UNIVERSITE AbouBakr Belkaïd **CORRECTION** *Date : 12-01-2015*

FACULTE des Sciences *E.M.D : 'Archi des Ordinateurs'* *2ème année Licence Info*

Département d'informatique *(UF 2111)* *Durée : 1 h 30 mn*

Exercice 1 (3 * 2 Pts) (Temps recommandé : 15 mn)

Répondre aux questions **Q1** à **Q3** suivantes :

Q1/ Donner les valeurs du registre AX après exécution des instructions suivantes :

```
MOV   AX , 20 ;   AX = 20 (14H)
SHL   AX , 9 ;    AX = 2800H
AND   AX , 2000H ; AX = 2000H
```

Q2/ Pour le programme suivant, trouver la valeur de **(V)** pour que la valeur FINALE du registre AX soit = 8002H :

```
MOV   AX , 0280H ;
ROL   AX , V ;     V = 8
```

Q3/ On donne la valeur initiale:
CX = **1111 1111 1111 1111 b** ;
Donner les valeurs finales des registres **BX** et **CX** **pendant et après** l'exécution du programme suivant :

```
AND   CX , FFFFH ;   CX = FFFF H
MOV   BX , 2H ;      BX = 0002 H  (= 'BX' Finale)
SHL   CX , 1H ;      CX = FFFE H
INC   CX ;           CX = FFFF H
SHL   CX , 2H ;      CX = FFFC H
OR    CX , BX ;      CX = FFFE H H  (= 'CX' Finale)
```

Exercice 2 (14 Pts) (Temps recommandé : 60 mn)

On considère le programme ASM '80x86' suivant :

```
            MOV AX, 10 ;                    (1)
            SHL AX, 12 ;                    (2)
BCL :       MOV BX, AX ;                    (3)
            AND AX, M1 ;                    (4)
            SUB AX, M2 ;                    (5)
            JZ CAS_1 ;                      (6)
            JMP CAS_2 ;                     (7)
CAS_1:      Instruction à identifier ;      (8)
            MOV DX OFFSET message ;         (9)
            MOV AH, 09H ;                   (10)
            INT 21H ;                       (11)
            MOV AX, 4C00H ;                 (12)
            INT 21H ;                       (13)


CAS_2:      Instruction à identifier    ;   (14)
            MOV DX OFFSET message ;         (15)
            MOV AH, 09H ;                   (16)
```

```
                INT 21H ;                       (17)
                MOV AX, 4C00H ;                 (18)
                INT 21H ;                       (19)
```

1- On donne les deux instructions suivantes :

I1 : { message DB 'AX contient une valeur NEGATIVE$' ; }

I2 : { message DB 'AX contient une valeur POSITIVE $' ; }

Placer chacune de ces deux instructions à l'endroit qui leur convient dans le programme (lignes (8) et (14)) : Justifier votre réponse *(1 Pt)*

REPONSE (1) :

« valeur NEGATIVE » du message est atteint après un « JZ » précédé d'un « AND » puis d'un « SUB » .. vu cette structure, l'instruction (I1) va en ligne (8), et (I2) en ligne (14), d'où :

```
                MOV AX, 10 ;                    (1)
                (. . .)
                JMP CAS_2 ;                     (7)
CAS_1:          message DB 'AX contient une valeur NEGATIVE$' ;   (8)
                MOV DX OFFSET message ;         (9)
                (. . .)

CAS_2:          message DB 'AX contient une valeur POSITIVE $' ;  (14)
                (. . .)
                INT 21H ;                       (19)
```

2- Quelle(s) valeur(s) peu(ven)t alors remplacer **(M1)** et **(M2)** ? *(1 Pt)*

REPONSE (2) :

M1 = M2 = 8000 h (on cible le bit (b15))

3- Quel est l'intérêt de l'instruction (3) : « MOV BX, AX ; » ? Justifier votre réponse *(2 Pts)*

REPONSE (3) :

Sauvegarde du contenu de **(AX)** avant son affectation à venir.

