



LMI-TD01: Outils de développement et set d'instruction de base en Assembleur (MS-DOS)

1 But du TD

- Premier contact avec l'environnement de travail sous Ms-Dos et Ms-Windows : un éditeur de texte, tasm, tlink, turbo debugger (td) et Helppc.
- Réalisation d'un premier programme en langage d'assemblage et exécution de celui-ci (en pas à pas) au sein de td.
- Familiarisation avec les instructions de base du langage d'assemblage et réalisation de petits programmes séquentiels.

Première partie: Outils de développement

2 Préparation de l'environnement de travail

2-1 Répertoire de travail

Pour programmer en assembleur il faut un compilateur. Sous Ms-Dos, en utilisera le compilateur Tasm. Ce dernier se trouve dans le eDistri\HAL\assembleur06-07 sous le nom de TASM_TD.ZIP. On vous demande de ne pas le déziper dans votre répertoire z:\. mais de le faire dans le répertoire c:\tasm. Si ce dernier existe déjà, écraser-le par un nouveau. Cependant, vos codes sources seront sauvés dans votre répertoire personnel.

2-2 Chemin d'accès (variable d'environnement PATH)

Il est possible de compiler un programme sous Ms-Dos sans devoir placer le fichier source dans le même répertoire que le compilateur. Pour ce faire il faut modifier la variable d'environnement PATH de sorte qu'elle contienne le répertoire où se trouve l'exécutable tasm.exe

Pour connaître la valeur de la variable d'environnement PATH, il suffit de taper path dans l'invite de commande, suivi de return.

La ligne d'instruction `path=c:\Tasm;%PATH%` ajoute le chemin c:\Tasm à l'ancien contenu de PATH.

Pour accéder directement à la documentation relative à la commande path, tapez `path /?`.

3- Programmation en assembleur

3.1 Edition du texte

Un éditeur de texte est un programme permettant de créer, modifier et sauvegarder un texte encodé en ASCII (Ms-Dos) ou ANSI (Ms-Windows).

Quelques éditeurs de texte: Edit sous Ms-Dos ; Notepad, Vim, gvim sous Ms-Windows).

Noter que certains éditeurs de texte dévolus à la rédaction de codes source sous Ms-Windows implémentent la très conviviale « coloration contextuelle » (ex. : context).

3.2 Compilation

Comme dit plus haut, le compilateur utilisé est tasm.

Pour l'invoquer, c'est-à-dire réaliser la traduction du code source en fichier objet (non exécutable) « .obj », vous devez :

1. En ligne de commande, vous placer dans le répertoire où le fichier source à compiler (essai.asm) est sauvegardé;
2. Encoder « tasm essai[.asm] /l/c/zi » (extension facultative) puis taper la touche « return ».

Une brève explication des options de compilation retenues :

- /l permet d'obtenir un listing du code source ;
- /c permet d'obtenir une table des références croisées (utilisée par le débogueur) ;
- /zi crée une table des symboles utilisée par td.

3.3 Édition des liens

Le fichier objet obtenu après compilation d'un source n'est pas exécutable. Pour le rendre exécutable, il faut réaliser l'édition des liens.

L'éditeur de lien utilisé est tlink.

Pour l'invoquer :

1. Être dans le répertoire où le fichier objet (essai.obj) est sauvegardé;
2. Encoder « tlink essai[.obj] /v » (extension facultative) puis taper la touche « return ».

L'option /v demande à tlink de générer le maximum d'informations possible à destination du débogueur (td).

3.4 Exécution

Après l'édition des liens, on obtient un fichier exécutable (binaire) « .exe ».

Pour exécuter le programme correspondant :

1. Être dans le répertoire où le fichier exécutable (essai.exe) est sauvegardé ;
2. Encoder « essai[.exe] » (extension facultative) puis taper la touche « return ».

3.5 Exécution au sein de turbo debugger

Un débogueur permet d'exécuter un programme instruction par instruction, c'est-à-dire en marquant une pause après l'exécution de chacune des instructions. Les contenus des registres (de travail et de flags) et des variables, mais aussi des segments de code CS, de données DS et de pile SS sont rendus visibles par le débogueur. Toutes les valeurs sont données en hexadécimal. Le debugger permet, entre autre, de mettre le doigt sur des erreurs de logique. Le débogueur utilisé est turbo debugger (raccourci en td).

LMI-TD01 : Outils de développement et sets d'instructions de base

Pour exécuter un programme (compilé avec tasm et dont les liens ont été édités avec tlink) dans turbo debugger, il faut:

1. Être dans le répertoire où le fichier exécutable (essai.exe) est sauvegardé ;
2. Encoder « td essai[.exe] » (extension facultative) puis taper la touche « return ».

