TP1 : Prise de contact avec le metteur au point

L'objectif de cette première séance est de vous faire prendre contact avec les logiciels qui seront utilisés par la suite pour les travaux pratiques d'assembleur, en particulier l'outil d'aide à la mise au point des programmes en assembleur.

Préparation

Nous allons utiliser un émulateur DOS (DosBox) pour tous les tp assembleur. Si vous ne l'avez pas encore fait, il faut télécharger l'archive *tasm.tar.gz* puis la désarchiver dans *votre répertoire personnel* (« home directory ». Ceci va créer le répertoire ~/.dosbox qui contient un fichier de configuration pour dosbox (à modifier, voir plus loin) ainsi que le répertoire ~/outils/TASM qui contient l'assembleur (*tasm*), l'éditeur de liens (*tlink*) et le debugger (*td*).

Ouvrez le fichier de configuration de dosbox avec un éditeur. Vers la fin se trouvent des commandes de montage de répertoire :

- mount c /home/jce/outils/TASM : répertoire des outils ; à adapter si besoin à votre organisation de répertoires ; dans dosbox, ce répertoire est désigné sous le « nom » c :
- mount d /home/jce/sys/asm : répertoire de vos fichiers sources assembleur ; à adapter si besoin à votre organisation de répertoires ; dans dosbox, ce répertoire est désigné sous le « nom » d :
- mount e . : cette ligne peut être supprimée
- e : ⇒ à remplacer par d :

1 Utilisation du metteur au point

Le logiciel « TurboDebugger » de Borland est destiné à faciliter la mise au point des programmes en assembleur. Il permet de charger un programme exécutable en mémoire, d'examiner le contenu des variables et des registres, et de contrôler l'exécution du programme.

1.1 Appel du metteur au point sur un petit exemple

On va d'abord créer le fichier exécutable *exemple.exe*. Vous devez :

- copier le fichier *exemple.asm* dans votre répertoire de TP.
- démarrer dosbox : dosbox &
- lister le contenu du répertoire : dir ; vous devriez voir le fichier *exemple.asm*.
- demander l'assemblage : c :tasm exemple /zi (vérifier la création du fichier *exemple.obj*)
- demander l'édition des liens : c :tlink exemple /v (vérifier la création du fichier *exemple.exe*)
- démarrer le metteur au point : c :td exemple

Au démarrage du metteur au point, il faut utiliser Alt-V C puis F5 pour obtenir l'affichage souhaité.

L'écran présenté par le metteur au point est composé de plusieurs volets : on peut passer de l'un à l'autre avec les touches Tab et Shift-Tab.

1.2 Volet de désassemblage

Le volet en haut à gauche donne le contenu de la zone de mémoire qui a reçu le code exécutable du programme d'exemple. Le metteur au point facilite l'interprétation du contenu de cette zone en montrant la représentation en langage d'assemblage de celui-ci (on dit qu'il désassemble). Remarquez bien que le metteur au point ne connaît pas les limites de la zone de code, en conséquence vous observez des instructions désassemblées parasites avant et après la zone de code significative.

1.3 Volet des registres

Le volet à droite du volet de désassemblage montre les valeurs des registres du 8086.

1.4 Volet d'état

Le volet en haut à droite montre les bits d'état qui sont utilisés par les opérations de test.

1.5 Volet d'affichage de la pile

Le volet en bas à droite montre l'état de la pile.

1.6 Volet d'affichage de la zone de donnée

Ce volet, en bas à gauche, permet de voir le contenu d'une zone de mémoire, qui est présentée dans son interprétation hexadécimale et ASCII.

2 Utilisation du metteur au point sur un exemple

1. Assurez-vous :
   — que le volet de désassemblage est le volet courant
   — que l'instruction courante est mov ax,@data ; elle est précédée du curseur ►, qui sert à montrer quelle est l'instruction désignée par le pointeur d'instruction ip.
   Ne pas confondre la barre de curseur, qui ne sert qu'à désigner une instruction, et n'a rien à voir avec la valeur de ip, et le curseur d'instruction, qui montre quelle est la prochaine instruction à exécuter.
2. Notez les valeurs courante de ip (**i**nstruction **p**ointer), ax et ds (**d**ata **s**egment), dans le volet des registres.
3. Tapez une fois sur la touche de fonction F7 : ceci exécute l'instruction désignée par IP ; observez la modification de ax et ip.
4. Frappez une autre fois sur F7, observez la modification de ds.
5. Prenez comme volet courant le volet d'affichage de la zone de donnée (utilisez la touche Tab plusieurs fois, ou mieux Shift Tab une seule fois), puis demandez l'affichage du début de la zone de donnée en frappant la séquence Ctrl-G ds :0 puis Entrée
6. Remarquez la place des variables *premier*, *second* et *gros* en mémoire. Remarquez également les valeurs initiales.
7. Avancez encore de deux instructions, remarquez également les changements en mémoire et interprétez-les.

3 Modification de la mémoire

Pour modifier le contenu de la zone des données, il suffit de positionner le curseur sur le premier octet à modifier, et taper ensuite les valeurs voulues.

