- TOUS documents PERSONNELS autorisés
- TOUT échange (documents ou autres biens) INTERDIT (effaceur, etc..)
- TOUS équipements électroniques ETTEINTS (phones, calculatrices . .)

Exercice 1 (5 * 1 Pts) *(Temps_référence : 15 mn)*

Pour chacun des 5 cas suivants, trouver le résultat de chaque instruction puis indiquer l'instruction « **intruse** », c'est-à-dire qui aboutit à un **résultat** différent de celui des autres:

Cas_1/	MOV AX , 0 ;	AND AX , 0 ;	XOR AX , AX ;	**OR AX, 0 ;**
Cas_2/	MOV AX , 0 ;	SHL AX , 16 ;	SHL AL , 8 ;	**ROR AX, 8 ;**
Cas_3/	MOV AX , 1 ; OR AX, AX ;	MOV AX , 0 ; INC AX ;	MOV AX , 0 ; INC AL ;	**MOV AX , 0 ;** **INC AH ;**
Cas_4/	MOV AX , 1 ; INC AX ;	MOV AX , 0 ; ADD AX, 2 ;	**MOV AX , 2 ;** **DEC AX ;**	XOR AX , AX ; ADD AX, 2 ;
Cas_5/	MOV AX , 10 ; MUL AX, 2 ;	MOV AX , 10 ; SHL AX, 1 ;	MOV AX , 14H ; SHR AX, 1 ;	**MOV AL , 14H ;** **SHR AX, 1 ;**

NB : comparer le résultat des **4 colonnes**, **cas par cas**.

Exercice 2 (6 Pts) *(Temps_référence : 30 mn)*

1)- Proposer un programme assembleur '80286' qui effectue les opérations suivantes :

* charge successivement 100 données, de 1 octet chacune, à partir d'une adresse (SI) quelconque ;

* identifie le **5ème bit** de chaque donnée chargée :

  * si ce 5ème bit est nul, la donnée est stockée dans une zone RAM pointée par DI quelconque ;

  * si ce 5ème bit est non nul, la donnée est stockée en pile.

NB : On suppose pour cette première question l'usage des registres (BX) et (DX) interdit.

2)- Transformer ce programme pour pouvoir récupérer dans (BX) le nombre de données stockées en pile, et dans (DX) le nombre de données stockées en zone DI.

1)-

```
                MOV CX, 100
REPRISE :       MOV AL , [SI]
                MOV AH, AL          ; Sauvegarde
                AND AL, 10H
                JZ 5eme_Bit_Nul
                PUSH AH
                JMP SUITE
5eme_Bit_Nul:   MOV [DI], AH
                INC DI
SUITE:          INC SI
                LOOP REPRISE
```

