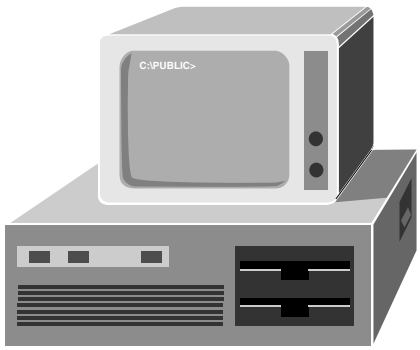
Aide–mémoire PC et MSDOS

Gérard Blanchet



Ecole Nationale Supérieure des Télécommunications
46 Rue Barrault 75634 Paris Cedex 13
Téléphone (33-1) 45 81 74 19 Télécopie (33-1) 45 88 79 35
E-mail blanchet@sig.enst.fr

TABLE DES MATIÈRES

AVERTISSEMENT

Les micro-ordinateurs du type IBM-PC et compatibles forment une famille de plusieurs dizaines de millions de machines dans le monde. Malgré ses faiblesses avérées, MS–DOS a constitué pendant des années le système d'exploitation de base de ces machines. Par la suite OS/2, "les" Windows 1, 2, 3, 95, 98 et 2000 puis NT (*Windows New Technology*) ont pris le relais. Ce document constitue un aide-mémoire pour ceux désirant comprendre le fonctionnement interne de ces machines. En effet, en dépit d'une évolution importante et très bénéfique pour l'utilisateur, bon nombre d'éléments architecturaux ont été conservés. Ce document explique en particulier comment appeler les fonctions résidant en mémoire morte (le BIOS) ou celles du système d'exploitation (le DOS) à partir de programmes écrits en langage évolué ou en langage d'assemblage. Ce n'est qu'à partir de Windows 95 que l'accès au matériel se fait en mode protégé. Cela exige une connaissance des mécanisme mis en œuvre par l'architecture 386 ainsi que celle des librairies d'accès aux ressources matérielles. On se limite donc ici à des informations sur :

– l'organisation de la mémoire des PC et les données que l'on y trouve,
– l'organisation des disques sous MS/DOS (*MicroSoft/Disk Operating System*),
– la réalisation de *Device Drivers*,
– l'appel aux fonctions du BIOS (*Basic Input/Output System*),
– l'appel aux fonctions du DOS (*Disk Operating System*).

Les fonctions seront toutes décrites de la façon suivante :

N° d'interruption N° de fonction	Version DOS Dénomination de la fonction
	AH fonction Paramètres d'entrée
	Paramètres de sortie
	Remarques

Les appels en langage évolué se font tous selon le même schéma : on déclare un structure de données devant recevoir le contenu des registres et on appelle une fonction ou une procédure qui active une interruption *software*.

– En MS/BASIC la structure de données associée aux registres du 8086 a pour nom `RegTypeX`. Sa déclaration se fait de la façon suivante :

```
' déclaration du type
TYPE RegTypeX
     ax    AS INTEGER
     bx    AS INTEGER
     cx    AS INTEGER
     dx    AS INTEGER
     bp    AS INTEGER
```

```
     si    AS INTEGER
     di    AS INTEGER
     flags AS INTEGER
     ds    AS INTEGER
     es    AS INTEGER
END TYPE

' déclaration des variables
DIM iRegIn AS RegTypeX, iRegOut AS RegTypeX
' appel de la procédure (ici on appelle la fonction 4
' du DOS par l'interruption de numéro hexa 21).
iRegIn.ax = &H400: 'numéro de fonction transmis dans AH=4, AL=00
iRegIn.bx = &H****: 'autres paramètres à transmettre
…
IntNum = &H21: CALL interruptX(IntNum, iRegIn, iRegOut)
Param1% = iRegOut.ax: 'récupération des paramètres
…
```

– En C-MicroSoft les appels aux fonctions du DOS nécessitent le fichier d'en-tête `dos.h` :

```
#include <dos.h>
union REGS inregs, outregs;
struct SREGS segregs;
int resultat, Param1;
/* paramètres d'entrée */
inregs.h.ah = 4;
inregs.x.bx = ****;
…
resultat = int86x(0x21, &inregs, &outregs, &segregs);
/* on pourrait utiliser  intdos(inregs, &outregs) à condition
   de ne pas utiliser DX ou AL
   ou intdosx(inregs, &outregs, &segregs) */
…
```

– Exemple en Turbo Pascal :

```
Uses Dos;
var regs : Registers;
    resultat, Param1 : integer;
…
{ paramètres d'entrée }
regs.ah := 4;
regs.bx := ****;
…
intr($21, regs); { on pourait appeler directement msdos(regs) }
Param1 := regs.ax;
…
```

CHAPITRE I

Processeur et mémoire

Les premiers IBM-PC étaient dotés de processeurs Intel I8088 dotés d'une horloge de base de fréquence 4,77 *MHz*. Actuellement des systèmes à base de micro-processeurs Intel Pentium à vitesse d'horloge interne de 700 *MHz* sont disponibles. Les performances des processeurs ont donc été augmentées dans un rapport de l'ordre de 50 et 100. Pendant longtemps, et malgré une augmentation considérable des performances, ces machines ont gardé un point commun : leur système d'exploitation qui, en dépit de nombreux perfectionnements, a conservé ses principales caractéristiques et faiblesses parmi lesquelles une gestion mémoire inexistante. Seul le premier méga-octet de mémoire était géré alors que le processeur pouvait adresser 4 téra-octets (architecture 386) de mémoire virtuelle sur un adressage physique de 4 giga-octets. Windows 1, 2 et 3 continuaient de passer par des appels à MS/DOS pour nombre de ses fonctions mais dispose d'une gestion de mémoire plus élaborée avec un espace d'adressage de 16 méga-octets de mémoire (architecture 286).

I.1. Quelques rappels sur les processeurs utilisés

Les microprocesseurs sont basés dur l'architectuer 80386 de chez Intel Corp. ou proviennent d'autres fondeurs tels que IBM, Cyrix ou AMD. Utilisés sous MS/DOS ces processeurs fonctionnent en mode *réel*, c'est-à-dire en un mode compatible 8086. Un programme écrit en assembleur 8086 doit en principe fonctionner quel que soit le microprocesseur utilisé, du 8088 ou Pentium. Les futurs processeurs (Merced) introduisent des concepts architecturaux très fortement teintés de parallélisme.

I.1.1. Structure d'adressage des processeurs 80X86 en mode réel

L'espace d'adressage des 80X86 et 80X88 en mode de fonctionnement dit *réel* est de 1 Mo. Cependant le jeu d'instructions ne permet d'accéder à un instant donné qu'à un espace de 64 ko.

I.1.1.1. Les registres de segmentation

En raison de sa structure 16 bits, le processeur n'a accès à un instant donné qu'à 64 kilo-octets de mémoire de programme, de données ou de pile. Le passage de cet espace accessible à l'espace d'adressage physique est réalisé à l'aide de registres de *segments*. L'adresse fournie par le compteur

ordinal ou émise lors de l'exécution de l'instruction est ajoutée au contenu d'un des registres de segment de la façon suivante :

adresse physique=16×segment + déplacement

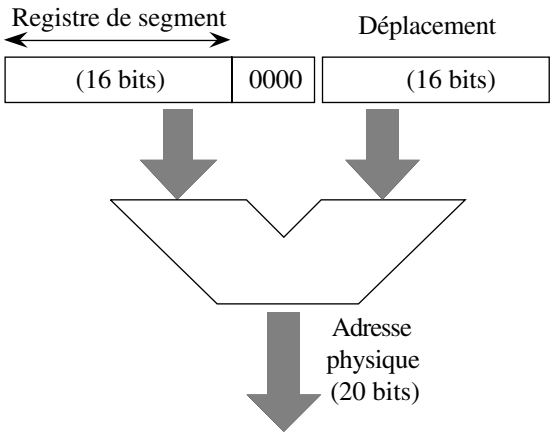


Fig.1 – *Structure d'adressage des processeurs I8086*

Chaque segment a une taille de 64 kilo-octets et commence sur une frontière de *paragraphe* (groupe de 16 octets) à l'adresse physique $XXXX0_{16}$. Le jeu d'instructions particularise le rôle des registres de segment. Ainsi une instruction d'écriture du contenu d'un registre vers la mémoire utilise par défaut le registre de segment DS, tandis qu'une écriture dans la pile (instruction `PUSH`) utilise le registre SS.

On dispose, dans ce mode de fonctionnement, de quatre registres de segmentation :

– le premier, CS, définit le segment dans lequel le processeur va chercher les instructions (*instruction fetch*),
– DS et ES sont utilisés pour les accès aux données,
– SS est associé aux lectures et écritures dans la pile.

segment de code CS
segment de données DS
segment de pile SS
segment supplémentaire ES

Fig.2 – *Les registres de segment des I8086*

I.1.1.2. Les registres de travail

Les registres de travail sont utilisés pour la manipulation des données dans le processeur. Ils peuvent éventuellement être utilisés comme registres d'index :

accumulateur AX
base BX
compteur CX
donnée DX

Fig.3 – *Les registres de travail des I8086*

I.1.1.3. Les registres d'adressage et d'index

Le pointeur de pile donne le déplacement (*offset* ou adresse dans le segment) du "sommet" de pile. Le pointeur BP sert d'index dans la pile, tandis que SI et DI servent généralement d'index dans les instructions répétitives de déplacement de blocs de données :

pointeur de pile SP
pointeur de base BP
index de source SI
index de destination DI

Fig.4 – *Les registres d'adressage des I8086*

I.1.1.4. Les registres de fonctionnement

Le compteur ordinal IP (*Instruction Pointer*) donne le déplacement dans le segment de code (défini par le contenu de CS) de l'instruction à charger dans le registre instruction. Les drapeaux sont un ensemble de bascules donnant l'état du processeur après exécution d'une instruction (par exemple "résultat nul" ou résultat donnant lieu à une retenue", etc …).

Compteur ordinal IP
Drapeaux (*flags*)

Fig.5 – *Les registres de fonctionnement des I8086*

I.1.2. Structure d'adressage des processeurs 80286 en mode protégé

Les processeur 80286 peuvent travailler en mode dit *réel* compatible avec le 8086.

En *mode protégé* l'adresse émise, dite *virtuelle*, est sur 30 bits. Les 13 bits de poids forts (*selector*) du registre de segment servent de pointeur dans une table de description de segment (*segment descriptor table*) de 8192 entrées.

