



HEB-ESI
1^{re} année

Année 2005-2006
BEJ, HAL, JCJ,
NVS, PBT

LMI-TD00 : Outils de développement (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 au sein de `td`.

2 Découverte et utilisation des outils de développement

2.1 Éditeur de texte

Il s'agit d'un programme permettant de créer, modifier et sauvegarder un texte encodé en ASCII (MS-DOS) ou ANSI (MS-WINDOWS).

Contrairement à un logiciel de traitement de texte, aucune mise en forme autre que par le biais de « caractères » (espace, retour à la ligne, tabulation) n'est possible avec un éditeur de texte. Cependant, certains éditeurs de texte dévolus à la rédaction de codes source implémentent la très conviviale « coloration contextuelle ».

2.2 Compileur

L'assembleur utilisé est `tasm`.

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

1. ouvrir une fenêtre d'« Invite de commande » ;
2. se placer dans le répertoire où le fichier source à compiler (`monPrg.asm` ici) est sauvegardé ;
3. éventuellement modifier la variable d'environnement `PATH` de sorte qu'elle contienne le répertoire où se trouve l'exécutable `tasm.exe` (voir le point 4.1 pour plus de détails à ce propos) ;

4. encoder « `tasm monPrg[.asm] /l/c/zi` » (extension facultative) puis frapper 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 `td`) ;
- `/zi` crée une table des symboles utilisée par `td`.

2.3 Éditeur de liens

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

L'éditeur de lien utilisé est `tlink`.

Pour l'invoquer :

1. si ce n'est déjà fait, ouvrir une fenêtre d'« Invite de commande » ;
2. si ce n'est déjà fait, se placer dans le répertoire où le fichier objet (`monPrg.obj` ici) est sauvegardé ;
3. si ce n'est déjà fait, éventuellement modifier la variable d'environnement `PATH` de sorte qu'elle contienne le répertoire où se trouve l'exécutable `tlink.exe` (*a priori* même répertoire que celui où se trouve `tasm.exe`) ;
4. encoder « `tlink monPrg[.obj] /v` » (extension facultative) puis frapper la touche « return ».

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

2.4 Exécution

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

Pour exécuter le programme correspondant :

1. si ce n'est déjà fait, ouvrir une fenêtre d'« Invite de commande » ;
2. si ce n'est déjà fait, se placer dans le répertoire où le fichier exécutable (`monPrg.exe` ici) est sauvegardé ;
3. encoder « `monPrg[.exe]` » (extension facultative) puis frapper la touche « return ».

2.5 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 et des variables, mais aussi des segments de code, de données et de pile sont rendus visibles par le débogueur. Ceci permet donc, par exemple, de mettre le doigt sur des erreurs de logique.

Le débogueur utilisé est `turbo debugger` (raccourci en `td`).

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

1. si ce n'est déjà fait, ouvrir une fenêtre d'« Invite de commande » ;
2. si ce n'est déjà fait, se placer dans le répertoire où le fichier exécutable (`monPrg.exe` ici) est sauvegardé ;
3. si ce n'est déjà fait, éventuellement modifier la variable d'environnement `PATH` de sorte qu'elle contienne le répertoire où se trouve l'exécutable `td.exe` (*a priori* même répertoire que celui où se trouve `tasm.exe`) ;
4. encoder « `td monPrg[.exe]` » (extension facultative) puis frapper 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` ;
- `F9` : pour exécuter le programme jusqu'à son terme ou un éventuel point d'arrêt ;
- `F7` : pour exécuter le programme en marquant une pause après chaque instruction ;
- `F8` : pour exécuter le programme en marquant une pause après chaque instruction mais en considérant un appel de procédure comme une seule instruction ;
- `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 `Escape` pour quitter ;
- `F6` : pour changer de fenêtre en avant-plan ;
- `Ctrl-F5` : pour déplacer ou redimensionner une fenêtre : flèches pour déplacer, flèches du pavé numérique pour redimensionner, frapper `return` pour valider.

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

2.6 Documentation

L'ensemble des instructions des processeurs de la famille du x86 allant du 8086 au 486, les directives standard du langage d'assemblage, les interruptions BIOS et DOS et bien d'autres thèmes relatifs à la programmation assembleur sont décrits et documentés dans `helppc`. 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 ! Bien entendu, cette source d'informations ne minimise pas l'intérêt d'acquérir un ouvrage de référence relatif au langage d'assemblage.

