TP 2 : Programmation assembleur avec EMU8086

BUT DU TP : Se familiariser avec l'EMU8086, maitriser les instructions de transfert des données et les calculs d'adresses.

1. Définition de l'assembleur

L'assembleur est un langage de programmation transformant un fichier texte contenant des instructions, en un programme que le processeur peut comprendre (programme en langage machine).

Ce langage machine à la particularité d'être difficile à programmer car il n'est composé que des nombres en hexadécimal.

2. Présentation d'EMU 8086

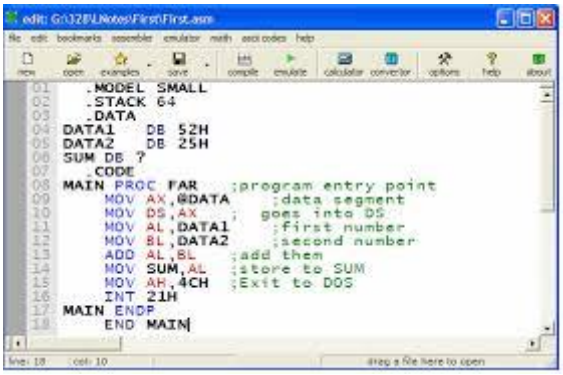
EMU8086 est un émulateur du microprocesseur **8086** (compatible Intel et AMD). Dans un premier temps, nous allons utiliser **EMU8086** pour consulter/modifier les registres, pour afficher/modifier le contenu de la mémoire et enfin pour éditer /exécuter des programmes élémentaires.

Les instructions peuvent être exécutées en arrière et en avant. L'interface visuelle est très facile à utiliser. Vous pouvez regarder les registres, les drapeaux et la mémoire pendant l'exécution de votre programme.

Au lancement de l'émulateur, une fenêtre contenant la boite de dialogue suivante s'affiche (figure 1), cette dernière permet de choisir entre plusieurs options et de créer un nouveau fichier.

Figure 1. Boite de dialogue bienvenue.

En cliquant sur le bouton New, une boite de dialogue s'ouvre, cette fenêtre permet de choisir un type de fichier à créer. Dans ce TP en utilisant des fichiers binaire (d'extension.bin) on sélectionne le format **BIN** et on clique sur **OK**. La fenêtre qui s'affiche présente l'éditeur de programme de l'Emu 8086.



```
.MODEL SMALL
.STACK 64
.DATA
DATA1   DB 52H
DATA2   DB 25H
SUM DB ?
.CODE
MAIN PROC FAR    ;program entry point
    MOV AX,@DATA    ;data segment
    MOV DS,AX       ; goes into DS
    MOV AL,DATA1    ;first number
    MOV BL,DATA2    ;second number
    ADD AL,BL    ;add them
    MOV SUM,AL   ;store to SUM
    MOV AH,4CH   ;Exit to DOS
    INT 21H
MAIN ENDP
    END MAIN
```

Figure 2. Editeur de programme de l'EMU8086.

Une fois la saisie du programme est terminée on peut soit le compiler ou l'émule, on cliquant sur le bouton **Emulate,** et après les corrections des erreurs syntaxiques les deux fenêtres suivantes s'ouvrent :

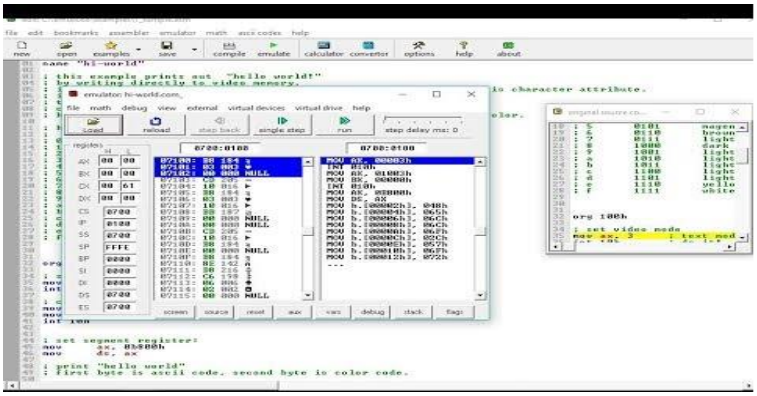


Figure 3. Les 2 fenêtres de l'EMU8086.

La petite fenêtre contient le code source, et la grande représente l'émulateur. Ce dernier permet d'exécuter le programme tout entier instruction par instruction. Il permet aussi de visualiser les contenus de tous les registres de microprocesseur ainsi que les différents segments de la mémoire et différents composants de système.

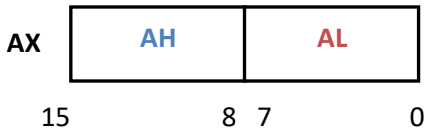
Les instructions peuvent être exécutées en arrière et en avant. L'interface visuelle est très facile à utiliser. Vous pouvez regarder les registres, les drapeaux et la mémoire pendant l'exécution de votre programme.

a. Registres sur 8086 :

- 8 registres généraux (16 bits) : **AX**, **BX**, **CX**, **DX**, **SI**, **DI**, **DP** et **SP**
- 4 registres de segments (16 bits) : **CS**, **DS**, **ES**, **SS**
- 2 registres spéciaux (16 bits) : **IP** et **FLAGS**

b. Décomposition des registres :

- Les registres A, B, C et D se décomposent en deux sous registres de 8 bits chacun :
- H (partie haute) et L (partie basse)
- Par exemple, **AX** se compose en **AH** et **AL**.



c. Instruction de base :

MOV : transfert d'information

L'instruction la plus utilisée est l'instruction MOV, qui copie la valeur d'un opérande ***source*** dans un opérande ***destination***

- **Format :** **MOV** destination, source

  Cinq formes sont possibles

  MOV reg, reg — reg : registre
  MOV reg, mem — mem : adresse mémoire
  MOV mem, reg
  MOV reg, immed — immed : valeur immediate(binaire, décimale, hexadecimal)
  MOV mem, imme

Notez qu'il n'existe pas de transfert d'une adresse mémoire à une adresse mémoire.