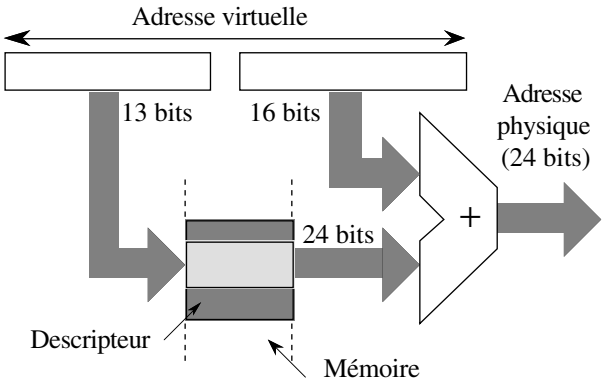


Fig.6 – *L'adressage des I80286*

Le descripteur de segment comporte 8 octets. L'*adresse de base* sur 24 bits qui y est rangée est ajoutée aux 16 bits de poids faible de l'adresse virtuelle et fournit l'adresse physique en mémoire du segment de 64K. Le bit 2 du même registre de segment indique dans quelle table (variables *globales*

ou *locales*) l'adresse physique de segment sera lue. On dispose donc de 4 Giga-octet de mémoire virtuelle avec un espace physique de 16 Méga-octets.

Lors du *Reset* le processeur se trouve en mode *réel*. Le passage en mode *protégé* exige de préparer le 80286. Cela est effectué en initialisant la table des vecteurs d'interruption (instruction LIDT) et en positionnant à 1 le bit PE du registre d'état permettant de contrôler le mode de travail du processeur et la présence d'un autre processeur :

```
mov     ax,1
lmsw    ax
```

Le retour au mode réel passe par un *reset*. Celui-ci est provoqué en utilisant une des lignes de sortie du contrôleur UPI qui passe à 0 pendant environ 6 µs.

```
mov     al,0FEH
out     64H,al
```

La spécification LIM/EEMS (*Lotus/Intel/Microsoft Enhanced Expanded Memory Specification*) permettait aux programmes écrits sous DOS d'accéder à l'espace mémoire disponible en mode *protégé* (interruption 67H).

I.1.3. Le cas du 80386 et du 80486

Ces processeurs ont une architecture beaucoup plus complexe que leur prédécesseurs. Le système d'adressage permet d'accéder à des tailles plus confortables (4 Go en mode *protégé* non paginé) et gère la *mémoire virtuelle* (jusqu'à 2^{46} octets de mémoire virtuelle). Ces processeurs permettent de toute façon de travailler en mode réel, donc en mode 8086. Un mode dit *mode 8086 virtuel* est utilisé par un certain nombre de produits commerciaux pour faire tourner plusieurs machines virtuelles 8086 sur le même processeur 80386 ou 80486. Chaque machine virtuelle possède son propre espace d'adressage de 1 Mo. Le système de pagination du 386 permet d'implanter cet espace n'importe où dans les 4 Go d'espace mémoire physique. On peut remarquer que dans une telle situation on a pratiquement un DOS plus sûr que le DOS d'origine.

L'architecture d'adressage du microprocesseur 80386[(1)] d'Intel en mode *protégé* est donné figure 7. Un premier niveau d'adressage fait appel à la segmentation. Les adresses sont fournies sous la forme *sélecteur de* segment et *déplacement*.

Le sélecteur est un registre de 16 bits dont les 13 bits de poids fort donnent un déplacement en mémoire vers un descripteur de segment de 8 octets. On a donc 8192 descripteurs de 8 octets. Le bit 2 du sélecteur indique si l'on a affaire à une table de descripteurs locaux ou globaux, et les deux bits de poids faible fournissent le niveau de privilège de la tâche à laquelle ce segment est associé. Chaque descripteur de segment comprend une adresse de base sur 32 bits, la longueur du segment sur 20 bits (2^{20} blocs de 4 Koctets au maximum), et des bits de service (droits d'accès et informations de service). L'espace mémoire virtuel adressable est donc de 16 kilo-fois (deux tables de 8 kilo-descripteurs) 4 giga-octets soit 64 téra-octets alors que l'espace physique est “limité” à 4 giga-octets.

[(1)]INTEL Corp. "80386 High Performance Microprocessor with Integrated Memory Management" Documentation Technique

La longueur des segments est définie par le champ *limite* sur 20 bits. Celui-ci donne le nombre de *granules* occupé par le segment. Un bit dit *bit de granularité* indique la taille d'un granule : un octet ou 4 K-octets en fonction de la valeur de ce bit. Ainsi la taille maximum du segment peut être de 1 Méga-octet ou de 4 Giga-octets.

Le système de gestion de mémoire virtuelle assure une pagination à deux niveaux sur les segments.

Deux caches sont ajoutés pour améliorer les performances de l'ensemble. Le premier (*cache de descripteur*) reçoit le contenu du sélecteur et les huit octets du descripteur associé. En fait, il ne s'agit pas vraiment d'un cache mais de registres, non accessibles au programmeur, qui conservent une copie du descripteur. Le second (*Translation Lookaside Buffer*) est un cache associatif par bloc de quatre blocs de 32 entrées, destiné à améliorer les temps d'accès sur les étages réalisant la pagination. Avec une taille de pages de 4 Koctets, cela donne un espace accessible de 128 Koctets. D'après Intel, le taux de succès sur les références (*hit rate*) est de l'ordre de 98% pour les systèmes multi-tâches courants.

Le système de pagination présenté peut être invalidé pour travailler seulement en mode segmenté. Le microprocesseur I80386 possède un mode de fonctionnement appelé *mode vituel 8086* dans lequel il est possible de définir des espaces de travail de 1 Méga-octets dans l'espace total de 4 Giga-octets. Ces espaces sont utilisés par le processeur I80386 comme par un I8086 standard. Par contre, leur emplacement physique en mémoire est géré par le mécanisme de pagination, offrant ainsi une grande souplesse de gestion de l'espace mémoire total.

Les *bits* de service présents dans chaque entrée du répertoire ou de la table de pages (figure 7) sont utilisés de la façon suivante :

– le *bit P* (*Present*) indique que cette entrée peut être utilisée dans la traduction d'adresse et donc être chargée dans le *TLB*,
– le *bit* A (*Accessed*) est mis à 1 lors d'une utilisation en lecture ou en écriture à l'adresse référencée dans cette entrée,
– le *bit* D (*Dirty*) est mis à 1 lors d'une écriture à l'adresse indiquée dans cette entrée,
– les *bits* U/S (*User/Supervisor*) et R/W (*Read/Write*) sont utilisés dans le mécanisme de protection des pages,
– trois *bits* sont utilisables par le système d'exploitation pour mettre en œuvre l'algorithme de remplacement des pages.

Si une entrée n'est pas présente dans le *TLB*, le bit *P*, noté P_r, du répertoire indique si la table de pages correspondante est ou non présente en mémoire. Si P_r=1, l'entrée correspondante de la table de page est lue et le bit A_r mis à 1. Si le bit P_t de la table de pages est égal à 1, le processeur met à jour les bits A_t et D_t et accède à la donnée. Si P_r ou P_t est égal à 0 alors l'exception 14 (*page fault*) est engendrée. L'adresse (adresse linéaire) de l'instruction ayant causé le défaut de page est rangée dans le registre de contrôle CR2. Un mot de 16 bits de *code erreur* (bits R/W, U/S et P) est écrit dans la pile du programme de traitement de l'exception.

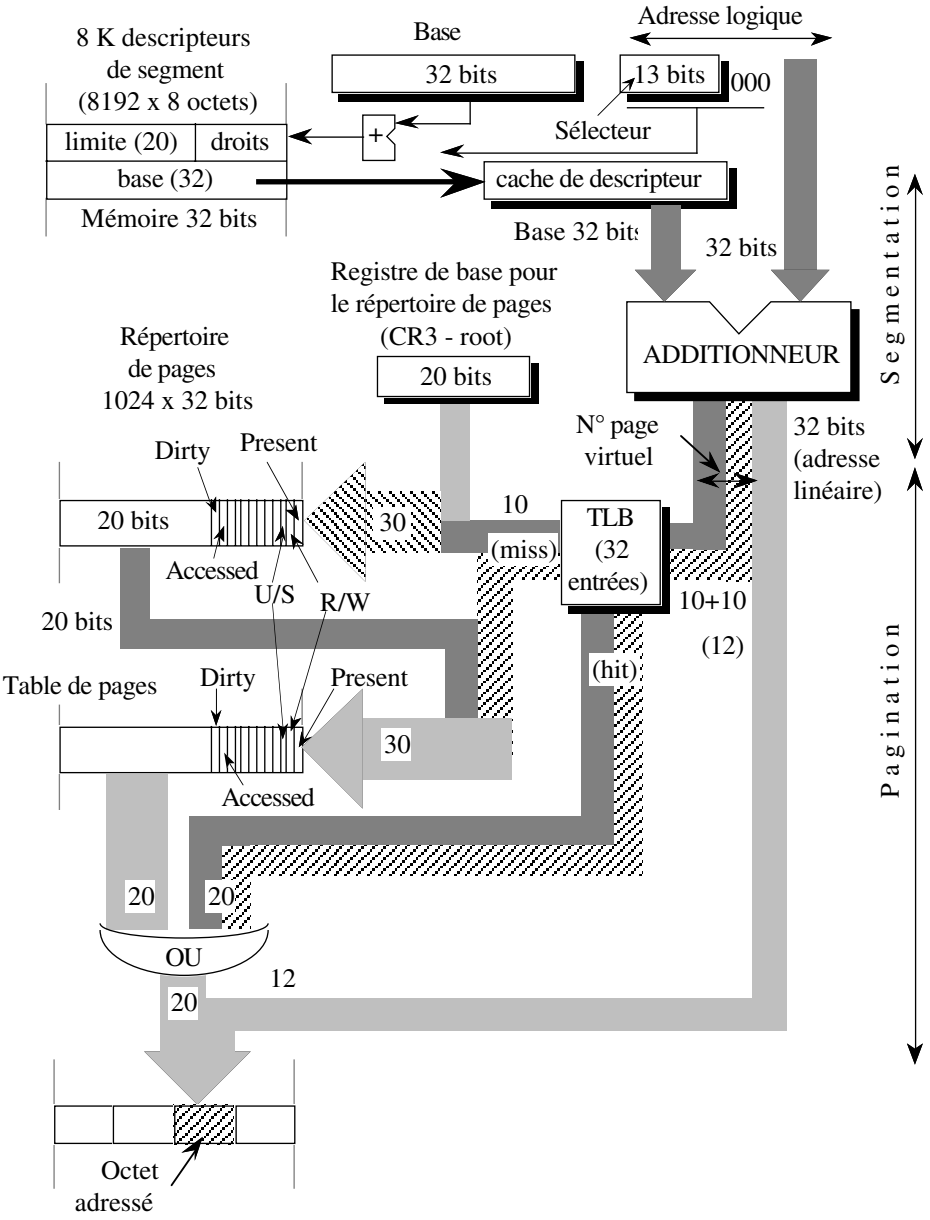


Fig.7 – *Segmentation paginée dans le microprocesseur* I80386

I.1.4. Les modèles mémoire

L'organisation physique de la mémoire explique pourquoi les compilateurs offrent ce que l'on désigne par *modèle de mémoire*. Selon que l'on s'autorise un ou plusieurs segments pour le *code* ou les *données* cela a des conséquences directes sur la taille des pointeurs et les adresses de branchements engendrées par le compilateur.