3 Un premier programme en langage d'assemblage

Voici un minuscule code source en langage d'assemblage :

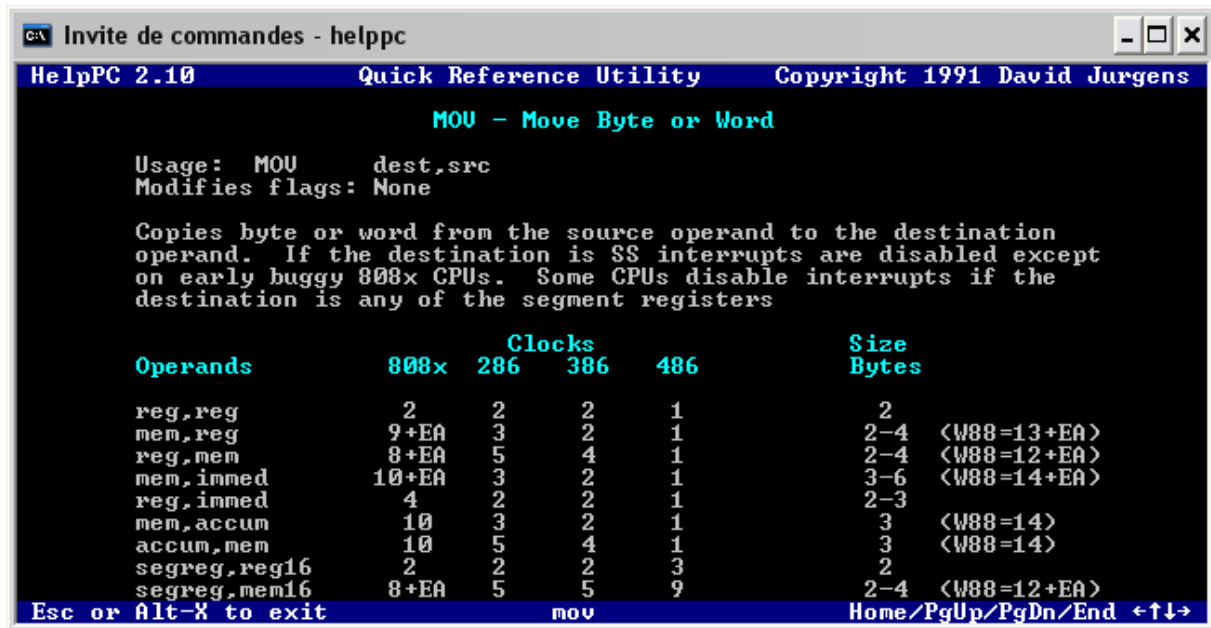


FIG. 1 – Utilisation de helppc.

```

1 ;
2 ; Nicolas Vansteenkiste (NVS)
3 ;
4 ; =====
5 ; monPrg.asm
6 ;
7 ; Auteur       : Nicolas Vansteenkiste (NVS)
8 ; Date        : 07/11/04, 17/01/06 (révisions mineures)
9 ; Description  : premier programme en langage d'assemblage
10 ;
11 ; =====
12 ;
13 ; =====
14 ; DIRECTIVES COMPILATEUR
15 ; =====
16 .MODEL small
17 .STACK 100h
18 ;
19 ; =====
20 ; DONNÉES
21 ; Déclaration et définition des variables (DB, DW, DD, DT)
22 ; =====
23 .DATA
24 a   DB 5
25 b   DB ?
26 ;
27 ; =====
28 ; CODE SOURCE
29 ; =====
30 .CODE

```

```

31 ; =====
32 ;   Programme principal
33 ; =====
34
35 main PROC
36 ; ----- Intro -----
37 mov  ax,@data
38 mov  ds,ax
39
40 ; ----- Programme -----
41 mov  bl,10
42 mov  ch,10h
43 mov  dl,a
44 mov  b,dl
45
46 ; ----- Épilogue -----
47 mov  ax,4C00h
48 int  21h
49 main ENDP
50
51 ; =====
52 ; FIN FICHER SOURCE ET POINT D'ENTRÉE
53 ; =====
54 END main

```

Remarquez l'abondance de commentaires. Veillez vous aussi à **bien commenter** vos sources !

Remarquez également la mise en page obtenue essentiellement à l'aide de tabulations.

Encodez ce source à l'aide de l'éditeur de texte choisi par votre professeur (`edit`, `CONTEXT`, `vim` ...). Sauvez le fichier source sous le nom « `monPrg.asm` ». Notez que vous pouvez changer ce nom (en respectant un maximum de huit (8) caractères) mais *pas* l'extension « `.asm` » ! Réalisez la sauvegarde à l'endroit (le répertoire, `Z:\...`) indiqué par votre professeur. Lors de l'encodage du source, n'hésitez pas à régulièrement sauvegarder votre travail !

Après l'avoir édité, compilez ce source. Ensuite, éditez les liens du fichier objet obtenu. Vous pouvez maintenant exécutez le programme correspondant ou l'exécutez pas à pas dans `td` en observant les contenus des registres et des variables `y` apparaissant (voir FIG. 2).

