

## Informatique 5 – Langage assembleur SMI – Semestre 3 Le programme DEBUG de MSDOS

### I. Syntaxe :

DEBUG permet de mettre au point des fichiers exécutables

#### Syntaxe :

**Debug** [*unite*:[*chemin*]*fichier*[*param*]

#### Paramètres :

**[unite] : [chemin]fichier[param]** : Précisent l'emplacement et le nom du fichier exécutable à tester.

**param** : Précise toute information à inclure sur la ligne de commande exigée par le programme à tester.

Si vous ne spécifiez pas le fichier à mettre au point, vous entrez en mode "commande" de DEBUG

### II. Les commandes de DEBUG

#### **A (Assemble)**

Assemble les mnémoniques directement en mémoire. La commande crée du code machine exécutable à partir d'instructions en langage d'assemblage.

#### Syntaxe :

**a** [*adresse*]

#### Paramètres :

**adresse** : Précise l'emplacement où commence l'assemblage. Si ce paramètre est omis, l'assemblage commence à l'adresse où la commande s'est arrêtée lors de sa dernière utilisation ou à défaut par l'adresse par défaut utilisée par DEBUG.

#### Exemples

```
-a
0CC9:0103 mov ah,9
0CC9:0105 int 21
0CC9:0107
-a 200
0CC9:0200 db "une chaine de caractère...$"
0CC9:021B
```

#### **C (Compare)**

Compare deux zones de la mémoire.

#### Syntaxe :

**c** *plage* *adresse*

#### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la première zone de mémoire à comparer.

**adresse** : Précise l'adresse de départ de la deuxième zone de mémoire à comparer

#### Exemples

```
-c 100,105 200
0CC9:0100 BA 75 0CC9:0200
0CC9:0101 00 6E 0CC9:0201
0CC9:0102 02 65 0CC9:0202
0CC9:0103 B4 20 0CC9:0203
0CC9:0104 09 63 0CC9:0204
0CC9:0105 CD 68 0CC9:0205
```

#### **D (Dump)**

Affiche le contenu d'une zone de mémoire en deux parties : une partie sous forme d'octet en hexa et une autre en format texte ASCII.

#### Syntaxe :

**d** [*plage*]

### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la zone de mémoire à afficher.

Si l'adresse de fin (ou la longueur) n'est pas spécifiée, la commande affiche les 128 octets à partir de l'adresse de début spécifiée

Si le paramètre est omis, la commande affiche les 128 octets à partir de l'adresse de fin mentionnée lors de la dernière utilisation de la commande, ou à défaut à partir de l'adresse par défaut utilisée par DEBUG

### Exemples

```
-d ds:200,225          ou      d ds:200 1 26
```

### **E (Enter)**

Permet d'entrer des données en mémoire à l'adresse spécifiée. Les données peuvent être soit en hexa soit une chaîne entre guillemets. Toute donnée stockée à l'adresse spécifiée est perdue.

### Syntaxe :

**e** *adresse* [*liste*]

### Paramètres :

**adresse** : Précise la première adresse à partir de laquelle les données seront stockées.

**liste** : Précise les données à ranger dans les octets successives à partir de l'adresse spécifiée.

Si ce paramètre est omis, la commande affiche l'adresse et le contenu du premier octet et attend une entrée :

La touche ESPACE permet de passer à l'octet suivant, la touche – (moins) permet de revenir à l'octet précédent et la touche ENTREE permet de mettre fin à la commande.

### Exemples

```
-e ds:203
0CC9:0203  20.-
0CC9:0202  40.42
-e ds:200 "écriture d'une donnée avec la commande e"
```

### **F (Fill)**

Permet de remplir une zone de mémoire avec une donnée.

### Syntaxe :

**f** *plage* *liste*

### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la zone de mémoire à remplir.

**liste** : Précise les données à entrer. Les données peuvent être soit en hexa soit une chaîne entre guillemets.

La commande affecte les valeurs de la liste de manière répétitive jusqu'à ce que tous les octets de la plage soient remplis.

Si la liste contient plus d'octets que celui de la plage, les valeurs supplémentaires sont ignorées.

### Exemples

```
-f ds:200 1 25 00
-d ds:200
0CC9:0200  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0CC9:0210  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0CC9:0220  00 00 00 00 00 00 65 20 65-20 72 82 70 65 72 74 6F .....e e r.perto
```

### **G (Go)**

Exécute le programme chargé en mémoire.