Exemple :

MOV AX, 1h ➡ AX= 1h
MOV AX, BX ➡ AX= BX

MOV AL, [BX] ➡ AL = Le contenu de l'adresse mémoire pointée par DS : BX

3. **Structure d'un programme EMU8086 :** Il existe deux structures**,** sa dépend de la version d'EMU 8086 utilisé :

//Structure 1//

```
TITLE nom  ; donner un nom au programme
   PILE  SEGMENT STACK  ; déclaration d'un segment de pile
                        ; dont le nom  est pile
   ……
   ……
   PILE  ENDS           ; fin de la déclaration de la pile

   DONNEE  SEGMENT ; déclaration d'un segment de données qui va
                   ; contenir les variables
   ……
   DONNEE ENDS      ; fin de la déclaration de données

   LECODE  SEGMENT  ; déclaration du segment de code  qui va contenir le
   code
        Debut:      ; étiquette d'indication du point d'entrée du code
              MOV  AX,DATA
              MOV  DS, AX
   ……
   ……
              MOV    AH, 4Ch
              INT    21h
   LECODE ENDS      ; fin de la déclaration du code
        END  Debut  ; fin du point d'entrée du code
```

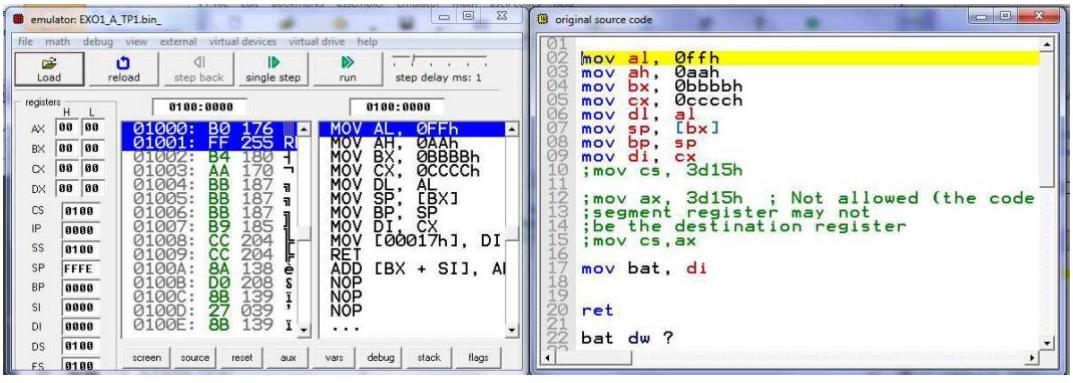
//Structure 2//

```
ORG 100h              ; Début de pgm

include emu 8086 inc  ; Appel de biblio 8086
.data                 ; Partie de déclaration
.code                 ; début du pgm ou le code

INT 21h               ; Appel de l'interruption 21h
END
RET
```

Exercice 1 :

1-1) Lancez emu8086, appuyez sur 'new' et 'cancel', puis 'save' or sauvegardez cette session sous: EXO1_A_TP1.asm dans le répertoire de travail.
Copiez les lignes suivantes dans de l'éditeur, corrigez les par l'exécution de ce fragment (signle step==> pas à pas):

```
mov al, ffh
mov ah, aah
mov bx, bbbbh
mov cx, ccccch
mov dl, al
mov sp, [al+bx]
mov bp, sp
mov di , cx
mov cs, 3d15h
mov bat, di
```



- Vérifiez/confirmez le contenu des différents registres du mp8086.
- Est ce que la variable '**bat**' contient la valeur '**cccch**'? Montrez-le.

1-2) - Identifiez les fenêtres suivantes: code source original, code source en exécution, émulateur en exécution.
- Identifiez les zones suivantes: zone registres, zone mémoire et zone code désassemblé.

Exercice 2 :

1. Ecrire un programme Assembleur 8086 qui affiche le texte suivant « *Voici mon premier programme Assembleur du microprocesseur 8086* »
2. Soit le programme suivant, remplir les vides

```
ORG100h
MOV AX, 243 ;…….
ADD AX, 243h ; .....
;  ……..
MOV AX, 0xA243; .....
MOV AX, 0A243h ; .....
MOV AX, 1010001001000011b; .....
MOV AL, 'a' ; .....ASCII du caractère 'a'
MOV AX, 'a' ; ..... ASCII du caractère 'a'
MOV AX,'ab' ; .....
```

Emulator du mp 8086 :dispositif qui émet le mp8086 => comprendre l'architecture 8086
Emulator <> simalator
Emulator = materiel (carte = kit) + logiciel
Simulator = Logiciel