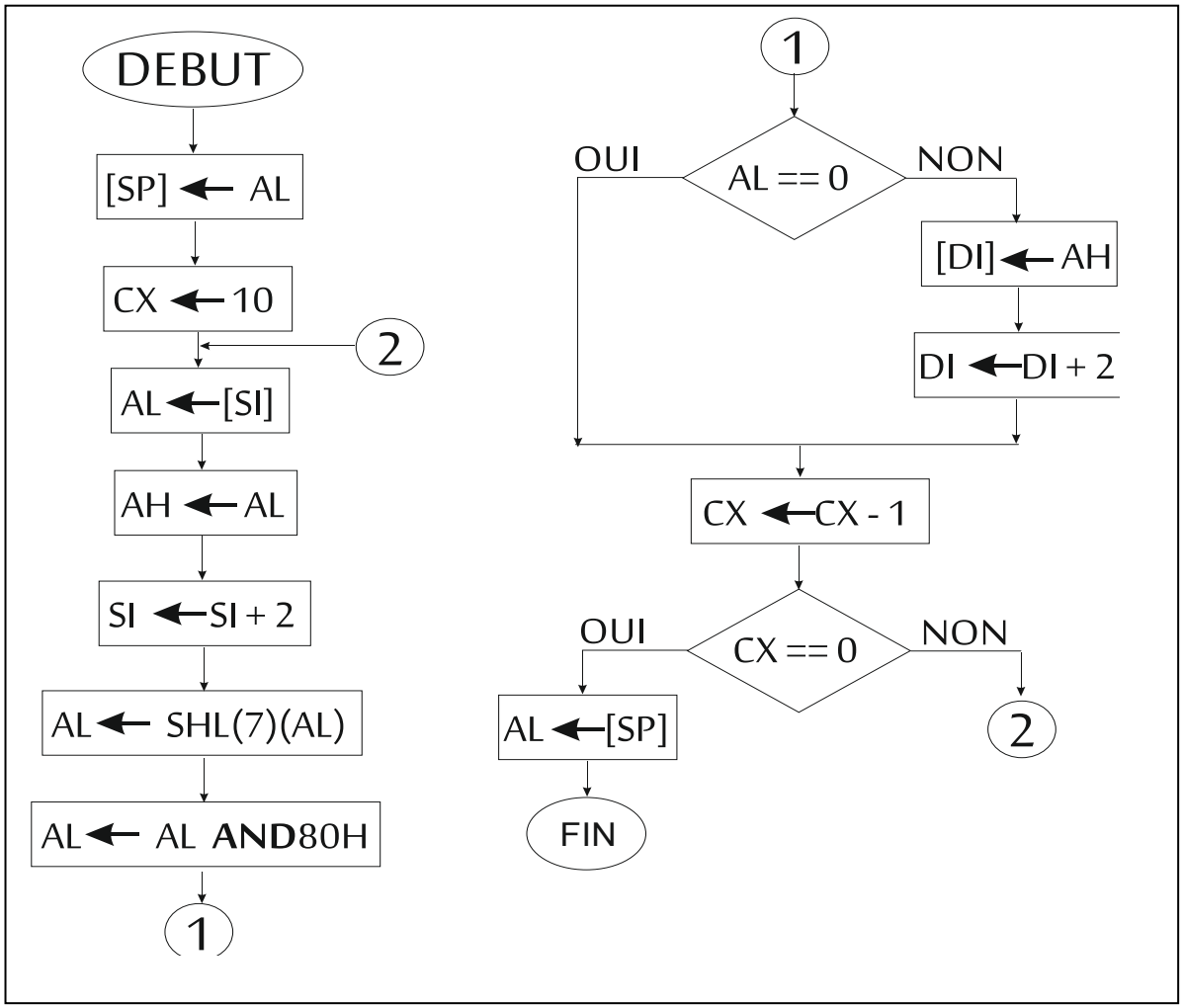
2)-

```
                MOV CX, 100
```

```
                AND BX, 0
                AND DX, 0          ; initialisation
REPRISE :       MOV AL , [SI]
                MOV AH, AL         ;  Sauvegarde
                AND AL, 10H
                JZ 5eme_Bit_Nul
                PUSH AH
                INC BX             ; MAJ Compteur BX = Nbre de sauvegarde ..
                                   ; .. .  en pile
                JMP SUITE
5eme_Bit_Nul:   MOV [DI], AH
                INC BX             ; MAJ Compteur DX = Nbre de sauvegarde ..
                                   ; .. .  en zone DI
                INC DI
SUITE:          INC SI
                LOOP REPRISE
```

Exercice 3 (5 Pts) *(Temps_référence : 20 mn)*

1)- Traduire l'organigramme suivant en un programme assembleur '80286' :



On rappelle que « AL ← SHL(7)(AL) » signifie : « AL reçoit AL après décalage à gauche de 7 bits » ;
2)- Indiquer la fonction principale de ce bout de programme.
3)- Cette fonction commence par « [SP] ← AL » et est achevée par « AL ← [SP] » : quel est l'intérêt de cette sauvegarde en pile ?

1)-

```
                PUSH AL
                MOV CX, 10
ETIQ_3 :        MOV AL , [SI]
                MOV AH, AL
                ADD SI, 2
                SHL AL, 7
                AND AL, 80H
                JZ ETIQ_2
                MOV [DI], AH
                ADD DI , 2
ETIQ_2 :        LOOP ETIQ_3
                POP AL
```

2)- Tri de parité de 10 données de 1 octet chacune : les données ayant le bit (b0) non nul (avant décalage) , donc impaires, sont stockées en zone (DI).

3)- L'empilement / Dépilement de AL permet de récupérer la valeur initiale en fi de programme.

Exercice 4 *(4 Pts)* *(Temps_référence : 20 mn)*

On considère l'organigramme de **tri** suivant, où « Indice » est une variable définie sur 2 octets :