Les différents modèles sont repertoriés dans le tableau suivant. On note n_c le nombre de segments de code et n_d le nombre de segments de données :

Modèle	nombre de segments	
tiny	n_c=1, n_d =1	Programmes .COM, CS=DS=ES=SS
small	n_c=1, n_d =1	CS différent de DS=SS=ES
medium	n_c>1, n_d =1	
compact	n_c=1, n_d >1	
large	n_c>1, n_d >1	la taille des structures de données reste <64K
huge	n_c>1, n_d >1	la taille des structures de données peut être >64K
flat	*non segmenté*	accès à 4 Go linéaires (OS2/2.x par exemple)

Les langages tels que BASIC, FORTRAN ou PASCAL Microsoft utilisent essentiellement les modèles *medium*, *large* ou *huge*.

I.1.5. Organisation générale matérielle et logicielle des PC

Les ressources matérielles de la machine (y compris les éléments périphériques qui lui sont rattachés) sont accessibles à partir des fonctions prédéfinies organisées en "couches" autour du noyau matériel que constitue la machine :

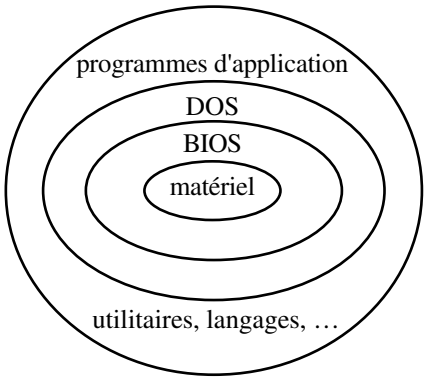


Fig.8 – *Organisation en couches matérielles et logicielles*

Le BIOS (*Basic Input Output System*) est constitué d'un ensemble de programmes résidant en mémoire morte et chargés de la gestion de l'affichage sur écran, des lecteurs de disques souples ou durs, de l'imprimante, du clavier, des lignes de communication série.

Le système d'exploitation MSDOS offre un ensemble de fonctions permettant :

– d'accéder aussi aux ressources matérielles de la machine (clavier, écran, …),

– de gérer l'espace mémoire,

– de charger et lancer l'exécution de programmes se trouvant sur un support de stockage externe (en général les disques),

– de gérer les informations enregistrées sur disque,

– de connecter des périphériques non standards.

I.2. L'organisation mémoire

La famille des processeurs 80X86 dispose d'un espace mémoire et d'un espace d'entrées-sorties qui sont séparés. Ceci implique, entre autres choses, que l'on dispose de deux types d'instructions de

lecture/écriture : `MOV`, `MOVS` … pour les accès à la mémoire, et `IN`, `OUT` … pour les accès à l'espace des entrées/sorties

I.2.1. L'espace mémoire

L'espace mémoire comprend essentiellement trois zones :

– 640 kilo-octets de mémoire vive (*RAM*) utilisée pour les programmes,
– 64 kilo-octets de mémoire morte pour les programmes du BIOS,
– le reste utilisé pour les unités d'entrées-sorties (disques durs, affichage vidéo, etc.).
Cet espace mémoire peut être schématisé de la façon plus précise :

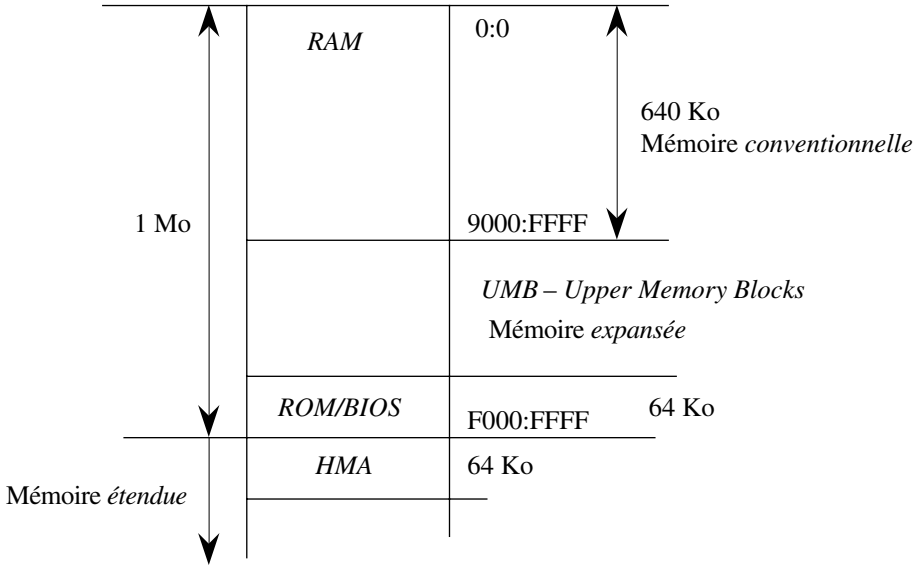


Fig.9 – *Organisation de la mémoire*

I.2.1.1. Organisation générale

Adresse (seg:dépl.)	Contenu
0:0	Trappes (0 à 7)
0:20	Vecteurs d'interruption du contrôleur I8259 d'interruptions (8 à F)
0:0086	Valeur du segment MSDOS.SYS
0:009E	Valeur du segment partie résidente COMMAND.COM
0:0400	Zone de communication de la mémoire morte
0:0500	Zone de communication DOS
0:0600	Début zone programmes
0:0600	IO.SYS
XXXX:0000	MSDOS.SYS
	– Tampon répertoire
	– Tampon disque
	– Blocs de paramètres : disque/FAT

XXXX:0000	Partie résidente de COMMAND.COM – Gestion INT 22, INT 23, INT 24 – Chargeur partie transitoire de COMMAND.COM – Commandes internes
XXXX:0000	Programmes .COM ou .EXE
XXXX:0000	Pile utilisateur pour fichier .COM
XXXX:0000	Partie transitoire COMMAND.COM – interpréteur de commandes – commandes internes – processeur de fichiers batch
A000:0000	Tampons d'affichage
A800:0000	"
B000:0000	"
B800:0000	"
C000:0000	BIOS affichage EGA
C800:0000	Adaptateur disque fixe (ROM)
D000:0000	*Cluster Adapter Bios*
FE00:0000	BIOS

I.2.1.2. Interruptions vectorisées

Les cases mémoire d'adresse physique 0 à $3FF_{16}$ sont en principe réservées pour y ranger les adresses de traitement des interruptions. Une adresse est constituée de quatre octets : deux octets pour spécifier le segment et deux autres pour le déplacement dans ce segment. A l'interruption numéro *n* sont associées les adresses rangées de la façon suivante :

adresse physique	contenu
4 *n*	déplacement poids faible (*offset*)
4 *n* + 1	déplacement poids fort
4 *n* + 2	segment poids faible
4 *n* + 3	segment poids fort

Les interruptions peuvent être activées par le matériel ou par le logiciel :

– dans le premier cas, un dispositif externe (le contrôleur d'interruptions) fournit à l'unité centrale un signal d'interruption sur le bus de contrôle. Le processeur, après avoir terminé l'instruction en cours, renvoie un acquittement. Le contrôleur fournit alors, sur le bus de données, le numéro de cette interruption. L'unité centrale charge alors dans le registre CS et dans le compteur ordinal l'adresse de traitement enregistrée en mémoire selon le schéma vu précédemment.

– dans le second cas, on peut activer le traitement d'une interruption à l'aide d'une instruction désignée par INT *n* où *n* est le numéro de l'interruption. Le mécanisme d'appel est identique à celui mis en œuvre pour les interruptions matérielles. Ainsi les fonctions du système d'exploitation seront appelées à l'aide de l'instruction INT 21H (21 hexadécimal), en transmettant dans les registres les paramètres nécessaires à l'exécution de cette fonction.

Les *PC* ne disposent que d'un seul contrôleur d'interruption I8259, alors que les AT en disposent de deux, mis en cascade.

Les lignes IRQ (*interrupt request*) sont ramenées sur les contrôleurs. Lorsqu'une interruption survient (transition de 0 à 1 ou niveau 1 sur la ligne correspondante d'IRQ), celle-ci est traitée par le contrôleur (qui effectue la gestion des priorités) qui répercute, si nécessaire, cette interruption vers le processeur par la ligne INT. Le processeur délivre ensuite deux cycles d'acquittement. Pendant le second, le contrôleur fournit un numéro d'interruption sur huit bits (lignes D0 à D7).

L'initialisation du contrôleur d'interruption consiste à définir un mode de fonctionnement *compatible 8086*, un mode de contrôle *maître* ou *esclave*, le type d'IRQ (front montant ou niveau), le contenu d'un registre de masquage et la valeur des cinq bits de poids fort du numéro qui sera envoyé vers le processeur. Le programme d'interruption doit, en fin d'exécution, revalider les contrôleurs d'interruption par écriture de la commande EOI (*End Of Interrupt*) à l'adresse 20H et éventuellement aussi à l'adresse A0H. Chaque IRQ possède ainsi son propre vecteur d'interruption qui lui est associé. Pour les lignes qui ne sont pas utilisées, le traitement est renvoyé vers INT 0AH (traitement de l'IRQ 2).

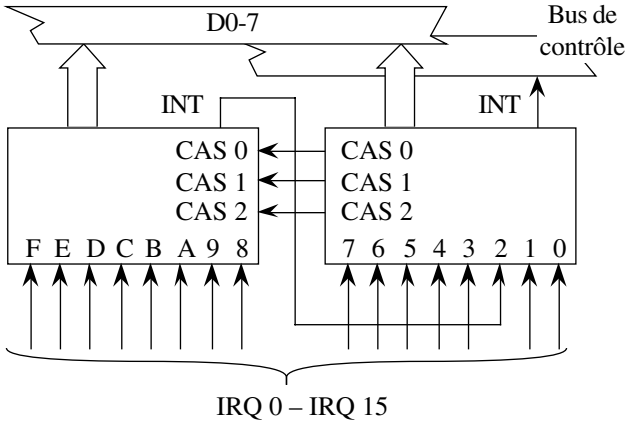


Fig.10 – *Contrôleurs d'interruption*

On peut inhiber une IRQ en écrivant à l'adresse 21H (contrôleur maître) ou A1H (contrôleur esclave) un masque d'interruption. Au bit de poids *i* correspond l'IRQ$_i$. Une mise à 1 du bit *i* inhibe IRQ$_i$.

Exemple d'inhibition des interruptions clavier :

```
in al,021H  ; lecture du masque courant
or al,2     ; mise à 1 (inhibition) du bit 1
out 21H,al  ; à partir de maintenant les interruptions clavier (INT 9)
            ; ne sont plus traitées
```

Attention : si rien n'est prévu pour revalider l'interruption clavier, il faudra redémarrer la machine !

