Solution de la série 2

**Exercice 1 :**

1. AL=+127=7Fh

AL=-128=80h, SF=1, ZF=0, OF=1.

1. AL=80h=-128h

AL=7Eh=+126, SF=0, ZF=0, OF=1.

1. AL=03h=+3

AL=-1=FFh, SF=1, ZF=0, OF=0, SF≠OF (DESTINATION<SOURCE).

1. AL=+3

AL=+2=02h, SF=0, ZF=0, OF=0, SF=OF (DESTINATION>SOURCE).

1. AL=05h=+5

AL=+0=00h, SF=0, ZF=1, OF=0, SF=OF (DESTINATION=SOURCE).

**Exercice 2 :**

MOV AX,0

MOV BX,1

MOV DX,10

BOUCLE: ADD AX, BX

INC BX

CMP BX, 10

LOOPNE BOUCLE

**Exercice 3 :**

MOV AX,#nombre

MUL AX

**Exercice 4 :**

Pour vérifier si un nombre est pair, nous devons le diviser sur 2 et vérifier le reste de la division. Si le reste est 0 le nombre est pair sinon, il est impair.

Le code assembleur est le suivant:

MOV AX, #nombre ; le contenu du rY + la valeur 5.

MOV BX, 2

DIV BX ; diviser AX sur 2. Le reste est stocké dans AH.

CMP AH, 0

JE SAUT ; effectuer un saut si le nombre est pair.

... reste du code.

**Exercice 5 :**

|  |  |
| --- | --- |
| org 100h  jmp start | set location counter to 100h |
| NOTES DB 08, 06, 19, 11, 18 | on donne 5 notes aléatoires pour vérifier l'exécution, mais vous pouvez laisser les indéfinis en mettant des ? |
| PLUS\_G DB?  start: |  |
| MOV CX, 5 | compteur de boucle il se décrémente automatiquement chaque itération |
| MOV BX, OFFSET NOTES | BX sera utilisé pour un adressage basé, il sert pour pointer les données NOTES |
| XOR AL, AL | Initialise AL à 0 |
| ENCORE: CMP AL, [BX] | compare la note prochaine à la note la plus élevée |
| JA PROCHAIN | Sauter si AL est encore la note la plus élevée |
| MOV AL, [BX] | sinon AL retient la plus élevée |
| PROCHAIN: INC BX | pointe vers la prochaine note |
| LOOP ENCORE | CX décrémente jusqu’à 0 pour sortir de la LOOP |
| MOV PLUS\_G, AL | sauvegarde de la note la plus élevée dans PLUS\_G |

**Exercice 06 :**

|  |  |
| --- | --- |
| org 100h | set location counter to 100h |
| jmp start  tab db 100,200,0,0,11, 75 dup( ?) | on va déclarer une variable tab constituer de 80 octet, les 5 premières valeurs ont été initialisées pour une éventuelle vérification |
| START:  mov cx, 5  xor al,al | pour initialiser al à zéro |
| lea bx,tab | cette instruction permet de charger du premier élement de tab  elle est équivalente à l’instruction mov bx, offset tab |
| encore:  mov dl, 0ffh test [bx], dl jnz nzero inc al  nzero:  inc bx  loop encore  mov [bx], al ret | on va stocker le résultat (nombres de zéro) dans la case mémoire qui correspond à l'adresse 300h mov bx, 300h |

# Exercice 7 :

Exploitation des registres :

Le compteur C sera remplacé par le registre CX, et comme tout le tableau de N octets sera examiné, donc CX va prendre N valeurs, la toute dernière est toujours égale à 1, donc la valeur initiale sera N :

CX = N, N-1, N-2, ... 2, 1.

Lorsque CX est initialisé à N, La valeur actuelle du tableau mémoire, qui est définie par DS:[SI], est initialisée par DS:[100h] ; à chaque décrémentation de CX correspond une incrémentation de SI.

Début

NON

T(C) = 0 ?

OUI

NZ <-- NZ + 1

C <-- C + 1

NON

OUI

C < N ?

Fin

NZ <-- 0

C <-- 1

BX <-- 0h CX <-- 100h SI <-- 200h

CMP DS:[SI],0

DS:[400h] <-- BX

Après la décrémentation de CX, on n’a pas besoin d’effectuer une comparaison entre CX et 0, parce que la décrémentation est suffisante pour affecter le flag Z dont on a besoin pour effectuer l’instruction de saut qui vient juste après la décrémentation de CX.

Début

NON

Z = 1 ?

OUI

BX <-- BX + 1

SI <-- SI + 1 CX <-- CX - 1

OUI

NON

Z = 1 ?

Fin

Donc les instructions de saut sont généralement utilisées après une comparaison ou après une décrémentation du compteur.

Le programme en langage assembleur 8086 : (partie la plus significative)

MOV CX, 100h MOV BX, 0000h MOV SI, 200h

Etq2 : CMP [SI], 0

JNZ Etq1 INC BX

Etq1 : INC SI

DEC CX JNZ Etq2

MOV [400], BX BRK

Contrairement à l’instruction HLT qui fait arrêter le 8086 de toute activité jusqu’à l’arrivée d’une interruption matérielle, l’instruction BRK (break) fait arrêter le programme en cours d’exécution et revenir à l’exécution du système d’exploitation.

**Exercice 8:**

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] JLE Etq1 MOV AL, [SI]

Etq1 : DEC CX

JNZ Etq2

MOV [400], AL BRK

Début

OUI

NON

AL > [SI] ?

AL <-- [SI]

CX <-- CX - 1

OUI

NON

Z = 1 ?

Fin

CX <-- FFh SI <-- 200h

AL <-- [SI]

SI <-- SI + 1

DS:[400h] <-- AL