ECOLE SUPERIEURE D'INGENIEUR DE RENNES

UNIVERSITÉ DE RENNES 1

ESIR 1ére année option informatique

SYSTÈMES

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'ISE, en particulier le programme d'aide à la mise au point.

1 Conventions de notation

Ce document (et tous les énoncés de travaux pratiques) utilisent les conventions suivantes :

- L'expression Ctrl+G signifie qu'il faut maintenir la touche Ctrl (dite Contrôle) enfoncée pendant le frappe de la touche G.
- Maj désigne la touche majuscule (dite aussi Shift).
- Tab désigna la touche tabulation.
- Le texte en caractères bâton représente vos réponses entrées au clavier.
- Les caractères en oblique doivent être remplacés par le texte pertinent ; par exemple, *votre nom* équivaudra à Dupont, si vous vous appelez Dupont.

2 Appel 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.

2.1 Appel du metteur au point sur un petit exemple

Vous allez utiliser un fichier, de nom **exemple.asm**, contenant un programme exécutable, qui se trouve sur le "disque réseau" G; dans le répertoire **G:\ESIR1\ise.**

Vous devez choisir EDI tasm dans le menu:

Demarrer/Langages programmation/Tasm/Tasm.

Deux fenêtres vont s'afficher sur votre écran.

- La première est une fenêtre d'édition que vous pouvez utiliser pour saisir vos tps.

L'appel à turbo-debugger se fait par la commande: **td nom_de_fichier.**

-La deuxième est une fenêtre de commandes. Vous pouvez utiliser cette fenêtre pour copier le fichier **exemple.asm** dans le répertoire local c:\temp\tpsys\ à l'aide de la commande :

copy g:\ESIR1\ise\exemple.asm c:\temp\nom_de_login\

Vous devez demander l'assemblage/l'édition des liens et l'exécution de vos programmes à l'aide respectivement des commandes

TASM Nom_du_fichier_source
TLINK Nom_du_fichier_objet
TD nom_du_fichier_exécutable

Un message vous informe qu'il n'y a pas de table des symboles (on fait de la mise au point simple, non symbolique).

->Pour le faire disparaître appuyez sur la touche ***entrée***

L'écran présenté par le metteur au point est composé de plusieurs volets :

2.2 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.

2.3 Volet d'état du 8086

Le volet en haut à droite montre les valeurs des registres du 8086, lorsque celui-ci n'est pas en train d'exécuter le programme d'exemple (on dira qu'il est *stoppé*).

2.4 Volet d'affichage de la zone de donnée

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

3 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 (son mnémonique est précédé 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 lui montre bien quelle est la prochaine instruction à exécuter.

---

2. Remarquez la place des variables premier, second et gros en mémoire. Remarquez également les valeurs initiales.
3. Notez les valeurs courante de ip, ax et ds.
4. Tapez une fois sur la touche de fonction F7, observez la modification de ax et ip.
5. Frappez une autre fois sur F7, observez la modification de ds.
6. 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 Retour
7. Remarquez les valeurs affichées en première ligne du volet de donnée.
8. Avancer encore de deux instructions, remarquez également les changements en mémoire et interprétez-les.

4 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 les unes après les autres, en les séparant par un blanc, 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 touches Flèche) sur le premier octet du double mot qui constitue la variable premier. Entrez 0005 Retour
  3. Observez le résultat. Interprétez.

5 Exécution d'une boucle

1. 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.
2. 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 et « à la main », soit utiliser la commande Ctrl+F2

---

3. 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.
4. Faites exécuter la boucle 15 fois, en pas-à-pas.

6 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. Supprimer 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+F2. Visualisez tous les points d'arrêt posés à l'aide de la commande Alt+V.

7 Ecriture 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. Initialiser IP avec l'adresse de début du programme
6. Faites exécuter le programme.

```
; Programme d'aide à la prise en main de TurboDebugger
; Demande une seule zone de code et une seule zone de donnee
                .model  small
                .stack
                .data                   ; Ouverture des declarations pour la zone de donnees

tl_gros         equ     200             ; On declare une constante symbolique

premier         dw      100             ;
second          dw       -100           ; Declaration de deux variables double octet

gros            dw      tl_gros dup (0) ; Declaration d'un groupe de `tl_gros' double octets

message         db      'Message a afficher',0AH,0DH,'$'
                                        ;Declaration d'une chaine de caracteres, avec
                                        ; retour chariot inclus et marqueur de fin de
                                        ; chaine a la mode MS-DOS
                                        ; Noter qu'on emploie la directive `db'

                .code                   ; Ouverture des declarations pour la zone de code
debut:
                mov     ax,@data        ; Initialisation de DS
                mov     ds,ax           ; Maintenant les variables sont bien definies

                mov     bx,premier      ; On calcule la somme des entiers de 1 à
                mov     second,bx       ; [premier]

                mov     ax,1            ; Indice de boucle
                mov     bx,0            ; Accumulateur de somme partielle
tq:             cmp     ax,premier
                jg      fin_tq
                add     bx,ax
                add     ax,1
                jmp     tq
fin_tq:
                end     debut
```