TP1 – Initiation à l’assembleur

## Objectif du TP :

Le but de ce TP est de s’approprier les concepts liés à l’architecture d’un ordinateur parmi lesquels :

* L’organisation d’un PC
* Les composantes d’un CPU
* La notion d’instruction et leur emplacement en mémoire
* Les différentes phases de l’exécution d’une instruction

## Prise en main de l’émulateur

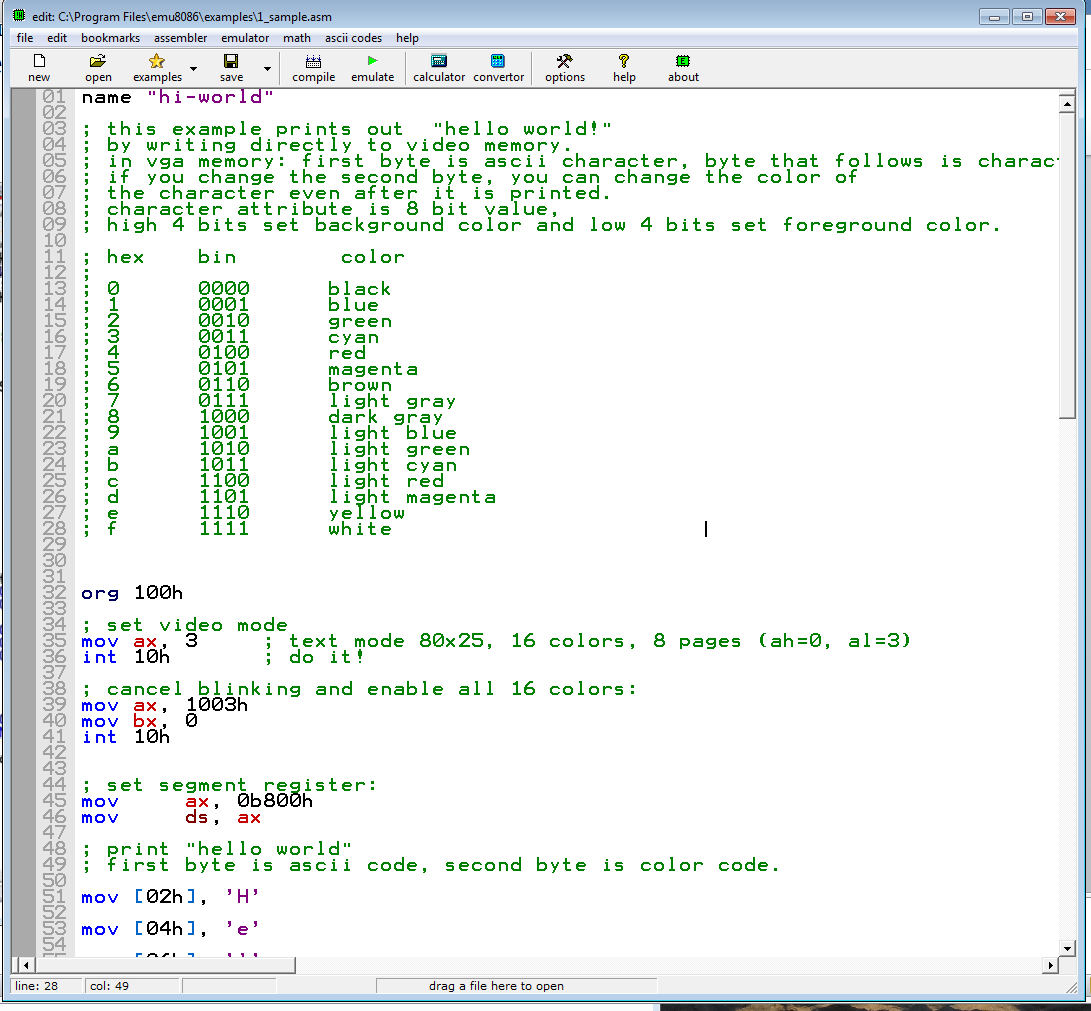
## Avant de commencer ce TP, créez dans votre répertoire de travail un dossier nommé

M211, et à l’intérieur de ce dossier un autre dossier nommé TP1. Vous travaillerez dans

ce répertoire.