Les principales interruptions utilisées dans un *PC/XT*® ou *PC/AT*® sont les suivantes :

N°(hexa)	Fonction
0	traitement division par zéro (instructions DIV et IDIV)
1	pas à pas
2*	NMI
3	Point d'arrêt
4	exception *overflow*
5	INTR (interruption masquable), utilisé pour *hardcopy* d'écran
7	exception (présence de l'extension processeur, instructions ESC ou WAIT)
8* (IRQ 0)	Horloge (18,2 *ticks* par seconde). Quartz à 1 193 180 Hz. Un *tick* tous les 64K impulsions.
9* (IRQ 1)	Clavier
A* (IRQ 2)	cascade vers IRQ8 à IRQ15 (inutilisé sur *PC*)
B* (IRQ 3)	COM2
C* (IRQ 4)	COM1
D* (IRQ 5)	LPT2 sur *AT*, disque dur sur *PC*.
E* (IRQ 6)	Contrôleur de disquettes
F (IRQ 7)	LPT1
10	Gestion des écrans
11	Test configuration (contrôle d'équipement)
12	Taille mémoire
13	Accès disquette
14	Communications RS232C
15	Cassette
16	Clavier
17	Imprimante
18	Pointeur vers BASIC en ROM ou appel moniteur
19	Pré-amorce
1A	Horloge/Calendrier
1B	Gestion <CTRL Break>
1C	Traitement des interruptions d'horloge (laissé à la libre disposition des utilisateurs). Appelée lors du traitement de l'interruption "matérielle" 8.
1D	Pointeur vers table d'initialisation vidéo
1E	Pointeur vers table de paramètres disquettes
1F	Pointeur vers table de caractères graphiques CGA
33	Gestion de la souris.
35 à 3F	Vecteurs utilisés pour fonctions coprocesseur.
40	Contrôle disque (BIOS).
41	Pointeur vers paramètres premier disque dur (contrôleur non ESDI)
42	Gestion BIOS écran EGA.
43	Pointeur vers paramètres d'initialisation en mode EGA
44	Pointeur vers table de caractères graphiques en mode EGA
46	Pointeur vers paramètres deuxième disque dur (contrôleur non ESDI)
4A	Alarme temps réel, appelée lors du traitement de l'interruption "matérielle" 70H.
50	Interruption d'alarme
51	Interruptions souris

5C	Point d'entrée NETBIOS
60 à 66	Réservées aux programmes d'interruption utilisateur
67	Point d'entrée pour gestion mémoire LIM/EMS
70* (IRQ 8)	interruption de l'horloge temps réel (AT)
71 (IRQ 9)	(AT)
72* (IRQ 10)	renvoie sur IRQ2 (AT)
73* (IRQ 11)	(AT)
74* (IRQ 12)	(AT)
75 (IRQ 13)	coprocesseur (redirigé vers INT 2)
76* (IRQ 14)	disque dur (AT)
77* (IRQ 15)	(AT)
F1 à FF	réservées aux programmes d'interruption utilisateur

(*) indique un changement de la pile si la commande *stack* a été installée avec un nombre minimal de huit piles.

Les interruptions 8, 9 et 13 sont aussi utilisées pour le traitement de *trappes* lors de violations d'espace mémoire intervenues en mode *protégé*. Les interruptions utilisées par le système d'exploitation sont les suivantes :

Numéro (hexa)	Fonction
20	Arrêt programme
21	Appels au DOS
22	Adresse de fin du programme appelé
23	Adresse de fin sur <CTRL Break>
24	Gestion d'erreurs critiques
25	Lecture absolue disque
26	Ecriture absolue disque
27	Arrêt programme qui reste résident
28	*Idle Signal* (gestionnaire de l'état inoccupé*).
29	Sortie Télétype.
2A	Services MS-Net.
2B	*spool* d'impression
2E	Rechargement de la partie transitoire (utilisé par COMMAND.COM).
2F	Interruption *Multiplex*.
30	*Long Jump Interface* (adresse de MS-DOS)
31	*Long Jump Interface* (adresse de MS-DOS)

(*) MS-DOS appelle l'interruption 28H lorsqu'il attend une entrée utilisateur. Les programmes qui fonctionnent en arrière-plan peuvent installer leur propre gestionnaire d'inoccupation pour s'assurer qu'ils peuvent prendre la main.

I.2.1.3. Zone de communication BIOS

Deux zones de mémoire sont réservées à l'exécution des différents programmes constituant le BIOS et le DOS. On les appelle les zones de communication.

Adresse (hexa)	Contenu
400 à 407	Adresses de base pour les ports d'E/S COM1 à COM4
408 à 40D	Adresses de base pour les ports d'E/S LPT1 à LPT3
410 à 411	Mot d'équipement (voir INT 11H)
413 à 414	Taille mémoire en Ko (voir INT 12H)
417	Mot d'état clavier
	– *bit* 7 : touche insertion bloquée
	– *bit* 6 : touche majuscules bloquée
	– *bit* 5 : touches numériques bloquées
	– *bit* 4 : défilement bloqué
	– *bit* 3 : ALT pressée
	– *bit* 2 : CTRL pressée
	– *bit* 1 : touche majuscule gauche pressée
	– *bit* 0 : touche majuscule droite pressée
418	Mot d'état clavier
	– *bit* 7 : touche insertion appuyée
	– *bit* 6 : touche CAPS LOCK appuyée
	– *bit* 5 : touches numériques appuyées
	– *bit* 4 : touche défilement appuyée
	– *bit* 3 : touche PAUSE bloquée
	– *bit* 2 : touche PAUSE pressée
	– *bit* 1 : touche ALT gauche pressée
	– *bit* 0 : touche CTRL gauche pressée
41A à 41B	Pointeur sur tête de tampon clavier
41C à 41D	Pointeur sur queue de tampon clavier
41E à 43D	Tampon clavier
43E à 448	Zone de données disquettes
	– drapeau *recalibrate* lecteur de disquette
	– état moteur du lecteur de disquette
	– compteur d'arrêt moteur du lecteur de disquette
	– état lecteur
	– état contrôleurs de lecteur de disquette (7 octets)
449 à 466	Zone affichage vidéo (palette, mode courant, position curseur,page active, …)
449	– mode d'affichage courant
44A à 44B	– nombre de colonnes de caractères affichables
44C à 44D	– taille du *buffer* vidéo
44E à 44F	– *offset* dans le tampon vidéo de la page courante
450 à 45F	– positions curseur (8 mots pour les huit pages)
460	– ligne début du curseur
461	– ligne fin du curseur
462	– numéro de page active
463 à 464	– adresse du port du contrôleur vidéo (*CRTC* : 3B4/3D4)
465	– valeur courante du registre *CRTC Mode Control*
466	– valeur courante du registre de palette CGA

46C à 46F	Données compteur (*timer*)
470	Drapeau de débordement compteur (*timer*)
471	Etat touche BREAK
472 à 473	Drapeau de RAZ
474 à 477	Zone de données disque dur
474	– état disque dur
475	– nombre de disques durs
476	– contrôle disque dur XT
477	– adresse du port du contrôleur de disque dur
478 à 47A	LPT1 à LPT3 valeur du *time out*
47C à 47F	COM1 à COM4 valeur du *time out*
480 à 481	Pointeur sur début tampon clavier (normalement 1EH)
482 à 483	Pointeur sur fin tampon clavier (normalement 3DH)
484 à 48A	Zone de donnée vidéo (suite)
484	– nombre de lignes – 1
485 à 486	– taille de la matrice de caractère
487 et 488	– état contrôleur vidéo
48B à 495	Zone de données disque dur
48B	– *step rate*
48C	– état contrôleur
48D	– état d'erreur du contrôleur disque dur
48E	– Contrôle des interruptions disque dur
490 à 495	Zone de données disquettes
490 à 491	– état des disquettes, lecteur 0 et 1
494 à 495	– numéro de cylindre courant, lecteur 0 et 1
496	Etat clavier
497	Drapeaux LED clavier (*PC/AT*)
498 à 49B	Adresse du drapeau d'attente utilisateur
49C à 49F	Compteur d'attente utilisateur
4A0	Drapeau d'attente active – bit 7 : le temps est écoulé – bit 0 : la fonction (INT15H, fonction 86H) a été lancée
4A8 à 4A9	Pointeur sur les paramètres vidéo
4AA à 4AB	0C000H (BIOS EGA)
4AC à 4EF	Réservés
4B0	– pointeur vers table de caractères EGA alphanumériques
4B4	– pointeur vers table de caractères EGA graphiques
4F0	Zone de communication inter-applications

I.2.1.4. Zone de communication DOS

Adresse (hexa)	Contenu
500	Indicateur état d'affichage écran
504	Etat mode (système à une seule unité de disquette)
510 à 521	Utilisé par le BASIC
522 à 52F	Utilisé par le DOS pour l'initialisation disque
530 à 533	Utilisé par la commande MODE
560	Utilisé pour gestion INT 17H (imprimante parallèle)
580	Utilisé pour gestion INT 14H (communication asynchrone)

I.2.2. L'espace des entrées/sorties

On trouve dans cet espace les registres des différents contrôleurs de périphériques. Les adresses de 0 à FFH sont réservées. Les adresses de 100H à 3FFH sont utilisables pour les canaux d'entrée/sortie.

Cette limite de 3FFH n'est pas liée au processeur puisque l'espace des entrées-sorties est de 64K sur les processeurs I8086. Elle est simplement dûe au fait qu'IBM n'a pas jugé bon, sur les premiers PC, de décoder plus de 10 lignes d'adresses.

Une connaissance au moins approximative de l'espace des entrées-sorties est nécessaire dès lors que l'on désire installer des cartes d'extension sur un PC. En principe, ces cartes prévoient la modification de leur adresse par positionnement de cavaliers pour éviter tout conflit avec d'autres cartes d'extension.