Ceci étant fait, remarquez les différents menus déroulants et leurs commandes ainsi que les différentes fenêtres accessibles.

Parmi les commandes « incontournables » de td :

- Alt-x : pour quitter td ;
- F2 : pour marquer un point d'arrêt;
- F7 : pour exécuter le programme en marquant une pause après chaque instruction ;
- F9 : pour exécuter le programme jusqu'à son terme ou un éventuel point d'arrêt ;
- Alt-v suivi de r (menu View : ligne Registers), pour observer les contenus de tous les registres ;
- Ctrl-F7 : pour mettre une variable (ou un registre) sous observation ;
- Ctrl-F4 : pour évaluer et modifier un registre ou une variable :
 - Utiliser la touche de tabulation ;
 - Valeur encodée par défaut en hexadécimal (précédée si nécessaire d'un 0 (zéro)), pour fournir une valeur en décimal ou en binaire, la faire suivre de d ou b, respectivement ;
 - Frapper la touche Escape pour quitter ;
- F6 : pour changer de fenêtre en avant-plan ;

Vous découvrirez d'autres fonctionnalités de td en cours d'année !

3.6 Documentation

Le Helppc est un programme qui fournit de la documentation sur l'ensemble des instructions des processeurs de la famille du x86 allant du 8086 au 486, sur les directives standard du langage d'assemblage, sur les interruptions Bios et Dos et sur bien d'autres thèmes relatifs à la programmation assembleur. En cas de problème ou de doute quant à l'utilisation d'une instruction, d'une directive ou d'un service d'interruption, ayez le réflexe de consulter Helppc. La réponse à votre question s'y trouve probablement !

Pour installer Helppc, un fichier auto-extractible du même nom (HELPPC.EXE) se trouve dans le répertoire eDistri\HAL\assembleur06-07 il suffit de le placer dans votre z:\ et de faire un double clic. Le résultat de cette action, est la création d'un répertoire Helppc.

Bien entendu, cette source d'informations ne minimise pas l'intérêt d'acquérir un ouvrage de référence relatif au langage d'assemblage. Quelques titres d'ouvrage suivent :

- Assembleur X86, *Kip Irvine* (Campus Press)
- Assembleur X86, *Jean-Bernard Emond* (Campus Press)
- La programmation en langage d'Assemblage *Jean-François Nadeau* (Modulo-Briffon)

4 Premier programme en langage d'assemblage : `essai.asm`

1 ; NVS

LMI-TD01 : Outils de développement et sets d'instructions de base

```
2 ;
3 ;=====
4 ; essai.asm
5 ;
6 ; Auteur : NVS, HAL
7 ; Date : 17/12/06 , 21/01/07 ( révisions mineures )
8 ; Description : premier programme en langage d'assemblage
9 ;
10 ;=====
11
12 ;=====
13 ; DIRECTIVES COMPILATEUR
14 ;=====
15 .MODEL small
16 .STACK 100h
17
20 ;=====
21 ; DONNEES
22 ; Déclarat ion et définition des variables (DB, DW, DD, DT)
23 ;=====
24 .DATA          ; début du segment de données
25 jour            DB          5
26 mois           DB          ?
27 c              DW          8
28 annee          DW          1985
29 annee_naissance DW          ?
30 ;=====
31 ; CODE SOURCE
32 ;=====
33 .CODE          ; début du segment de code
34 ;=====
35 ; Programme principal
36 main PROC      ; programme principal
37 ; ----- Intro -----
38 mov ax , @data          ; initialisation du segment de données
39 mov ds , ax
40
41 ; ----- Programme -----
42 MOV     bl , 10
43 MOV     ch , 10 h
44 MOV     mois , 12
45 MOV     dl , jour
46 MOV     mois , dl
47 MOV     AX , c
48 MOV     BX , AX
49 MOV     DX , annee
50 MOV     annee_naissance , DX
51 ; ----- épilogue -----
52 mov ax , 4C00h
53 int     21h          ; interruption de fin de programme
54 main ENDP
```

```
55
56 ; =====
57 ; FIN FICHIER SOURCE ET POINT D'ENTREE
58 ; =====
59 END main
```

Tout comme pour vos laboratoires de Java on vous demandera :

- De bien commenter vos sources
- D'indenter le code source.

Ce code source se trouve dans le edistri\HAL sous le nom `essai.asm`. Copiez-le dans votre répertoire de travail `z:\...`. Notez que les noms de fichier sous Dos doivent respecter un maximum de huit (8) caractères sans l'extension en l'occurrence « `.asm` » ici !