Dans un premier temps, pour simplifier l’apprentissage de l’assembleur x86, nous allons

utiliser un logiciel qui émule un processeur 8086 : **Emu8086**. Ce logiciel se présente dans un premier temps comme un éditeur de texte classique, avec le support d’une coloration syntaxique du code assembleur (mode éditeur).



Vous trouvez ce logiciel dans

\\DC-info-04.iutnice.unice.fr\Logiciels\Langages\Asm\emu8086.

Créez un raccourci sur votre bureau et lancez le programme. Vous pouvez cliquer sur new puis cocher EXE template pour avoir un squelette d’un programme en assembleur. Les segments de pile, code et données sont déjà définis, il ne vous reste plus qu’à écrire aux endroits indiqués pour créer votre programme.

Afin de créer un exécutable dans votre répertoire de travail, cliquez sur « assembler» dans le menu puis sur set output directory. Décochez « use default [emu8086/MyBuild] directory » puis parcourez vos répertoires pour aller dans

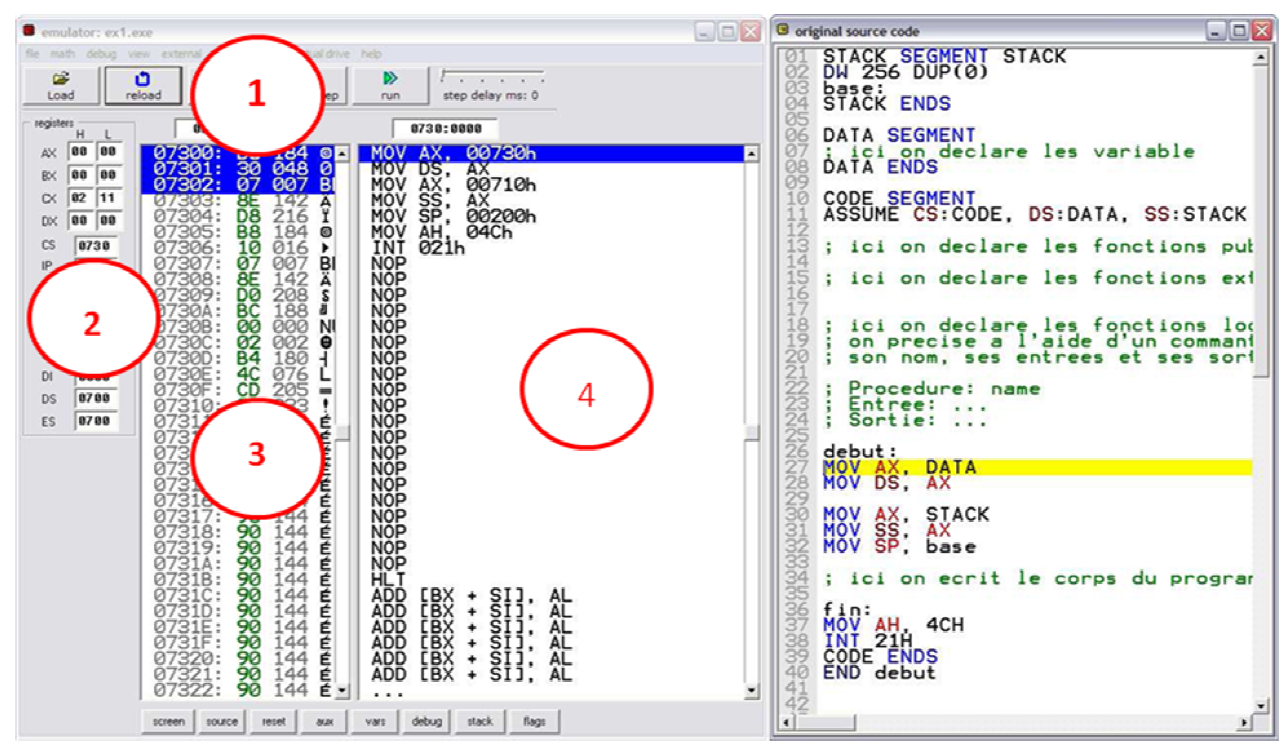
ASR1\TP1 puis cliquez sur ok.