— Dans le volet « données » placez le curseur sur l'octet d'adresse 38 (CTRL-G, DS :38) et entrez trois fois la valeur « douze », successivement en décimal, en hexadécimal et en binaire (frappez les valeurs l'une après l'autre, en les séparant par un espace, puis validez par « entrée »). Observez le résultat : l'affichage numérique et l'interprétation ASCII.
— Positionnez vous à l'adresse 3bH, et entrez de même, les valeurs hexadécimales « 41 », « 42 », « 43 »
— Positionnez vous à l'adresse 0

L'exemple comporte une boucle, repérez-la. La variable *premier* donne le nombre d'itérations à effectuer. Cette valeur étant égale à 100 (base 10), mettez-la à 5. Pour ce faire, effectuer les étapes suivantes :

1. Mettez le volet de données comme volet courant (Tab ou Shift Tab).
2. Placez le curseur (à l'aide des flèches) sur le premier octet du mot qui constitue la variable *premier*. Entrez 0005.
3. Observez le résultat. Interprétez.

4 Exécution d'une boucle

— Avancer en pas-à-pas en observant l'évolution des registres ax et bx. Observez l'évolution de la barre curseur sur le code lors de l'itération.
— Notez l'adresse de la troisième instruction du programme et replacez le pointeur d'instruction sur cette adresse (il est inutile de réexécuter les deux premières instructions, le registre ds n'ayant pas été modifié). Remarquez bien que seul ip est modifié par la manipulation ci-dessus, et bien que vous repreniez l'exécution au début, les variables ne reprennent pas leurs valeurs initiales. En bref, la valeur de ip ne détermine pas seule l'état de l'exécution de votre programme, et si vous voulez vraiment vous replacer dans les conditions initiales, il faut soit rétablir tous les registres et toutes les variables un par un « à la main », soit utiliser la commande Ctrl-F2

— Rétablissez les conditions initiales (Ctrl-F2) et vérifiez les valeurs des variables et des registres (notamment ip et ds). Si le volet de désassemblage montre des instructions fantaisistes au début du programme, employez la commande de rétablissement de l'origine (Ctrl-O), qui fait coïncider la barre curseur et l'instruction courante.
— Faites exécuter la boucle 15 fois, en pas-à-pas.

5 Utilisation de points d'arrêt

Le mode pas-à-pas est le mode de contrôle de l'exécution le plus fin possible, puisqu'on opère instruction par instruction. Une autre manière d'exécuter le programme est l'exécution libre ; dans ce mode, le programme s'exécute à pleine vitesse, hors du contrôle direct du metteur au point. Pour que ce mode libre soit utile pour la mise au point de programme, il est nécessaire de placer des points d'arrêt, qui agisse comme des panneaux « Stop » placés sur certaines instructions. Lorsque le 8086 atteindra une instruction où un point d'arrêt a été placé, l'exécution en mode libre cessera et l'utilisateur aura à nouveau le contrôle du metteur au point.

Un point d'arrêt ne provoque l'arrêt du mode libre que lorsque le 8086 atteint réellement l'adresse du point d'arrêt. Un saut « par dessus » un point d'arrêt ne déclenchera pas le retour au metteur au point.

1. Rétablissez les conditions initiales.
2. Mettez un point d'arrêt sur l'instruction de comparaison ; pour ce faire déplacez la barre de curseur sur cette ligne et frappez la touche F2. Remarquez qu'une instruction munie d'un point d'arrêt est soulignée dans le volet de désassemblage.
3. Pour plus de sûreté, mettez un autre point d'arrêt sur l'instruction nop en fin de programme. Ce point d'arrêt servira de garde-fou pour reprendre le contrôle du metteur au point lorsque la boucle s'achèvera.
4. Après avoir mis 10 dans la variable *premier*, faites exécuter le programme en mode libre en frappant la touche F9 : l'exécution débute dans l'état courant du programme et s'arrête sur le point d'arrêt. Vérifiez les valeurs des registres ax, bx et ip. Repartez en utilisant F9.
5. Recommencez l'étape précédente jusqu'à atteindre le point d'arrêt garde-fou. Vérifiez les valeurs des registres.
6. Supprimez le point d'arrêt posé sur l'instruction de comparaison : placez la barre curseur sur la ligne correspondante et frappez F2 (qui agit donc comme une bascule).
7. Repérez l'adresse de l'instruction add bx,ax. Placez un point d'arrêt sur cette instruction en utilisant la commande Alt-B A.Visualisez tous les points d'arrêt posés à l'aide de la commande Alt-V B.

Écriture d'un programme en hexadécimal (juste pour l'exemple !).

1. En vous servant du polycopié, écrivez le code hexadécimal d'un programme qui place les valeurs « huit » et « quatre» dans AL et AH, calcule leur somme, et range le résultat dans l'octet d'adresse 10.
2. Dans le volet « données », visualisez la zone d'adresse 100 dans la zone de code (CTRL-G, puis CS :100).
3. Entrez-y ce programme en hexadécimal (attention un nombre hexadécimal doit commencer par un chiffre).
4. Visualisez le programme dans le volet « désassemblage»
5. Initialisez IP avec l'adresse de début du programme
6. Faites exécuter le programme.