Adresse (hexa) *PC*	Adresse (hexa) *AT*	Contrôleur
000-00F	000-01F	Contrôleur DMA
020-021	020-03F	Contrôleur d'interruptions (I8259)
040-043	040-05F	Horloge (Compteur 0 du I8253)
060-063		Interface programmable PPI 8255
	060-06F	Clavier (UPI-8742) : les registres d'adresse 60H et 64H sont utilisés pour contrôler le clavier. Le bit 0 du port de sortie de l'UPI est en particulier utilisé pour provoquer un *shutdown* lors du passage du mode *protégé* au mode *réel* (voir octet 0FH de la mémoire CMOS). Le port 61H est utilisé pour le contrôle de parité. Le 8742 est aussi utilisé pour le contrôle de la ligne d'adresse A20 pour l'accès à la mémoire HMA. Le bit 1 du port de sortie est utilisé à cet effet.
	070-071	Horloge CMOS (MC146818) – 00 à 0D heure /alarme (codés en BCD) – 0A bit 7 mise à jour en cours – 0E état contrôleur – 0F octet d'état de retour en mode non *protégé* – 10 type de lecteur de disquettes

		– 11 type de lecteur de disquettes – 12 type de lecteur de disque dur – 14 octet d'équipement – 15 à 18 taille mémoire au-delà de 1 Mo (progr. de *setup*) – 19 et 1A autres types de disques durs – 2E à 2F *checksum* des octets 10 à 2D – 30 à 31 taille mémoire au-delà de 1 Mo (progr. de *reset*) – 32 siècle (BCD) – 33 drapeau d'état à la mise en route
080-083	080-09F	Registres de page DMA
	0A0-0BF	Deuxième contrôleur d'interruptions (I8259)
	0C0-0DF	Deuxième contrôleur DMA (I8237)
	0F0-0F1	Coprocesseur mathématique
	0F8-0FF	Coprocesseur mathématique
	1F0-1F8	Contrôleur disque dur
200-20F	200-207	Manette de jeu
	208 à 20B	Mémoire expansée (ou 218, 258, 268)
210-217		Unité d'extension
220-24F		Réservé
	278-27F	Deuxième imprimante parallèle
2F0-2F7		Réservé
2F8-2FF	2F8-2FF	Communications asynchrones (secondaire)
300-31F	300-31F	Carte de mise au point
320-32F		Disque dur
	360-36F	Carte réseau
378-37F	378-37F	Première imprimante parallèle
380-38C		Communications SDLC
380-389	380-38F	Communications BSC (secondaire)
3A0-3A9	3A0-3AF	Communications BSC (primaire)
3BO-3BF	3BO-3BF	Affichage monochrome IBM + imprimante
3C0-3CF		Réservé
3D0-3DF	3D0-3DF	Affichage graphique/couleur
3E0-3E7		Réservé
3F0-3F7	3F0-3F7	Contrôleur de disquette
3F8-3FF	3F8-3FF	Communications asynchrones (primaire)

Remarque : les PC/AT possèdent un composant appelé RAM/CMOS qui permet de stocker des informations sur la configuration matérielle de la machine. Cette mémoire est maintenue à l'aide d'une pile ou d'une batterie et son contenu peut être modifié à l'aide d'un programme de *setup*. Pour accéder à la RAM CMOS on écrit (`OUT` …) en 70H l'adresse à laquelle on veut accéder, puis on va lire en 71H (`IN` …) l'information recherchée. Lorsque l'adresse fournie est supérieure à 07FH, l'interruption non masquable NMI est inhibée. Ceci est utilisé notamment lorsqu'on passe en mode *protégé*.

Adresse	Horloge (MC146818)
	– Heure (codée en BCD)
0	– secondes
2	– minutes
4	– heure
6	– jour de la semaine
7	– jour du mois
8	– mois
9	– 9 année
	– Alarme (codée en BCD)
1	– secondes
3	– minutes
5	– heure
A–D	– Registres d'état de la CMOS. Ces octets sont utilisés pour le contrôle de l'horloge temps réel (INT 1AH). Bit 7 de 0AH : mise à jour en cours de l'horloge temps réel.
E	– Etat contrôleur : – bit 2 heure incorrecte – bit 3 type de disque dur incorrect – bit 4 taille mémoire incorrecte – bit 5 octet de configuration incorrect – bit 6 *checksum* incorrect – bit 7 problème de pile ou batterie
F	– Octet d'état de retour en mode *non protégé* : lors d'un *shutdown* cet octet est testé et une action entreprise en fonction de sa valeur. Lors d'un passage du mode *protégé* au mode *réel* cette valeur est 9.
10	– Type de lecteur de disquettes (bits 4 à 7 pour A, bits 0 à 3 pour B) : – 0 pas de lecteur – 1 lecteur 360 Ko – 2 lecteur 1,2 Mo – 3 lecteur 720 Ko – 4 lecteur 1,44 Mo
11	– Type de lecteur de disquettes (AT) (informations identiques, mais pour lecteurs de disquettes installés en C et D).
12	– Type de lecteur de disque dur (bits 4 à 7 pour premier disque dur, bits 0 à 3 pour le second) : – 0 pas de lecteur – 1 à 14 : type 1 à 14 – 15 le type est codé en 19 ou 1A
14	– Octet d'équipement – bits 7-6 00 1 lecteur de disquettes 01 2 lecteurs de disquettes 10 3 lecteurs de disquettes (AT) 11 4 lecteurs de disquettes (AT)

	– bits 5-4 00 EGA/VGA 01 CGA 40 x 25 10 CGA 80 x 25 11 MDA – bit 1 : indique la présence du coprocesseur 80287 – bit 0 : indique la présence de lecteur de disquettes
15 à 16	– Taille mémoire en-deça de 1 Mo (progr. de *setup*)
17 à 18	– Taille mémoire au-delà de 1 Mo (progr. de *setup*)
19	– Autre type de disque dur (premier disque dur)
1A	– Autre type de disque dur (deuxième disque dur)
2E à 2F	– *Checksum* des octets 10 à 2D
30 à 31	– Taille mémoire au-delà de 1 Mo (progr. de *reset*)
32	– Siècle (BCD)
33	– Drapeau d'état à la mise en route

Les types de disque dur sont identifiés par un numéro qui caractérise :

– le nombre de cylindres,

– le nombre de têtes,

– le numéro de cylindre pour pré-compensation (modification de l'intensité du courant d'écriture en fonction du numéro de cylindre),

– le nombre de secteurs par piste,

– le numéro du cylindre réservé à l'atterrissage des têtes.

Ces informations sont obtenues à l'aide de la fonction 32H du DOS à partir de la version 5.

I.2.3. Expansions et extensions mémoire

MSDOS n'est capable de gérer que les 640 premiers Ko de mémoire, désignés comme *mémoire conventionnelle*. Un certain nombre d'artifices permettent aux applications d'accéder à de la mémoire supplémentaire. On trouve essentiellement trois concepts :

I.2.3.1. Mémoire étendue

La mémoire *étendue* (*Extended Memory* ou *XMS*) est la mémoire d'adresse supérieure à 1 Mo. Cette mémoire n'est accessible qu'aux processeurs 80286, 386 ou 486 en mode *protégé*. Le premier segment de 64 Ko situé à partir de l'adresse 10000_{16} est appelée HMA (*High Memory Area*). Celui-ci peut être géré en mode réel (*driver* HIMEM.SYS) pour y ranger les programmes résidents (*TSR*) et autres *device drivers*. Cette zone de mémoire n'est pas utilisable sous DOS pour y l'exécution des programmes mais seulement sous des systèmes tels que Windows ou OS/2.

I.2.3.2. Mémoire expansée

La mémoire *expansée* (*Expanded Memory*) est une mémoire paginée vue du processeur par une "fenêtre" de 16 Koctets :

– la norme LIM-EMS 3.2 permet d'accéder à 8 Mo de mémoire expansée, vue à travers 4 pages de 16K, situées entre 640 Ko et 1 Mo.

– les normes EEMS et LIM-EMS 4.0 (1987) augmentent cette capacité à 32 Mo et permet d'installer la fenêtre n'importe où entre 0 et 1 Mo.

Lorsqu'on acquiert une carte d'expansion mémoire, il faut être sûr de la version de *driver* fourni avec la carte car tous les logiciels ne savent pas prendre en compte de la même façon la mémoire expansée.

I.2.3.3. Les UMB

Les UMB (*Upper Memory Blocks*) sont situés dans la zone de *mémoire haute* d'adresses A000:0 à FFFE:F. Ceux-ci sont accessibles à la condition que les gestionnaires HIMEM.SYS et EMM386.SYS soient chargés et que la commande DOS=UMB existe dans le fichier CONFIG.SYS. Un programme chargé avec la commande LOADHIGH est installé en mémoire haute. Des fonctions sont maintenant disponibles dans le DOS (fonction 5802H et 5803H à partir de la version 5) pour établir un *lien* avec cette mémoire haute.

Les machines équipées de mémoire étendue (ce qui est maintenant courant) peuvent utiliser cette mémoire comme mémoire expansée à la condition de disposer d'une machine à base de 386 et d'un *driver* d'émulation de mémoire expansée (du type CEMM.EXE, QEMM.SYS de Quaterdeck, 386 Max de Qualitas ou EMM386 donné avec MS-DOS).

I.2.4. Les canaux DMA

Le *PC* peut utiliser pour ses échanges avec les périphériques le *mode d'accès direct à la mémoire*. Les *PC* de base disposent d'un seul contrôleur, tandis que les *AT* en disposent de deux mis en cascade selon le schéma de la figure 10.

Chaque contrôleur gère quatre canaux d'échange : cela signifie qu'il dispose de quatre ensembles *registre d'adresse+compteur*. Sur l'*AT*, les canaux sont numérotés de 0 à 7 :

– canal 0 pour le rafraîchissement mémoire,
– canal 1 utilisable pour les cartes d'extension,
– canal 2 réservé au contrôleur de disques souples,
– canal 3 réservé pour le disque dur,
– canal 4 réservé pour la mise en cascade des deux contrôleurs,
– canaux 5 à 7 disponibles.

Les adresses engendrées par les contrôleurs sont sur 16 bits. Cela explique la présence de deux jeux de 4 registres pour disposer du complément nécessaire à l'adressage de 20 lignes (*PC*) ou 24 lignes (*AT*) : ce sont les *registres de page*.

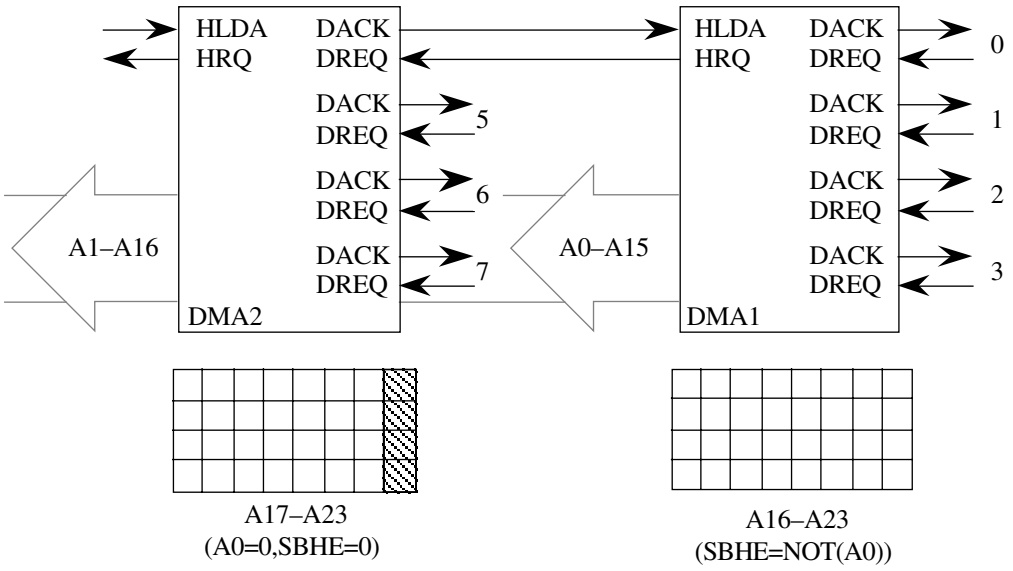


Fig.11 – *Canaux DMA*

Pour le second contrôleur DMA, les adresses des registres doivent être multipliés par 2 !