4 Quelques mots sur MS-DOS

4.1 Chemin d'accès

Les commandes internes (`dir`, `cd` ...) sont directement reconnues par `cmd.exe`. Elles peuvent donc être exécutées quel que soit le dossier courant.

Par contre, les commandes externes, par exemple `tasm.exe`, ne peuvent, par défaut, être exécutées à partir de la ligne de commande que si le répertoire courant est celui qui contient l'exécutable en question.

Il est cependant possible de permettre l'exécution, à partir de la ligne de commande, d'un

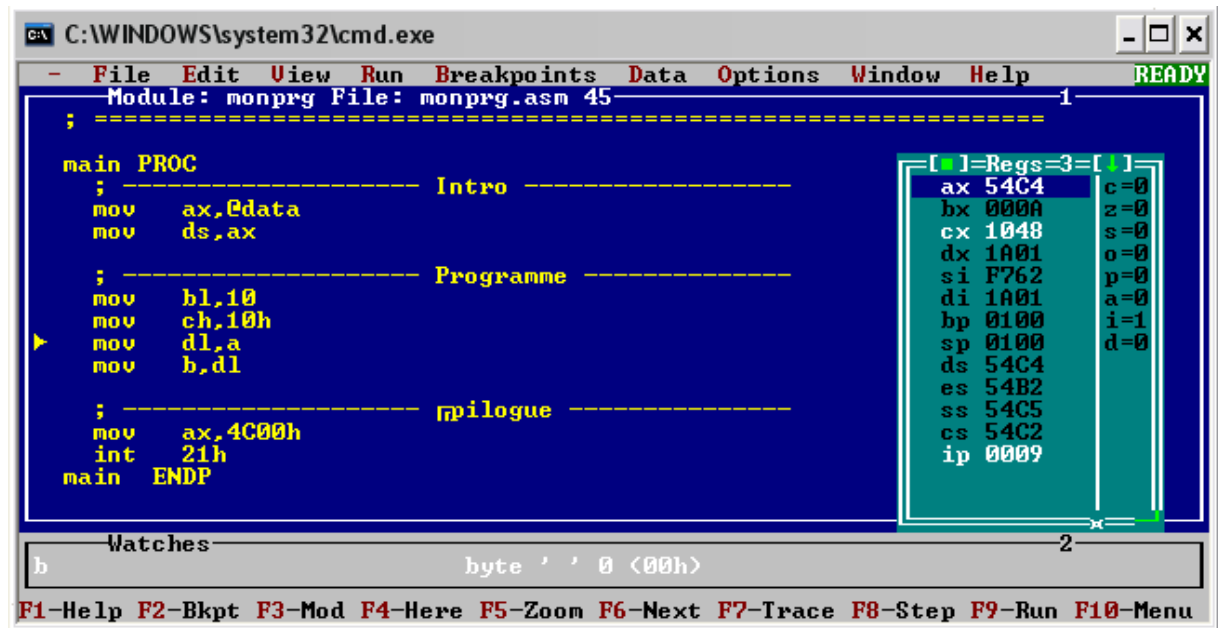


FIG. 2 – Utilisation de td.

programme quelconque et ce indépendamment du répertoire courant. Pour cela, il faut que la variable d'environnement PATH contienne le chemin d'accès (l'emboîtement des répertoires depuis une partition, un disque racine) de cet exécutable.

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

Pour modifier le contenu de PATH, il faut utiliser la commande interne `path=chemin d'accès`. Ainsi, la commande `path=c:\Tasm` modifie la valeur de PATH de sorte qu'elle soit égale à `c:\Tasm`. Si on ne désire pas écraser le contenu précédent de PATH mais simplement lui ajouter un chemin, il faut exécuter `path=chemin d'accès;%PATH%`. Donc, la ligne d'instruction `path=c:\Tasm;%PATH%` ajoute le chemin `c:\Tasm` au contenu de PATH.

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

4.2 Fichier batch

Un fichier batch est un fichier texte exécutable. 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` ».

Les fichiers batch permettent d'alléger l'encodage d'instructions en ligne de commande, en particulier lorsque plusieurs instructions doivent être lancées à la chaîne.

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

Ainsi, soit un fichier batch constitué des deux lignes suivantes :

```
PATH=%PATH%;c:\Tasm  
dir
```

et sauvé sous le nom `bro1.bat`. Son exécution a pour effet de modifier `PATH` et de lister le contenu du répertoire courant. Remarquez que *chaque* exécution de ce batch ajoute le répertoire `c:\Tasm` au chemin d'accès, ce qui risque de poser problème étant donné la taille limitée des variables d'environnement (dont `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 `asmTd.bat` :

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

permet d'assembler, d'éditer les 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 produit lors de l'une de ces étapes ! Un exemple typique d'utilisation de ce fichier batch est : `asmTd monPrg`.