4- Les lignes (15 à 19) sont identiques (donc répétition) aux lignes (9 à 13) ; Proposer une nouvelle version de ce programme en évitant cette répétition. *(3 Pts)*

REPONSE (4) :

```
                MOV AX, 10 ;                    (1)
                SHL AX, 12 ;                    (2)
BCL :           MOV BX, AX ;                    (3)
                AND AX, M1 ;                    (4)
                SUB AX, M2 ;                    (5)
```

```
                JZ CAS_1 ;                          (6)
                JMP CAS_2 ;                         (7)   (à éliminer)

CAS_2:    message DB 'AX contient une valeur POSITIVE $' ;   (14 → 7)

                JMP SORTIE_UNIQ ;                           (8)

 CAS_1:    message DB 'AX contient une valeur NEGATIVE$' ;   (8)→ (9)

SORTIE_UNIQ :   MOV DX OFFSET message ;             (9)→ (10)
                MOV AH, 09H ;                       (10)→ (11)
                INT 21H ;                           (11)→ (12)
                MOV AX, 4C00H ;                     (12)→ (13)
                INT 21H ;                           (13)→ (14)

      CAS_2:    message DB 'AX contient une valeur POSITIVE $' ;   (14)
                MOV DX OFFSET message ;             (15)
                MOV AH, 09H ;                       (16)
                INT 21H ;                           (17)
                MOV AX, 4C00H ;                     (18)
                INT 21H ;                           (19)
```

5- On remplace cette fois-ci les valeurs **M1** & **M2** des lignes (4 & 5) par la même valeur égale à « 1 » ; modifier les lignes (2), (4), (5), (8) et (14) en conséquence. *(3 Pts)*

REPONSE (5) :

```
                SHR AX, 1 ;       (ou '3')          (2)
BCL :           MOV BX, AX ;                        (3)
                AND AX, 1 ;                         (4)
                SUB AX, 1 ;       (Respt. « enlever ligne (5) »)  (5)

                (. . .)

CAS_1:          message DB 'AX contient une valeur PAIRE$' ;    (8)
                        ;         Respt. « IMPAIRE »            (8)

                (. . .)

CAS_2:          message DB 'AX contient une valeur IMPAIRE$' ;  (14)
                        ;Respt. « IMPAIRE »                     (14)
```

6- **Question d'excellence :** On souhaite ré-exécuter le programme ci-haut de manière cyclique, jusqu'à ce que l'utilisateur tape la touche « ENTER » (code ASCII : : **1C0Dh**); modifier le programme en conséquence. *(4 Pts)*

REPONSE (6) :

UNIVERSITE AbouBakr Belkaïd **CORRECTION** *Date : 12-01-2015*

FACULTE des Sciences *E.M.D : 'Archi des Ordinateurs'* *2ème année Licence Info*

Département d'informatique *(UE 2111)* *Durée : 1 h 30 mn*

REMARQUE : ***n'importe quelle version de ce programme peut être utilisée (PAIRE/IMPAIRE, POSITIVE/NEGATIVE, optimisé/non optimisé) :***

Le cas le plus naturel est de supposer que AX reçoit une valeur acquise à partir du clavier : cette valeur doit être alors identifiée « Négative / Positive », ou « Paire / Impaire » à condition qu'elle soit différente de la touche « ENTER » (sinon (en 2nde hypothèse), on peut faire l'acquisition d'une touche / clavier, et si elle est différente de « ENTER », on continue le traitement d'identification :

```
                message DB 'Taper ENTER pour arrêter $  ;
                MOV DX OFFSET message ;
                MOV AH, 09H ;
                INT 21H ;
                MOV AX, 10 ;        ne sert plus à rien !!  (1)
RELECTURE :     MOV AH, 00h;        caractère acquis par "clavier" ..
                INT 16h;            . . . & comparé avec . . .
                CMP  AX, 1C0Dh;     . . . "ENTER"
                JE EXIT;
                MOV BX, AX ;        Ici, SAUVEGARDE INDISPENSABLE !!
                SHL BX, 12 ;        on travaille sur « BX » (2)
                MOV BX, AX ;        permut (2) avec (3)     (3)
                AND BX, 8000h ;          On maintient le MASQUE        (4)
                SUB BX, 8000h ;     pr tester le bit « ciblé »     (5)
                JZ CAS_1 ;                              (6)
                JMP CAS_2 ;                             (7)
CAS_1:      message DB 'AX contient une valeur NEGATIVE$;       (8)
                MOV DX OFFSET message ;                 (9)
                JMP SORTIE_UNIQ;
                MOV AH, 09H ;       voir (Q4) pr éviter la . .  (10)
                INT 21H ;            . . . la REPET     (11)
                MOV AX, 4C00H ;                         (12)
                INT 21H ;                               (13)
CAS_2:      message DB 'AX contient une valeur POSITIVE $ ;     (14)
SORTIE_UNIQ :   MOV DX OFFSET message ;                 (15)
                MOV AH, 09H ;                           (16)
                INT 21H ;                               (17)
                MOV AX, 4C00H ;          ne sert plus à rien car ..
                INT 21H ;           boucle jusqu'à « ENTER » !!  (19)
                JMP RELECTURE;
EXIT:           MOV AX, 4C00h;      .. sinon .. SORTIE
                INT 21h;
```