I.2.5. Exemples

– Utilisation du canal 1 pour effectuer des transferts d'octets :

```
; Configuration du contrôleur DMA
; Les instructions JMP $+2 sont nécessaires sur les AT pour laisser
; le temps au contrôleur de prendre en compte les commandes et
; paramètres transmis.
;
; On commence par inhiber les interruptions
     cli                    ; On inhibe les interruptions
     mov     AL,05h         ; on inhibe le canal 1
                            ;(bits 0-1=code canal, bit 2=1)
     out     0Ah,AL         ;
; transfert en écriture, mode "demande", incrément compteur, canal 1
     mov     AL,05h         ;
     out     0Bh,AL         ;
; ordre du transfert (poids faible d'abord) des adresses et compteurs
     xor     AL,AL          ;
     jmp     $+2            ; delai
     out     0Ch,AL         ; mise à zéro flip flop
; traitement de l'adresse de transfert (on la suppose ici dans DI).
; Cette adresse nécessite un traitement pour ne pas risquer un
; dépassement dans le segment
     mov     AX,DI          ;
     jmp     $+2            ; delai
     out     02h,AL         ;
```

```
     mov     AL,AH           ;
     jmp     $+2             ; delai
     out     02h,AL          ;

     mov     AX,ES           ; segment
     mov     AL,AH           ;
     shr     AL,1            ;
     shr     AL,1            ;
     shr     AL,1            ;
     shr     AL,1            ;
     jmp     $+2             ; delai
     out     083h,AL         ; registre de page correspondant au canal 1
                             ; (87H canal 0, 81H canal 2, 82H canal 3)
                             ; (8BH canal 5, 89H canal 6, 8AH canal 7)
                             ; (8FH rafraîchissement)

; traitement du nombre d'octets à transférer (on le suppose dans CX)
     mov     AX,CX           ;
     jmp     $+2             ; delai
     out     03h,AL          ;
     mov     AL,AH           ;
     jmp     $+2             ; delai
     out     03h,AL          ;
; On revalide les DREQ sur le canal 1
     mov     AL,1            ; on valide le canal 1
     jmp     $+2             ; delai
     out     0Ah,AL          ;
; on revalide les interruptions
     sti                     ;
;
; on démarre le contrôleur de périphérique
     …
; attente traitement et fin de DMA
; on teste généralement l'état du contrôleur de périphérique.
; Lorsque le DMA travaille en mode bloqué, le contrôle
; est perdu par le processeur dès qu'un DREQ est reçu par le
; contrôleur de DMA.
boucle:
     in      al,dx           ; lecture contrôleur de périph.
     …
     jne     boucle          ; test
;
; le transfert est terminé. On invalide le canal 1.
     mov     al,05H
     out     0Ah,alL
```

CHAPITRE II

Introduction à MSDOS

MSDOS est né au mois d'Août 1981 avec les premiers micro-ordinateurs PC d'IBM. Ces derniers étaient dotés d'un lecteur de disquettes simple face d'une capacité de 160 kilo-octets, d'une mémoire centrale de 128 kilo-octets de vive, de 64 Ko pour le BIOS et d'un interpréteur BASIC lui aussi en mémoire morte. Les versions successives du système ont apporté le support de nouveaux périphériques et des fonctionnalités étendues pour le réseau, la mémoire paginée ou étendue, etc. Ces améliorations apparaissent clairement à travers les utilitaires fournis avec le système mais aussi moins visiblement dans les fonctions internes soutenues par celui-ci. Ces versions sont(1) :

05/82	1.1	Support des lecteurs FD 5"1/4 DF 320 Ko
03/83	2.0	Support des lecteurs FD 5"1/4 DF 360 Ko Introduction des répertoires et sous-répertoires
08/83	3.0	Support des lecteurs FD 5"1/4 DF HD 1,2 Mo Jeu de caractères français et allemand
10/83	2.1	Introduction de PC-DOS
03/85	3.1	Support de LAN-Manager et PC LAN
12/85	3.2	Support des lecteurs FD 3"1/2 SF 720 Ko
04/87	3.3	Introduction de la notion de page de codes Gestion de plusieurs partitions
11/88	4.01	Support de partitions de taille > 32 Mo et < 4 Go Dosshell
06/91	5.0	Gestion de la HMA, des UMB et de la mémoire étendue Commutation de tâches
03/93	6.0	Optionnalisation des configurations de CONFIG.SYS Outils de compression, antivirus, etc. Intégration MSDOS et Windows

MSDOS est un système d'exploitation relativement rudimentaire dans lequel on trouve des fonctions d'entrée-sortie sur console, de gestion de l'espace mémoire et des disques. La plupart de ces fonctions sont accessibles au programmeur à partir de l'interruption 21H. Certaines d'entre elles

(1)l'Edition de Liens ‘MSDOS en six tableaux’ Microsoft, 06/07/08 1993

ne sont cependant pas documentées car susceptibles de modification de la part de Microsoft lorsqu'il y a changement de version de système d'exploitation.

II.1. La phase de démarrage

A la mise sous tension de la machine, le registre CS est initialisé avec la valeur F000H et le compteur ordinal avec la valeur FFF0H. A cette adresse on trouve un branchement vers un programme résident en ROM qui effectue un certain nombre de tests et d'initialisations. Ce programme effectue ensuite un appel INT 19H (cet appel est aussi activé lorsqu'on appuie simultanément sur les touches <CTRL> <ALT> et <DEL>) qui exécute un programme dit de *pré-amorce*. Ce dernier :

– initialise un pointeur vers une table donnant les caractéristiques des lecteurs de disque ou disquettes. Ces paramètres sont utilisés par le contrôleur des disques. Ce pointeur se trouve dans la table des vecteurs d'interruption (INT 1EH pour les disquettes et INT 41H pour les disques durs),

– charge en mémoire le secteur 1 de la piste 0 du cylindre 0 (le premier secteur physique du disque) contenant le secteur de partitionnement (voir chap. III) ou le secteur d'amorce,

– dans ce dernier cas, cette copie est faite à l'adresse 0:7C00.

– effectue un branchement vers le programme d'amorce.

Le programme amorce charge les fichiers IO.SYS et MSDOS.SYS. IO.SYS est chargé en 0060:0000. Il contient les *drivers* des périphériques de base de la machine : clavier, écran et lecteur de disquettes. Le programme amorce lui donne la main :

– il effectue un certain nombre d'initialisations (test de configuration, initialisation des unités d'échange, définition de vecteurs d'interruption, et branchement vers MSDOS.SYS),

– il installe MSDOS.SYS à sa place définitive en mémoire,

– et transfère ensuite le contrôle à MSDOS.SYS.

MSDOS.SYS commence par initialiser ses zones de travail, réserve de la mémoire pour chaque table d'allocation des fichiers, le répertoire et divers tampons, et initialise les vecteurs d'interruption 20H à 27H. Il recherche ensuite le fichier CONFIG.SYS lui permettant d'achever sa configuration et d'introduire éventuellement de nouveau *drivers* de périphériques. Il construit un préfixe de segment programme (PSP) pour l'interpréteur de commande COMMAND.COM.

En dernier lieu, l'interpréteur de commande COMMAND.COM est chargé.

II.2. Le processeur de commande COMMAND.COM

COMMAND.COM est composé :

– d'une partie *résidente* chargée de traiter les événements liés à la fin d'exécution -normale, sur incident ou forcée- des programmes (fin forcée par <CONTROL C> ou <CONTROL BREAK> par exemple). Elle contient une routine de chargement de la partie *transitoire* et un programme qui se charge de traiter le fichier AUTOEXEC.BAT. Ce dernier est écrasé par le premier programme chargé.

– et d'une partie *transitoire* chargée d'envoyer sur l'écran le message d'interrogation (*prompt*) et de traiter les commandes introduites au clavier ou dans les fichiers de type *batch*. Elle contient le code des commandes résidentes (DIR, COPY, …). Cette partie est chargée en mémoire haute.

Lorsqu'une commande externe est introduite au clavier, le *processeur de commande* effectue les opérations suivantes :

– détermination de l'emplacement du *Segment Programme* (SPr) (voir gestion de l'espace mémoire),

– construction à l'adresse 0 du *segment programme* d'un *Préfixe de Segment Programme* (*PSP*) de 256 octets,

– chargement du programme à l'adresse 100H (ou en mémoire haute pour les .EXE si c'est spécifié) du SPr.

Le préfixe de segment de programme possède la structure suivante (les informations non documentées sont notées avec (*) :

$Adresse_{16}$	Contenu
0 à 1	INT 20H
2 à 3	Taille mémoire totale en paragraphes (adresse de segment du premier octet non disponible). Il ne faut rien modifier au-delà à moins d'avoir appelé au préalable la fonction 48H du DOS.
5 à 9	FAR CALL de INT 21H.
A à D	Adresse de fin d'exécution (offset:segment).
E à 11	Adresse de sortie sur <CTRL C> (offset:segment).
12 à 15	Adresse de sortie sur erreur matérielle (offset:segment).
16 à 17(*)	Adresse de PSP du processus père.
18 à 2B(*)	*Job File Table* par défaut du programme.
2C à 2D	Adresse de segment de l'environnement (au moins COMSPEC = … pour pouvoir retrouver COMMAND.COM sur disque, ainsi que les PATH, SET, etc …).
32 à 33(*)	Nombre maximum de fichiers ouverts (taille de la *JFT*)
34 à 37(*)	Adresse de la *JFT* active.
50	INT 21H
5C	FCB non ouvert.
6C	FCB non ouvert (écrasé si le FCB en 5C est ouvert).
80	Nombre de caractères (début DTA par défaut).
81	Caractères (les redirections n'apparaissent pas ici).
FF	Fin PSP.

Il faut noter que lorsque le programme prend la main :

– l'adresse de transfert disque (*DTA* pour *Disk Transfert Address*) est initialisée par défaut à 80H,

– des *FCB* (*File Control Block*), constitués de 37 ou 44 octets, associés aux deux premiers paramètres de la ligne de commande sont formatés en 5CH et 6CH (on notera les risques d'écrasement …),

– en 80H est écrit le nombre de caractères suivant le nom de la commande dans la ligne de commande,

– les caractères suivant le nom de la commande sont écrits à partir de 81H,

– le mot à l'adresse 6 donne le nombre d'octets disponibles dans le segment,

– AH et AL contiennent 0 ou FF selon que la désignation d'unité de disque, dans les deux premiers paramètres, est correcte ou non.

Pour les programmes d'extension .EXE (*programmes chargeables*), DS et ES pointent sur le PSP. Les registres CS, CO, SS et SP contiennent les valeurs définies par l'éditeur de liens.