1. Compilez ce code source, faites l'édition des liens et exécutez-le pas à pas dans `td`.
2. Observez les contenus des registres et des variables y apparaissant
3. Que fait l'instruction `MOV` ?
4. Observez l'évolution du registre `IP`.
5. Pourquoi `IP` n'évolue t'il pas de façon linéaire ?
6. Remplacez la valeur de début de la variable `jour` en 13 sans quitter `td`.
7. Est-il possible de changer la valeur d'une variable, d'un registre à n'importe quel moment sous `td` ?
8. Est-il possible de remplacer les instructions 45 et 46 par l'instruction suivante :
`MOV mois,jour`

4.2 Fichier batch

Un fichier batch est un fichier texte exécutable par le système d'exploitation. Il ne s'agit pas d'un programme, mais d'un script, c'est-à-dire d'une suite de commandes à exécuter. Il est fabriqué à l'aide d'un éditeur de texte et sauvegardé avec l'extension « `.bat` ».

Le fichier batch vous permet d'alléger l'encodage en ligne de commande d'une série de commandes (ici commande de compilation, d'édition des liens et d'exécution).

Pour exécuter un fichier batch préalablement édité, il suffit d'encoder son nom (extension facultative) dans la ligne de commande, à condition que son chemin d'accès se trouve dans `PATH`.

Il est en outre possible de transmettre des paramètres lors de l'exécution d'un script. Dans le fichier batch, ces paramètres sont notés `%1`, `%2`, . . ., `%9`, où `%1` fait référence au premier paramètre, `%2` au second, etc.

Ainsi, le fichier batch `go.bat` :

```
tasm %1 /l/c/zi
tlink %1 /v
td %1
```

permet d'assembler, de faire l'édition des liens puis d'exécuter au sein de `td` le fichier qui lui est fourni en argument (obligatoirement sans extension). Ceci bien entendu à condition qu'aucune erreur ne se produise lors de l'une de ces étapes ! Un exemple typique d'utilisation de ce fichier batch est : `go essai`

Deuxième partie : Instructions arithmétiques

5- Instructions arithmétiques sur des entiers non signés et signés

5-2 Soustraction entière

Ex2. Soit v1 une variable de la taille d'un octet et v2 et v3 deux variables de taille deux octets. Ecrivez un code qui stocke dans v3 la différence des contenus de v1 et v2 ($v3 = v1 - v2$).

Testez votre code à l'aide de td avec les mêmes valeurs que dans l'exercice relatif à l'addition entière.

5-3 Multiplication entière

5-3-1 Entiers non signés

Ex3. Soient v1 et v2 deux variables de taille un octet.

Quelle taille (minimale) doit avoir la variable v3 destinée à recevoir le produit des contenus de v1 et v2 ?

Ecrivez un code qui stocke ce produit dans cette variable v3.

Testez votre code dans td avec les valeurs suivantes :

- v1 : 2 et v2 : 10 ;
- v1 : 20 et v2 : 30 ;
- v1 : 255 et v2 : 255.

Qu'observez-vous ?

Peut-on multiplier deux variables de taille différentes (ex. : v1 est un byte et v2 un word ?) si oui comment ?. Consulter le Helppc pour l'instruction CBW

5-3-2 Entiers signés

Ex4. Reprenez le code de l'Ex3 et testez-le avec les valeurs : v1 : -1 et v2 : 10.

Que constatez-vous ?

Ecrivez une modification du code de l'Ex3 permettant le traitement d'entiers positifs ou négatifs, c'est-à-dire d'entiers signés.

Testez ce nouveau code avec les valeurs :

- v1 : 2 et v2 : 10 ;
- v1 : -10 et v2 : 10 ;
- v1 : 10 et v2 : -10 ;
- v1 : -10 et v2 : -10 ;
- v1 : 255 et v2 : 255.

Qu'observez-vous ?

6- Exercice récapitulatif :

L'exercice suivant, servira à évaluer votre travail au cours de ce premier travail dirigé.

Ex9. Calculer la valeur de Prod suivante en respectant les consignes demandées :

$$\text{Prod} = 6 * (2 + (A - B)) - (10 * H + (2 * P)^2)$$

P = 3 et A, B et H des nombres entiers

Ecrivez un code qui stocke dans la variable Prod le résultat de ce calcul.

Les variables A et B et H ont une taille respectivement un, deux et un octet.

LMI-TD01 : Outils de développement et sets d'instructions de base

Testez dans td le bon fonctionnement de votre programme (l'exactitude du résultat pour différentes valeurs de A et B et H).

Vous devriez être capable au terme de ce travail d'expliquer correctement le comportement de votre programme à votre professeur.