'on a donc à écrire que les lignes que vous placerez à la place du commentaire précédant dans le segment de code

```
        MOV   BX  offset  CHAIN
        MOV  CX, 0
suite :  MOV  AL, [BX]
         INC     BX
         ADD   CX, 1
          CMP   AL, '$'
          JNE     suite
          MOV   N, CX
```

Question 4 Dites ce que le programme suivant range dans AL en fin d'exécution, dites ce que cela represente et donnez la valeur

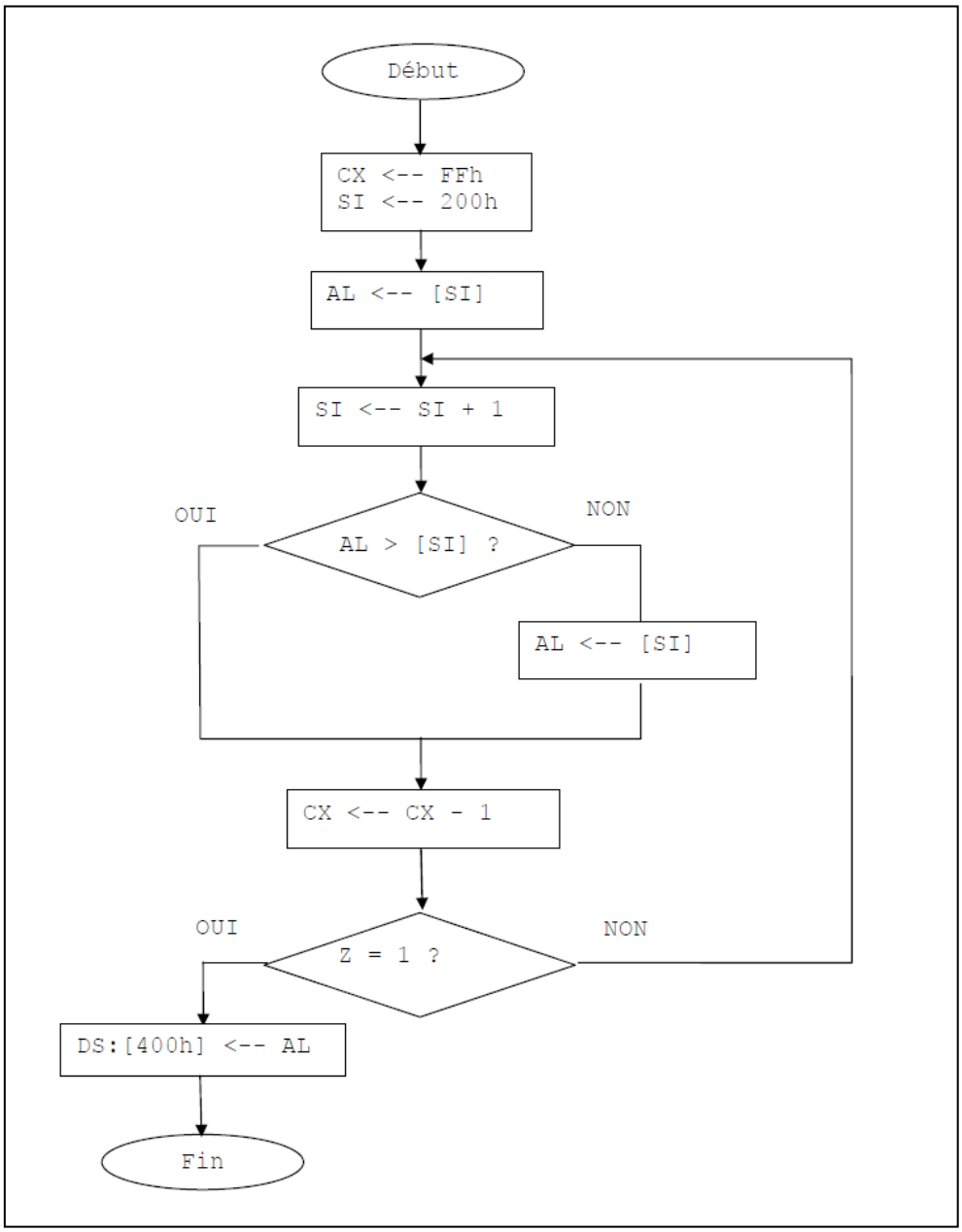
```
ASSUME  CS : CODE,  DS :DATA
```

```
DATA        SEGMENT
  TAB       DB  18, 11, 29, 7, 15, 34, 42, 89, 8, 76, 4, 61, 43, 12, 6
  NELT      DW  14
DATA        ENDS

CODE        SEGMENT
 Tri :      MOV   AX, DATA
            MOV   DS, AX
            MOV   BX, offset  TAB
            MOV   AL, [BX]
            INC   BX
            MOV   CX, NELT
 Boucle :   MOV    AH, [BX]
            CMP    AL, AH
            JB     Suite              ; Test <
            MOV    AL, AH
  Suite :   INC    BX
            DEC    CX
            JNE    Boucle             ; Test ≠
  Fin :     MOV    AH, 4CH
            INT    21H
  CODE      ENDS
            END    Tri
```

- **En fin du programme AL contient le minimum du tableau.**
- **Dans ce cas, AL contient la valeur 4**

Question 5 **Ecrire un programme qui permet de déterminer le maximum dans un tableau d'octets mémoire de longueur 100h et débutant à l'adresse [200h], le résultat sera placé à l'adresse [400h]. voici l'organigramme algo)**



Le compteur CX a été initialisé à : N-1 = 100h - 1 = FFh ; En fait le registre AL prend la première valeur du tableau, c à d [200h] ; Puis il est comparé avec les N-1 valeurs suivantes. Pour chaque comparaison, CX prend une valeur, pour la comparaison de la dernière valeur du tableau, la valeur de CX et égale à 1, donc la valeur initiale de CX est N-1 : CX = N-1, N-2, ... 2, 1. Le programme en langage assembleur 8086 : (partie la plus significative)

```
MOV CX, FFh
MOV SI, 200h
```

```
        MOV AL, [SI]
Etq2 :  INC SI
        CMP AL, [SI]
        JAE Etq1
        MOV AL, [SI]
Etq1 :  DEC CX
        JNZ Etq2
        MOV [400], AL
        HLT
```