Pour les programmes de type .COM :

– ES, DS, CS, SS contiennent l'adresse du bloc de contrôle du PSP,

– le compteur ordinal vaut 100H,

– SP contient l'adresse de fin du SPr (ou l'adresse basse de la partie transitoire de COMMAND.COM et la taille indiquée à l'adresse 6 dans le PSP est diminuée de 100H pour laisser 100H octets pour la pile,

– un mot à 0 est placé en sommet de pile (ceci permet au programme de retourner à COMMAND.COM en exécutant un RET).

Après exécution du programme, le contrôle est rendu à la partie résidente de COMMAND.COM (fonction 4CH de INT 21) qui effectue les actions suivantes :

– la restauration des vecteurs correspondant à INT 22H, INT 23H et INT 24H, à partir des valeurs sauvegardées dans le PSP du programme qui se termine,

– le transfert du contrôle à la partie transitoire de COMMAND.COM (un contrôle de type *checksum* est effectué pour savoir si la partie transitoire doit être rechargée).

II.3. Quelques compléments

II.3.1. Les exécutables .EXE

Les programmes exécutables engendrés par l'éditeur de liens sont enregistrés sur disque avec un en-tête contenant les informations utiles au chargement du programme en mémoire.

Déplacement (hexa)	Signification
	Partie formatée de l'en-tête
0 à 1	En-tête mis par l'éditeur de liens (5A4DH = "ZM").
2 à 3	Nombre d'octets dans le dernier secteur du fichier.
4 à 5	Taille du fichier en nombre de secteurs (512 octets).
6 à 7	Nombre d'entrées dans la table de translation.
8 à 9	Taille de l'en-tête en paragraphes (1paragraphe=16 octets).
A à B	Nombre minimum de paragraphes nécessaires au delà de la fin du programme une fois chargé.
C à D	Nombre minimum de paragraphes nécessaires au delà de la fin du programme une fois chargé. Si FFFFH, le programme est chargé en partie haute de mémoire.
E à F	Adresse relative du segment de pile dans le module exécutable sous forme de
10 à 11	segment.
12 à 13	Valeur initiale du pointeur de pile au démarrage du programme.
	Mot de contrôle (par somme sur 16 *bits*).

14 à 15	Valeur du compteur ordinal au démarrage du programme.
16 à 17	Adresse relative du segment de code dans le module exécutable sous forme de segment.
18 à 19	Adresse relative de la première entrée translatable dans le fichier.
1A à 1B	Niveau de recouvrement (0 pour la racine) (*overlay*).
	Partie non formatée de l'en-tête
	Table de translation. Chaque entrée est sous la forme *segment:déplacement* et donne l'adresse relative dans le module exécutable du mot à modifier.
	Module exécutable

– L'en-tête formatée précédente est chargée en mémoire,

– une zone mémoire est allouée en fonction des informations contenues en A-B et C-D (MS/DOS tente d'affecter FFFFH paragraphes : comme ce n'est pas possible, cette tentative permet de récupérer la taille de l'espace disponibles),

– Un PSP est créé,

– Le calcul de la taille du module exécutable est effectué (taille du fichier - taille de l'en-tête) en utilisant les informations situées en 4-5 et 8-9,

– Le module exécutable est amené en mémoire (haute ou basse) dans le segment dit "de départ",

– Les éléments de la table de translation sont amenés en mémoire (zone de travail),

– La valeur "segment" de chaque entrée est modifiée de la valeur du segment de "départ",

– Après mise à jour des informations dans le module exécutable, les registres SS et SP sont initialisés à partir des informations contenues dans l'en-tête et de la valeur du segment de départ. ES et DS contiennent l'adresse de segment du PSP,

– CS est calculé à partir du contenu de l'en-tête et de la valeur du segment de départ.

II.3.2. Les blocs de contrôle de fichiers (FCB)

Le bloc de contrôle de fichier constitue le moyen privilégié sous les versions 1 et 2 du DOS pour accéder aux fichiers. C'est une structure de données de 37 ou 44 octets selon qu'il s'agit d'un FCB *normal* ou *étendu*. Le bloc de contrôle étendu est utilisé pour créer ou rechercher des fichiers dotés d'attributs particuliers. Il est constitué en ajoutant au bloc de contrôle normal 7 octets précédant celui-ci.

Si l'on veut accéder aux informations d'un fichier à l'aide de "fonctions FCB", il faut tout d'abord écrire le nom du fichier dans le FCB (on n'accède qu'aux fichiers du répertoire courant) : on peut le faire directement ou à l'aide de la fonction 29H du DOS. Le fichier peut ensuite être ouvert. Cette fonction d'ouverture met à jour le FCB et fixe par défaut une longueur d'enregistrement à 128 octets, et une zone de transfert disque dans le PSP à l'*offset* 80H. L'emplacement de la zone de transfert peut être modifiée à l'aide de la fonction 1AH du DOS, mais seulement après ouverture du fichier.

Déplacement (décimal)	Signification
–7	Drapeau de FCB étendu (FFH désigne un FCB étendu).
–6 à –2	Réservé.
–1	Attribut.
0	Lecteur : 0 pour A, 1 pour B, etc … à 0 pour spécifier le lecteur par défaut.
1 à 8	Nom de fichier complété éventuellement avec des "espaces". Pour un nom réservé on ne met pas le ":".
9 à 11	Extension complétée éventuellement avec des "espaces".
12 à 13	Bloc courant. Un bloc = 128 enregistrements. bloc courant + enregistrement courant = pointeur d'enregistrement
14 à 15	Taille d'enregistrement. Par défaut égal à 128 (au moment de l'ouverture).
16 à 19	Taille du fichier (en octets) (poids faible en tête).

20 à 21	Date de la dernière modification
	`15 : A A A A A A A M`
	`14 : M M M J J J J J`

22 à 23	Heure de la dernière modification
	`17 : H H H H H M M M`
	`16 : M M M S S S S S`

24 à 31	Réservé
32	Enregistrement courant. Pointeur vers un des 128 enregistrements du bloc courant (non initialisé par la fonction d'ouverture).
33 à 36	Enregistrement relatif. Pointeur vers l'enregistrement courant (le premier a pour numéro 0). Non initialisé par la fonction d'ouverture. Si la taille de l'enregistrement est < 64 , les quatre octets sont utilisés. Dans le cas contraire (≥64) seuls les trois premiers le sont.

II.3.3. Gestion de l'espace mémoire

Le DOS utilise une liste chaînée pour gérer l'espace mémoire. Chaque élément de cette liste (ou MCB : *Memory Control Block*) est constitué de 16 octets (*arena structure*) implantés en limite de paragraphe (adresses XXXX:0) et suivi de la zone mémoire allouée.

– le premier octet est un 'M' ou un 'Z'. Le 'M' signifie qu'il y a un élément suivant, tandis que le 'Z' indique la dernière zone de données allouée.

On note par des minuscules les octets de poids faible.

– deux octets (*aaAA*) donnent l'adresse, en nombre de paragraphes, du code de l'exécutable correspondant (en fait l'adresse du *préfixe de segment programme*) :

– en principe cette adresse est celle du paragraphe qui suit le MCB (*XXxx*+1),
– si ce n'est pas le cas, les paragraphes qui suivent contiennent l'environnement du programme.

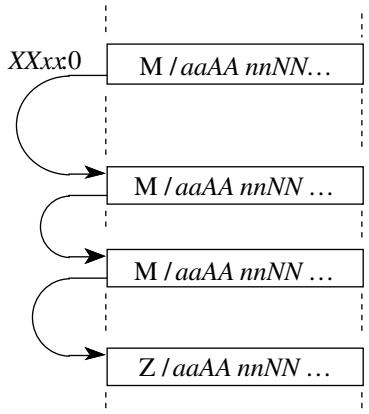


Fig.1 – *Liste chaînée des MCB*

Lors de la libération d'un espace mémoire (désinstallation d'un programme résident par exemple) il suffit de remettre ce mot à zéro pour que l'espace soit effectivement pris en compte par MS–DOS (à la condition qu'il s'agisse du dernier résident installé !)

– deux octets (*nnNN*) donnent, en nombre de paragraphes, l'espace occupé par la zone allouée.

– On a ensuite 3 octets réservés puis 8 octets donnant le *nom* du propriétaire du segment.

Les adresses des MCB suivants sont obtenues à partir de l'adresse du MCB courant et du nombre de paragraphes occupés : *XXxx*+*NNnn*+1.

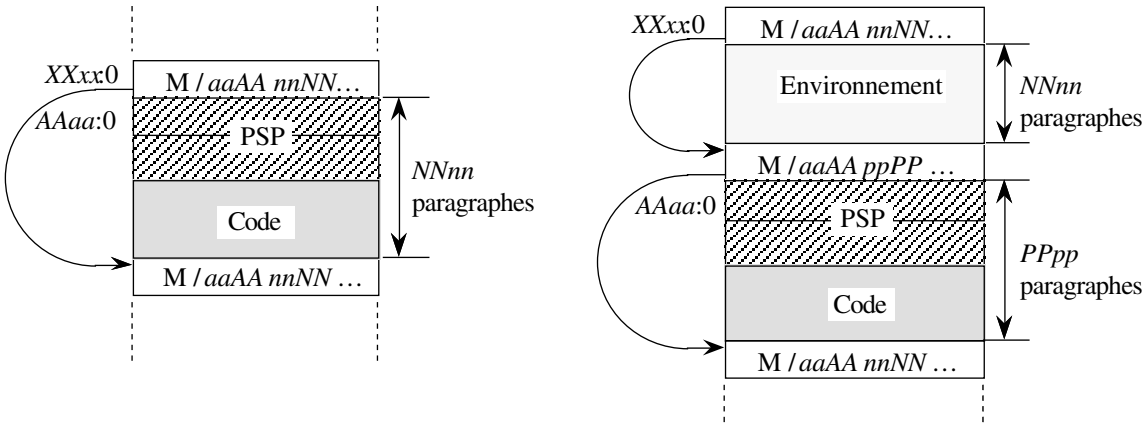


Fig.2 – *Structure des MCB*

Le problème est celui de l'obtention de l'adresse du premier MCB de la liste. Les versions 3 du DOS offrent une fonction 52H non documentée qui fournit dans ES:BX une adresse à laquelle il faut soustraire 4 pour obtenir l'adresse de l'adresse du premier MCB. Cette méthode est par essence peu recommandée puisque la fonction 52H est susceptible d'être modifiée par Microsoft.

Il est préférable de rechercher directement en mémoire ce premier MCB. En effet le premier MCB commence par les trois octets : 4D 08 00 (0008 est l'adresse de paragraphe de l'INT 20H). Il suffit donc de rechercher cette séquence (aux adresses de la forme XXXX:0) puis de vérifier que le

chaînage 'M', 'M', …, 'Z' est correct (la probabilité pour qu'une telle séquence existe et ne soit pas celle des MCB est quasi nulle).