Lorsque vous compilez un programme, cliquez sur Compile. Vous pouvez tester cela en ouvrant dans le fichier « ex1.asm » présent dans SupportCours\ASR1. Deux nouvelles fenêtres doivent s’ouvrir, il s’agit du mode exécution de l’émulateur. La fenêtre « *original* *source code* » contient le code tel que vous l’avez écrit. La ligne surlignée en jaune est la prochaine instruction qui va être exécutée.

Comme on le verra plus loin dans ce TP, il est possible d’interagir avec ce code en

cliquant dessus. La zone mémoire de la fenêtre « e*mulator :* » change pour correspondra

à la partie sélectionnée par le click.



La fenêtre « *emulator* : » contient 4 zones que nous allons détailler ci-dessous :

1. Les boutons vont nous permettre d’exécuter notre programme soit *Pas à Pas*, avec la possibilité de revenir en arrière, soit automatiquement (en pouvant régler un délai d’attente entre chaque instruction)

2. La deuxième zone contient une représentation des registres, et permet de visualiser leur valeur hexadécimale tout au long de l’exécution de votre programme. On notera que pour les registres AX, BX, CX et DX, on a une décomposition en 2 registres de 8 bits chacun

3. La zone 3 représente la mémoire octet par octet. Sur une ligne, on retrouve l’adresse mémoire sur 20 bits, la valeur de l’octet en hexadécimale, en décimale et sa représentation en ASCII.

4. La 4ème zone contient le code de votre programme une fois les traductions d’adresses terminées. Il n’y a plus aucune étiquette ni nom de variable. Il s’agit du code réellement exécuté.

## Prise en main de l’émulateur

Vous allez charger un programme exemple dans l’émulateur 80x86 Calculate\_Sum et vous allez l’exécuter en mode pas à pas .

* Tout d’abord charger ce programme dans l’émulateur
* Compilez le (compile)
* Exécutez-le (run)
* Recharger ce programme (reload) et observez :

Vous répondrez en particulier à ces questions

* Quelle est l’adresse de la première instruction de ce programme (valeur du compteur ordinal) 0700:0100
* Quel registres êtes-vous allez voir pour repérer cette adresse ? Compteur Ordinal CS IP
* Quel est le code assembleur de la première instruction du programme ? MOV CX, 00005h
* Quel est le code machine qui correspond à cette instruction ? B9 05 00
* Dans ce code qu’est ce qui correspond à l’opération et l’opérande ? B9 mov qq chose dans CX et 0005 est la valeur
* Allez voir en mémoire programme où est stockée l’instruction. Expliquez comment vous faite. On affiche la vue mémoire et on positionne l’adresse sur 0700:0100
* Exécutez en mode pas à pas et observer comment évolue le registre CX. Quelle est sa valeur avant et après l’instruction ? 006F puis 0005
* Quelle est la nouvelle valeur du compteur ordinal ? 0700 :0103
* A quoi correspond cette adresse ? adresse de la prochaine instruction
* De combien a été augmenter le compteur ordinal ? 3
* Selon vous à quoi est due cette augmentation ? à la taille de l’instruction qui vient de s’exécuter et qui était sur 3 octets
* Continuez l’exécution du programme en pas à pas, regardez l’évolution des registres et dites-moi ce que fait ce programme. Il fait la somme des éléments d’un vecteur

## Premier code assembleur

Regardez attentivement ce programme et répondez aux questions suivantes.

PILE SEGMENT STACK

DW 256 DUP(?)

pile:

PILE ENDS

DATA SEGMENT

N1 DW 5

N2 DW 3

N3 DW 3 DUP(?)

N4 DB ?

DATA ENDS

CODE SEGMENT

;ceci est un commentaire

; positionnement des registres de segment.