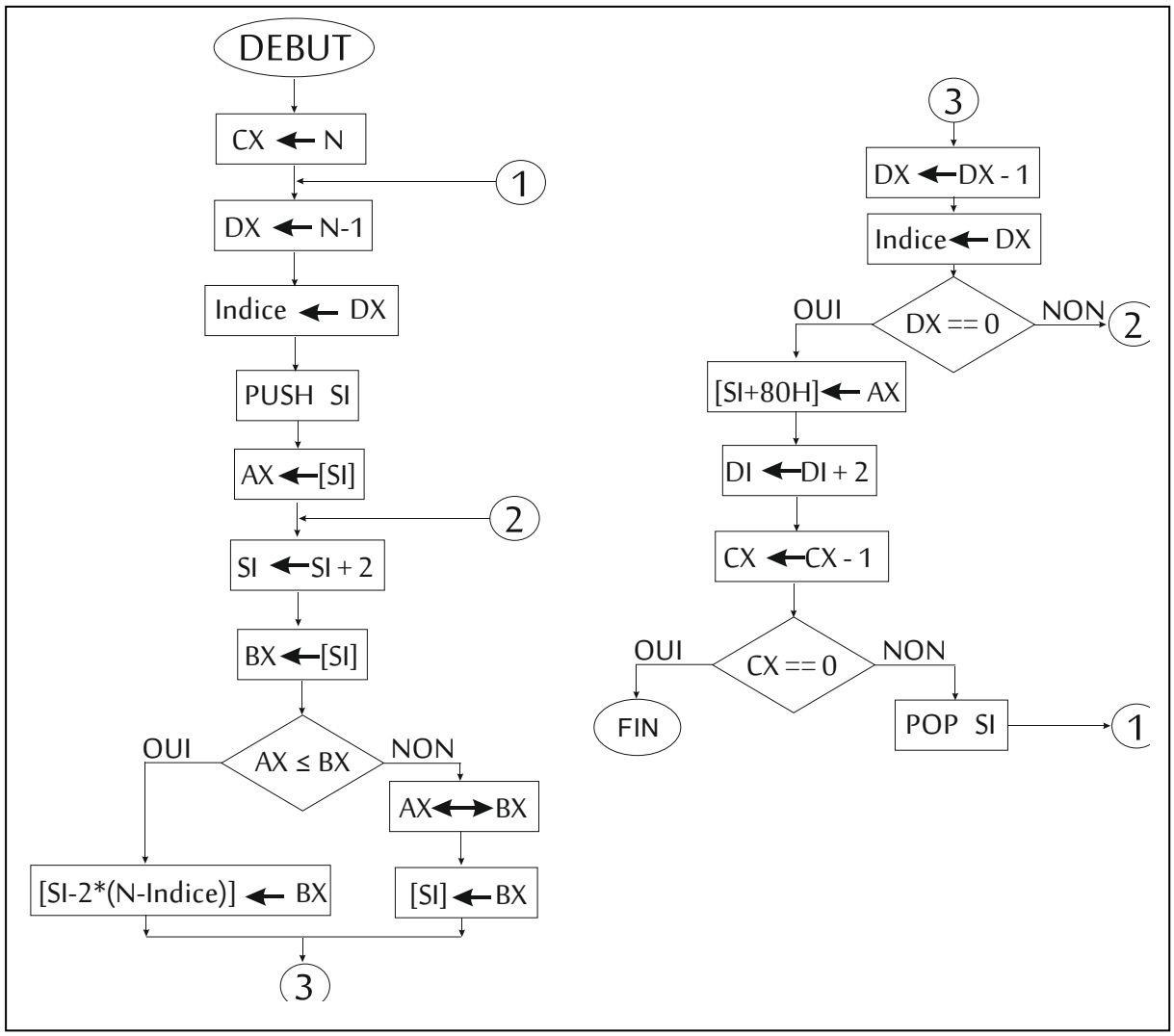
1)- Quelle condition est posée sur la valeur limite de 'N' ?
2)- **Question d'excellence :** Si on remplace respectivement l'instruction « [SI+80H] ←AX » par « **SAUVEGARDE de (AX) en PILE** », quelles modifications supplémentaires doivent être apportées au programme ? (la réponse peut être fournie indifféremment sous forme d'organigramme ou de programme assembleur).

1)- 'N' est tel que:

$2*N < 80H$ donc $N < 40H$

2)- 'N' est tel que:
On va « cadrer » le ' PUSH AX ' introduit comme suit :

```
SUB SP, 80H
PUSH AX
ADD SP, 80H
```


DEBUT
CX ← N
1
DX ← N-1
Indice ← DX
PUSH SI
AX ←[SI]
2
SI ←SI + 2
BX ←[SI]
OUI
AX ≤ BX
NON
AX↔BX
[SI-2*(N-Indice)] ← BX
[SI] ←BX
3
3
DX ←DX - 1
Indice ← DX
OUI
DX == 0
NON
2
[SI+80H] ← AX
DI ←DI + 2
CX ←CX - 1
OUI
CX == 0
NON
FIN
POP SI
1

Bon Courage