Série de TD N°6

Exercice 1 : Expliquer les instructions suivantes et indiquez le mode d'adressage :

MOV EAL, [E58]
ADD EAL, F632
MOV [EBX], [ebx + ecx * 4 + 8]
SUB RAL, [RBX+ESI]
MOVZX EAX, Word [edi]
MOV EAX, RBX
MOV [Rdi + Rsi], Edx
MOV EAX, [EBX]
MOV RAX, [RBX+4]

Exercice 2 : 1- Ecrire un programme en assembleur 8086 qui permet de :

- Trouvez le maximum entre deux nombre entiers
- Ajoutez le code qui fait la recherche du maximum pour 04 nombres entiers
- Effectuer un JUMP dans votre code vers une instruction qui stocke le résultat dans un nouveau registre
- Réalisez le programme avec l'utilisation des registres de l'architecture 32 et 64 bits

Correction Série 6

Exercice 1 : Expliquer les instructions suivantes et indiquez le mode d'adressage :

MOV EAL, [E5**8**] **Mode direct : copier le contenu de l'adresse mémoire E58 dans le registre EAL, en utilisant l'architecture 32 bits**
ADD EAL, F632 **Mode immédiat : Copier la valeur en hexadécimal F632 dans le registre EAL, en utilisant l'architecture 32 bits**
MOV [EBX], [**ebx** + **ecx * 4** + **8]** **Incorrect : Impossible de copier une adresse vers une autre adresse**
SUB RAL, [RBX+ESI] **Mode indirect basé indexé : soustraction de contenu de l'adresse mémoire des registres RBX+ ESI de la valeur contenue dans le registre RAL en utilisant l'architecture 64 bits**

MOVZX EAX, Word [edi] **Mode direct : copier un mot mémoire sur 16 bits et** modifier **eax** en mettant dans **al** le mot mémoire pointé par **edi** et en mettant à 0 les 16 autres bits. **, en utilisant l'architecture 32 bits**

MOV EAX, RBX **Incorrect : Impossible de copier le contenu de registre 64 bits dans un registre 32 bits**
MOV [**Rdi** + **Rsi**], **Edx** **Mode indirect indexé : stocker dans l'adresse Rdi + Rsi la valeur contenue dans le registre edx en utilisant l'architecture 64 bits**
MOV EAX, [EBX] **Mode indirect basé : stocker dans le registre Eax le contenu de registre EBX en utilisant l'architecture 32 bits**
MOV RAX, [RBX+4] **Mode indirect basé : stocker dans le registre RAX le contenu de l'adresse mémoire RBX +4 en utilisant l'architecture 64 bits**

Exercice 2 : 1- Ecrire un programme en assembleur 8086 qui permet de :

- Trouvez le maximum entre deux nombres entiers
- Ajoutez le code qui fait la recherche du maximum pour 4 nombres entiers
- Effectuer un JUMP dans votre code vers une instruction qui stocke le résultat dans un nouveau registre
- Réalisez le programme avec l'utilisation des registres de l'architecture 32 et 64 bits

1. Le maximum entre deux nombres entiers (en utilisant l'architecture 32 bits) :

```
MOV EAX, #nombre1
MOV EBX, #nombre2
Cmp EAX, EBX
JG  stocker
Mov  EDX, EBX

Stocker:
Mov  EDX, EAX

JMP Fin

Fin int 21H
```

2. Le maximum entre quatre nombres entiers (en utilisant l'architecture 64 bits) :

Solution 1

```
MOV RAX, #nombre1
MOV RBX, #nombre2
MOV RDX, #nombre3
MOV RCX, #nombre4

CMP RAX, RBX
JG   Max1
Mov  R1, RBX

Max 1:
Mov  R1, RAX

CMP RDX, RCX
 JG   Max 2

  Mov  R2, RCX


Max 2 :

Mov  R2, RDX

     Cmp R1, R2

     JG Max

Mov  R3, R2

Max :

Mov R3, R1

JMP FIN
```

Solution 2

```
.MODEL SMALL
.STACK 100H
.DATA
   nombres DW 12, 45, 7, 23 ; Les 4 nombres à comparer (peuvent être
modifiés)
   count DB 3 ; Nombre de comparaisons à faire (4 nombres = 3
comparaisons)
   max_msg DB 'Le maximum est: $'

.CODE
Fonction Max  PROC near
   MOV AX, @DATA
   MOV DS, AX

   ; Initialisation avec le premier nombre
   MOV AX, nombres[0]
      ; Initialisation pour la boucle

   MOV SI, 2 ; Décalage pour accéder aux nombres suivants
   MOV CX, count ; Compteur de boucle

comparaison
   ; Charger le nombre suivant dans BX
   MOV BX, nombres[SI]

   ; Comparaison
   CMP AX, BX
   JGE pas_plus_grand
   MOV AX, BX ; AX = nouveau maximum

pas_plus_grand:
   ; Préparation pour la prochaine itération
   ADD SI, 2 ; Passer au nombre suivant (chaque nombre = 2 octets)
   DEC CX
    JNZ
Loop comparaison ; Continuer tant qu'il reste des nombres

   ; Affichage du résultat
   MOV AH, 02H
   LEA DX, max_msg
   INT 21H

   ; Conversion et affichage du nombre (supposé < 100)
   MOV BX, AX ; Sauvegarde du maximum

   ; Affichage des dizaines
   MOV AL, BL
   MOV AH, 0
    MOV DL, 10
   DIV DL ; AL = dizaines, AH = unités

   MOV DL, AL
    ADD DL, '0'
   MOV AH, 02H
   INT 21H

   ; Affichage des unités
   MOV DL, AH
   ADD DL, '0'
   MOV AH, 02H
   INT 21H

   ; Fin du programme
   MOV AH, 4CH
   INT 21H
ENDP
ENDS
```