ASSUME CS:CODE, DS:DATA, SS:PILE

main:

MOV AX,DATA ; initialisation du segment de données

MOV DS,AX

MOV AX,PILE

MOV SS,AX ; initialisation du segment de pile

MOV SP,Base

; Code du programme

MOV AX,N1

CMP AX,N2

JE etiq

MOV N4,0

JMP fin

etiq:

MOV N4,1

fin:

MOV AH,4CH ; Les 2 lignes nécessaires pour la…

INT 21H ; …fin du programme

CODE ENDS

END main

1. Que fait ce programme ? il compare N1 et N2 et si ils sont égaux met N4 à 1 sinon N4 à 0

2. Quelle sera la valeur de N4 à la fin de l’exécution du programme? N4 =0

Les valeurs de N1, N2, N3 sont-elles modifiées ? Non

3. Si à la place de N2 DW 5, j’écris N2 DB 5, est-ce que le code fonctionnera

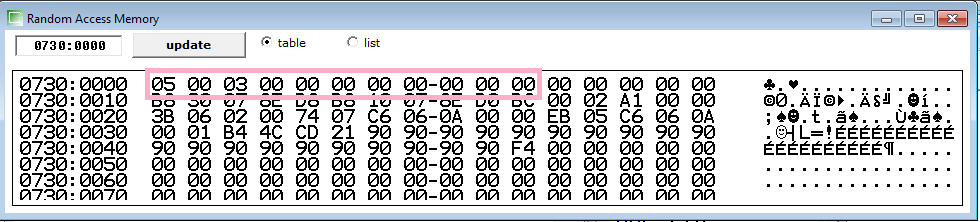
toujours? Non il ne sera pas compiler car le compilateur fait de la vérification de type et n’accepte pas que l’on déplace une variable de un octet dans un registre plus grand

Que dois-je changer pour que cela marche? Mettre N2 dans AL et non dans AX

4. La pile est composée de combien d’octets? 256 DW soit 512 octets

5. Combien de variables créées dans le segment de données ? 4 variables Quelle taille occupent-elles? 2+2+3\*2+1=11 octets

Représentez l’état du segment de données après la déclaration de N1, N2, N3 et N4.



A Quelle adresse se situe N4 ? 0730 :000A

6. Que se passe-t-il si vous enlevez l’instruction JMP fin ? le programme continue en séquence Quelle sera la valeur de N4 dans ce cas là à la fin de l’exécution du programme ?

N4 vaudra 1

Vous pouvez récupérer le fichier contenant ce programme dans R:\M211\TP1\simple.asm et vous pouvez le copier dans votre répertoire ASR1\TP1.

Lancez emu8086 (disponible dans R:\). Pour cela, créez un raccourci dans votre répertoire de travail ASR1. Une fois que emu8086 est lancé (double-cliquez sur le raccourci), cliquez sur Compilez et cliquez ensuite sur Run afin d’émuler le programme.

Une seconde fenêtre doit se lancer et émuler le programme. Vous pouvez faire une émulation pas-à-pas, c'est-à-dire instruction par instruction en cliquant sur Single step.

Ce mode va vous être particulièrement utile pour tester votre programme s’il ne fonctionne pas et pour le débugger.

Vous pouvez observer les valeurs des registres de travail, et des registres de segments.

1. Quels sont les registres de segments ? CS DS SS Comment sont-ils initialisés ? par le programme Pourquoi est-ce important de les initialiser ? pour que puisse se faire l’accès aux variables par le code.
2. Qu’est-ce que IP ? Instruction pointeur Quel est l’intervalle de valeurs qu’il prend durant le programme, au début et à la fin du programme ? F310 :0000 de à F310:0022 Pour observer cela, faites une exécution pas à pas de votre programme et regardez comment varie sa valeur. Regardez les valeurs prises par IP après le JE etiq 0016 et après le JMP fin 0024
3. Regardez l’état de la mémoire (View Memory) pour voir où sont stockés N1, N2, N3 et N4 et connaître leurs valeurs. Pour cela, regardez les 2 valeurs séparées par les deux points. A quoi correspondent-elles ? F320 est le numéro du segment de donnée ensuite c’est le numéro de l’octet par rapport au début du segment. Changez la première valeur pour accéder au segment de données. Et mettez la deuxième valeur à 0.
4. Créez deux nouvelles variables N5 et N6, qui sont des octets initialisés respectivement à 7 et à 8. Regardez où sont stockées ces deux nouvelles variables. Elles sont stockées à le suite des précédentes