### Syntaxe :

**g** [= *adresse*] [*pointsArrêt*]

### Paramètres :

**=adresse** : Précise l'adresse à partir de laquelle le programme sera exécuté. Si l'adresse n'est pas spécifiée, le programme sera exécuté à partir de l'adresse courante CS:IP des registres.

Le signe = permet de distinguer entre l'adresse de départ et les points d'arrêt.

**pointsArrêt** : Précise entre 1 et 10 points d'arrêt temporaires pour la commande.

### Exemples

```
-g cs:0107
Un programme exécuté avec DEBUG

AX=0924 BX=0000 CX=012E DX=010B SP=FFFE BP=0000 SI=0000 DI=0000
DS=0D21 ES=0D21 SS=0D21 CS=0D21 IP=0107 NV UP EI PL NZ NA PO NC
0D21:0107 B44C          MOV     AH,4C
```

### H (Hex)

Donne la somme et la différence de deux nombres hexadécimaux.

#### Syntaxe :

**h** *valeur1 valeur2*

#### Paramètres :

**valeur1, valeur2** : Deux nombres hexadécimaux compris entre 0 et FFFFh. La commande affiche *valeur1+valeur2* et *valeur1-valeur2*.

### Exemples

```
-h 125 a8
01CD 007D
```

### I (Input)

Lit et affiche la valeur d'un octet venant d'un port.

#### Syntaxe :

**i** *port*

#### Paramètres :

**port** : Précise l'adresse du port d'entrée. Cette adresse est codée sur 16 bits

### Exemples

```
-i 0379
DF
```

### L (Load)

Charge en mémoire le fichier spécifié dans la ligne de commande de DEBUG ou le dernier fichier nommé par la commande n la plus récente. Cette commande permet aussi de charger le contenu de secteurs d'un disque.

#### Syntaxe :

**l** [*adresse [unite depart nombre]*]

#### Paramètres :

**adresse** : Précise l'adresse où commencera le chargement du fichier ou des secteurs du disque.

Si ce paramètre est utilisé tout seul, la commande charge en mémoire le fichier (spécifié sur la ligne de commande de DEBUG ou nommé précédemment par la commande n) à partir de l'adresse spécifiée.

Si ce paramètre est omis, la commande charge ce dernier fichier à partir de l'adresse CS:100 et place dans BX et CX le nombre d'octets chargés.

S'il s'agit d'un fichier .EXE, le paramètre adresse est ignoré.

**unite** : Précise l'unité de disque : 0=A, 1=B, 2=C, etc.

**depart** : Précise le numéro (en hexa) du premier secteur à charger

**nombre** : Précise le nombre (en hexa) de secteurs à charger

### Exemples

```
-n tpdebug1.com
-l

-l cs:100 0 0 20
```

### M (Move)

Déplace le contenu d'un bloc de mémoire vers un autre bloc.

#### Syntaxe :

**m** *plage adresse*

#### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la zone de mémoire source à copier

**adresse** : Précise l'adresse de départ de la zone de destination.

### Exemples

```
-m cs:0100,105 cs:200
```

### **N (Name)**

Précise un nom de fichier pour une commande l (load) ou w (write) ou spécifie des paramètres pour le fichier exécutable mis au point.

#### Syntaxe :

**n** [*chemin*] *fichier*

**n** *param*

**n**

#### Paramètres :

**[chemin]fichier** : Précise l'emplacement et le nom du fichier à utiliser avec la commande l ou w, ou à mettre au point.

**param** : Précise les paramètres et commutateurs pour le fichier exécutable à mettre au point.

La commande n sans paramètres efface les spécifications en cours

### Exemples

```
-n prog.com
-l
-n param1 param2
-g

debug prog.com
-n param1 param2
-g
```

### **O (Output)**

Envoie la valeur d'un octet vers le port spécifié.

#### Syntaxe :

**o** *port* *valeur*

#### Paramètres :

**port** : Précise l'adresse du port de sortie.

**valeur** : Précise la valeur de l'octet à envoyer au port.

### Exemples

```
-o 0378 aa
```

### **P (Proceed)**

Exécute une boucle, une sous-routine ou plusieurs instructions successives.

#### Syntaxe :

**p** [= *adresse*] [*nombre*]

#### Paramètres :

**=adresse** : Précise l'adresse de la première instruction à exécuter.

Si ce paramètre est omis, la commande utilise l'adresse CS:IP contenue dans les registres.

Le signe = permet de distinguer entre l'adresse et le nombre.

**nombre** : Précise le nombre d'instructions à exécuter. La valeur par défaut est 1.

### Exemples

```
-p =0105 6
```

### **Q (Quit)**

Quitte DEBUG.

