Exercice 7:

1. Donner les valeurs du registre AX après exécution des instructions suivantes :

```
MOV AX , 20 ;
SHL AX , 8 ;
AND AX , 0200H ;
```

2. Trouver la valeur de (V) pour que la valeur FINALE du registre AX soit = 800H :

```
MOV AX , 0028H;
SHL AX , V ;
AND AX , 0000 1000 0000 0000b;
```

Correction

Exercice 7:

1. Donner les valeurs du registre AX après exécution des instructions suivantes :

```
MOV AX , 20 ;
SHL AX , 8 ;
AND AX , 0200H ;
```

2. Trouver la valeur de (V) pour que la valeur FINALE du registre AX soit = 800H :

```
MOV AX , 0028H;
SHL AX , V ;
AND AX , 0000 1000 0000 0000b;
```

1. Donner les valeurs du registre AX après exécution des instructions suivantes :

```
MOV AX , 20 ; AX=20=0014h
SHL AX , 8 ; AX=1400h
AND AX , 0200H ; Ax = 0
```

2. Trouver la valeur de (V) pour que la valeur FINALE du registre AX soit = 800H :

```
MOV AX , 0028H;AX=0000000000101000b
SHL AX , V ; V = 6 ou 8 pour que. . .
AND AX , 0000 1000 0000 0000b; AX=800H
```

Exercice 8 :

On donne la valeur initiale: CX=1111 1111 1111 1111b ; Donner les valeurs finales des registres BX et CX après l'exécution du programme suivant :

```
AND CX , FFFFH ;
MOV BX , 2 ;
SHL CX , 16 ;
INC CX ;
OR CX , BX ;
```

Correction

Exercice 8 :

On donne la valeur initiale: CX=1111 1111 1111 1111b ; Donner les valeurs finales des registres BX et CX après l'exécution du programme suivant :

```
AND CX , FFFFH ;
MOV BX , 2 ;
SHL CX , 16 ;
INC CX ;
OR CX , BX ;
```

On donne la valeur initiale: CX=1111 1111 1111 1111b ; Donner les valeurs finales des registres BX et CX après l'exécution du programme suivant :

AND CX , FFFFH ; CX = FFFFH
MOV BX , 2 ; BX = 2
SHL CX , 16 ; CX = 0000H
INC CX ; CX = 0001H
OR CX , BX ; CX = 0003H=11b & . . .. . .BX=0002H=10b

Exercice 9

Ecrire un programme en assembleur 8086 permettant de faire la somme des différents élements de tableau ci dessous

```
jmp start
table db 1 dup (30h,20h,33h,0xFE,5Eh,0xA5,

0xB6,0xC2,1Dh,81h)

Aresult db ?
start:
xor si,si
mov cx.10
ET1: mov bl,table[si];
   add al,bl
  inc si
   loop ET1
    mov Aresult,al ;
   ret
```

Adresse RAM	Valeur contenue
100 H	30h
102 H	20h
104 H	33h
106 H	FE h
108 H	5E h
10A H	A5 h
10C H	B6 h
10E H	C2 h
110 H	1D h
112 H	81h

2) On donne dans le tableau ci-contre les valeurs des données (0, 1, 2, ...9) contenues aux adresses respectives (100H à 112H) de la RAM. Utiliser ces valeurs pour déduire les valeurs finales de (AX) et (BX) pour le programme suivant :

```
MOV CX , 10H;
MOV BX , 0100H;
MOV AX , 0 ;
ALPHA: ADD AX , [BX] ;
ADD BX , 2 ;
DEC CX ;
LOOP ALPHA ;
```

Adresse RAM	Valeur contenue
100 H	0
102 H	1
104 H	2
106 H	3
108 H	4
10A H	5
10C H	6
10E H	7
110 H	8
112 H	9

Correction ex9

On donne dans le tableau ci-contre les valeurs des données (0, 1, 2, …9) contenues aux adresses respectives (100H à 112H) de la RAM. Utiliser ces valeurs pour déduire les valeurs finales de (AX) et (BX) pour le programme suivant :

```
MOV CX , 10H; CX =16 (decimal)
MOV BX , 0100H; BX = 0100H
MOV AX , 0 ; AX=0
ALPHA: ADD AX , [BX] ; AX=0+0/0+1/1+2/3+3/6+4/10+5/15+6/21+7
ADD BX , 2 ; BX=102H/104H/106H/108H/10AH/10CH/10EH/
110H
DEC CX ; CX=15/13/11/9/7/5/3/1
LOOP ALPHA ; CX=14/12/10/8/6/4/2/0
```

Donc AX final = 21+7=28 ; BX final = 110H

Adresse RAM	*Valeur contenue*
100 H	0
102 H	1
104 H	2
106 H	3
108 H	4
10A H	5
10C H	6
10E H	7
110 H	8
112 H	9

Exercice 10

On donne les valeurs initiales: AX=01101010b. Donner les instructions logiques et les masques m permettant de :

a. effectuer le complément à 1 de AX.
b. effectuer le complément à 2 de AX
c. forcer à 1 le bit 6.
d. forcer à 0 les 4 bits de poids faible
e. forcer à 0 les bits 1, 2, 4, 6, forcer à 1 les bits 0, 5, 7 et laisser inchangé le bit 3

Correction ex 10

a. effectuer le complément à 1 de AX.

```
MOV AL, 01101010B
NOT AL
```

b. effectuer le complément à 2 de AX

```
MOV AL, 01101010B
NEG AL
```

c. forcer à 1 le bit 6.

```
MOV AL, 01101010B
MOV BL, 01000000B
OR AL, BL
```

d. forcer à 0 les 4 bits de poids faible

```
MOV AL, 01101010B
MOV BL. 11110000B
AND AL, BL
```

e. forcer à 0 les bits 1, 2, 4, 6, forcer à 1 les bits 0, 5, 7 et laisser inchangé le bit 3

```
MOV AL, 01101010B
MOV BL, 10101001B
AND AL, BL
MOV BL, 10100001B
OR AL, BL
```