#### Syntaxe :

**q**

### **R (Register)**

Affiche ou modifie le contenu d'un ou de plusieurs registres.

#### Syntaxe :

**m** [*registre*]

#### Paramètres :

**registre** : Précise le nom du registre à consulter ou à modifier.

Si ce paramètre est omis, la commande affiche l'état de tous les registres.

Les noms des registres valides : ax, bx, cx, dx, sp, bp, si, di, ds, es, ss, cs, ip(ou pc) et f

Les indicateurs d'état sont désignés par deux lettres

Noms	Valeur si = 0	Valeur si = 1
Débordement	NV	OV
Direction	UP	DN
Interruption	EI	DI
Signe	PL	NG
Zéro	NZ	ZR
Auxiliaire	NA	AC
Parité	PO	PE
Retenue	NC	CY

#### Exemples

```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CC9 ES=0CC9 SS=0CC9 CS=0CC9 IP=0100 NV UP EI PL NZ NA PO NC
0CC9:0100 EB3C JMP 013E
-r ax
AX 0000
:0ffa
-r f
NV UP EI PL NZ NA PO NC -ng zr
```

#### S (Search)

Cherche une liste d'octets dans la zone mémoire spécifiée.

##### Syntaxe :

**s** *plage* *liste*

##### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la zone de mémoire à examiner.

**liste** : Précise les valeur d'octets ou la chaîne à rechercher.

Les données peuvent être soit en hexa soit une chaîne entre guillemets.

#### Exemples

```
-s ds:00,300 "AB"
-s cs:00,300 6D
0D31:0017
0D31:0022
0D31:0023
0D31:002D
0D31:01A2
```

#### T (Trace)

Exécute une instruction et affiche le contenu des registres.

##### Syntaxe :

**t** [= *adresse*] [*nombre*]

##### Paramètres :

**=adresse** : Précise l'adresse de l'instruction à exécuter.

Si ce paramètre est omis, le commande utilise l'adresse CS:IP contenue dans les registres.

Le signe = permet de distinguer entre l'adresse et le nombre.

**nombre** : Précise le nombre d'instructions à exécuter. La valeur par défaut est 1.

#### Exemples

```
-t
AX=0D32 BX=0000 CX=0033 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0D21 ES=0D21 SS=0D31 CS=0D31 IP=0003 NV UP EI PL NZ NA PO NC
0D31:0003 8ED8 MOV DS,AX
```

#### U (Unassemble)

Désassemble des octets et affiche les instructions en assembleur correspondantes.

##### Syntaxe :

**u** [*plage*]

#### Paramètres :

**plage** : Précise les adresses de départ et de fin ou l'adresse de départ et la longueur de la zone de mémoire à décoder.

Si l'adresse de fin (ou la longueur) n'est pas spécifiée, la commande décode les 32 octets à partir de l'adresse de début spécifiée

Si le paramètre est omis, la commande décode les 32 octets à partir de l'adresse de fin mentionnée lors de la dernière utilisation de la commande, ou à défaut à partir de l'adresse par défaut utilisée par DEBUG

#### Exemples

-u			
0D31:0000	B8320D	MOV	AX,0D32
0D31:0003	8ED8	MOV	DS,AX
0D31:0005	BA0000	MOV	DX,0000
0D31:0008	B409	MOV	AH,09
0D31:000A	CD21	INT	21
0D31:000C	B44C	MOV	AH,4C
0D31:000E	CD21	INT	21

#### **W (Write)**

Ecrit un fichier ou des secteurs spécifiques sur disque

Le fichier concerné est celui spécifié dans la ligne de commande de DEBUG ou est le dernier fichier nommé par la commande n la plus récente.

Cette commande ne permet pas d'écrire un fichier EXE ou HEX.

#### Syntaxe :

**w** [adresse [unite depart nombre]]

#### Paramètres :

**adresse** : Précise l'adresse de début du fichier (ou d'une partie du fichier) en mémoire à écrire sur disque

Si ce paramètre est utilisé tout seul, la commande écrit le contenu du nombre d'octets spécifié dans les registres BX:CX.

Si ce paramètre est omis, la commande commence l'écriture à partir de l'adresse CS:100.

Si une des commandes g, t, p ou r est utilisée, il faut réinitialiser les registres BX et CX, avant d'utiliser la commande w sans paramètres

**unite** : Précise l'unité de disque : 0=A, 1=B, 2=C, etc.

**depart** : Précise le numéro (en hexa) du premier secteur où les données seront écrites

**nombre** : Précise le nombre (en hexa) de secteurs à écrire

#### Exemples

-n tpdebug1.com
-w