II.3.4. Note sur l'utilisation de debug

On peut créer des utilitaires de petite taille en utilisant l'utilitaire DEBUG de la façon suivante :

```
>DEBUG              Appel à debug
-n b:util.com       Nomme l'utilitaire UTIL.COM sur disquette B (name)
-a cs:100           Début d'assemblage (à partir de l'offset 100H) (assemble)
-.........          code du programme
-.........
-.........
-mov ah,4c          Fin du programme (fonction de fin d'exécution)
-int 21
-
-r cx               BX:CX contient le nombre d'octets à écrire
-.........
-r bx
-.........
-w                  Ecriture de l'utilitaire sur disque (write)
XXXX octets écrits
-q                  Sortie de debug (quit)
>b:util            Lancement de l'utilitaire
```

II.3.5. La rédaction de programmes résidents en mémoire

Les programmes *résidents* ou *TSR* sont, avec les *pilotes de périphériques* ou *device drivers*, un moyen d'étendre les possibilités du micro-ordinateur. Leur principe repose sur le 'déroutement' d'une des interruptions pour installer son propre programme de traitement(2).

Considérons la situation initiale schématisée ci-après. Le pointeur de l'interruption *i* donne l'adresse du programme de traitement associé. La mémoire totale occupée au départ est indiquée par la flèche double :

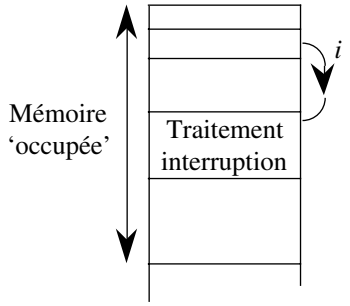


Fig.3 – *Situation initiale*

(2)Microsoft Corp. "Mcrosoft Macro Assembler Programmer's Guide Version 6.00"

Après installation du programe résident ‘accroché’ à cette interruption la situation peut être schématisée ainsi :

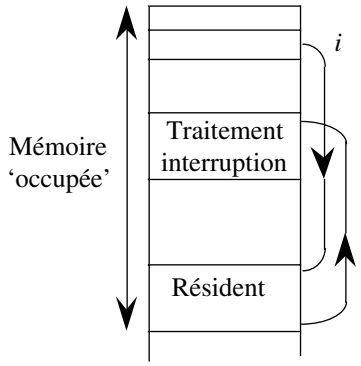


Fig.4 – *Situation après installation du résident*

Le traitement du *TSR* peut s'effectuer avant ou après l'interruption pré-existante :

– avant en effectuant un branchement (instruction `JMP`) vers le *TSR*,

– en faisant appel au programme d'interruption à l'aide d'un `CALL`. Dans ce cas il est indispensable de conserver à la pile un état correct car le programme d'interruption se termine par une instruction `IRET` et non `RET`. Le `CALL` doit donc être suivi d'un `pop`.

II.3.5.1. Installation

D'une certaine manière, installation et traitement sont un seul et même programme. Le programme d'installation constitue en effet le ‘*container*’ du programme de traitement. Une des différences avec un programme normal est qu'il se termine par un appel à une fonction ‘garde le programme en mémoire’. Sa structure est la suivante :

```
; vérifier si le TSR est déjà chargé.
; Dans l'affirmative on termine (fonction 4CH de INT 21H).
…
  JMP suite
; drapeau d'indication d'activité pour ce traitement
drapeau     byte      0
oldAdd      dword     ?
;*------ TSR proprement dit
traitement:
  cmp  c:drapeau,0
  jne traitOK
  jmp  cs:oldAdd
traitOK:
  inc  cs:drapeau
  …
  IRET ou JMP vers le programme d'interruption initial
;*-------------------------
suite:
; on lit l'ancienne adresse de traitement rangement en oldAdd
; (fonction 35H de INT 21H)
```

```
    int  21H
    mov  word ptr[oldAdd],bx
    mov  word ptr[oldAdd+2],es
…
; on accroche l'interruption i à l'adresse 'traitement'
; (fonction 25H de INT 21H)
…
; sortie et garde programme en mémoire
; (fonction 31H de INT 21H)
```

La désinstallation pose un certain nombre de problèmes. En effet MSDOS ne gère pas l'espace mémoire. Il est possible de désinstaller correctement le dernier *TSR* installé en remettant à 0 le mot 'AAaa' dans la structure *arena* décrite dans le chapitre suivant. Par contre, s'il n'est pas le dernier installé, MSDOS ne peut pas récupérer l'espace mémoire libéré.

II.3.5.2. Programme

Les programmes résidents étant 'accrochés à une interruption' ils posent le problème de la *réentrance* des programmes qu'ils interrompent et utilisent tout à la fois. C'est en particulier le cas des fonctions du DOS. Ces dernières définissent leur propre pile et ne doivent pas être interrompues pour de nouveau être appelées. Le *TSR* doit donc vérifier à chaque fois qu'il doit appeler une fonction du DOS que celle-ci n'est pas celle qui est interrompue.

DOS utilise essentiellement trois piles : une pile pour les entrées-sorties (fonctions 01 à 0CH), une pile pour les fonctions disque, et une pile auxiliaire.

Les règles à respecter sont les suivantes :

– si l'interruption 24H (gestion d'erreur critique disque) est en train de s'exécuter, aucune fonction du DOS ne peut être utilisée,

– si on interrompt le DOS, on ne peut utiliser que les fonctions 01 à OCH de INT 21H,

– si l'on n'interrompt pas le DOS, on peut utiliser n'importe quelle fonction du DOS.

Il faut donc savoir si l'on était ou non dans le DOS lorsqu'on entre dans le *TSR*. Cela peut être fait à l'aide de la fonction 34H de INT 21H. Cette fonction permet d'accéder :

– au drapeau `InDOS` à l'adresse `es:[bx]` où `es` et `bx` sont rtetournés cette fonction. Il s'agit d'un octet dont la valeur est différente de 0 si l'on est dans le DOS,

– au drapeau `ErrorMode` à l'adresse `es:[bx-1]` (DOS version≥3.1). Il s'agit d'un octet dont la valeur est différente de 0 si l'on est en train de traiter une erreur critique disque.

La seule utilisation du drapeau `InDOS` est suffisante. Cependant certaines fonctions d'entrée-sortie font de l'attente active et dans ce cas le *TSR* serait aussi obligé d'attendre. L'interruption 28H (*Idle Interrupt*) apporte une solution à ce problème. Lors de toute attente active, DOS appelle cette interruption de telle sorte que d'autres fonctions puissent effectuer leur traitement, en évitant toutefois d'altérer la pile des entrées-sorties. En contrepartie si un *TSR* doit à un moment ou à un autre faire de l'attente active, il doit, dans sa boucle d'attente, faire appel à cette interruption.

II.4. Le contrôle des unités d'entrée-sortie

MSDOS offre la possibilité de gérer la connexion à un PC d'éléments périphériques non standards. Le programme de gestion du contrôleur de ce périphérique est appelé un "pilote d'élément périphérique" ou *Device Driver* (*DD*).

Lorsqu'un *DD* a été défini, le moyen de l'utiliser est de passer par les fonctions "IOCTL" du DOS (fonction 44H).

Les éléments périphériques sont de deux types :

– type "caractère" : le transfert s'effectue "caractère" par "caractère" (clavier par exemple). Dans ce cas, on a deux modes de fonctionnement :

– mode *cooked* dans lequel le DOS examine s'il y a des caractères de contrôle dans le flot de données (<CTRL P> pour redirection sur imprimante, <CTRL S> pour suspension, etc …),

– mode *raw* dans lequel aucun test n'est effectué. Le nombre de caractères à transférer est transmis quoi qu'il arrive.

On peut *ouvrir* un tel type de périphérique, c'est à dire lui attacher un *handle* ou un *FCB*.

– type "bloc" : les informations sont transférées par blocs et on peut faire des entrées-sorties directes (*random I/O*). Un *DD* de ce type peut gérer plusieurs unités priphériques "bloc". Ces périphériques ne peuvent pas être *ouverts* directement. Ils peuvent comporter plusieurs *unités* (les disques *logiques* sur un disque *physique*).

Les pilotes d'éléments périphériques disposent d'un en-tête identifiant logiquement l'élément contrôlé. Ce fichier binaire ne possède pas de PSP (origine en 0). MSDOS, lors de son lancement, installe les *DD* standards tels que CON, PRN, AUX, NUL et ceux des lecteurs de disques et disquettes. Les *DD* rajoutés sont définis dans le fichier CONFIG.SYS par la directive DEVICE = ….

Les *DD* sont chaînés en mémoire de telle sorte que ceux qui sont nouvellement définis soient placés en tête de cette chaîne. Il est ainsi possible de remplacer les *DD* standards prédéfinis par un *DD* particulier. Le système d'exploitation va chercher les *DD* dans l'ordre et s'arrête sur le premier *DD* qui porte le nom cherché.

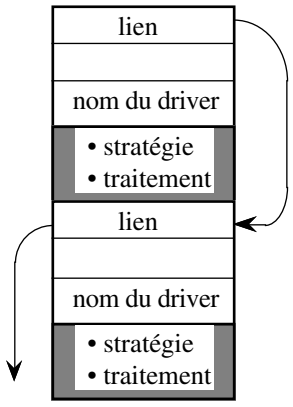


Fig.5 – *Liste chaînée des pilotes de périphériques*

Il faut cependant s'assurer, dans ce cas, que les bits 0 et 1 du mot d'attribut dans l'en-tête du *DD* sont tous deux à 1 (voir ci-après l'en-tête de *DD*). Les seules exceptions concernent le périphérique

désigné par NUL, qui est toujours en tête de liste, et les descripteurs de disquettes qui sont automatiquement nommés par le système (Il faut alors utiliser la commande ASSIGN pour rediriger les entrées-sorties sur le nouveau disque).

La routine de "stratégie" sert simplement à ranger l'adresse d'un bloc de paramètres, adresse fournie par le DOS dans ES:BX. Elle rend ensuite la main au DOS qui appelle la procédure de traitement proprement dite.

II.4.1. Structure des en-têtes

Chaque *DD* dispose d'un bloc d'en-tête dont la structure est la suivante :

Déplacement	Contenu
0 à 3	Lien (segment:déplacement) vers *DD* suivant. Au départ il vaut –1. Il est initialisé par MSDOS au moment du chargement du *DD*. S'il y a plus d'un *DD* dans le fichier, le premier mot de ce champ n'a pas la valeur –1 mais donne le déplacement du *DD* suivant. Le dernier *DD* a un lien égal à –1.
4-5	Mot d'attribut. Bit : – bit 0 = 1 si périphérique de sortie standard – bit 1 = 1 si périphérique d'entrée standard (A partir de la version DOS 4.0, si ce bit est à 1 pour un périphérique de type bloc alors le numéro de secteur est sur 32 bits) – bit 2 = 1 si périphérique de type "horloge" – bit 3 = 1 si NUL utilisé (le bit 15 doit être à 1) – bit 4 = 1 si type "spécial" – bits 5 à 10 = 0 – bit 11 indicateur de changement de support (version DOS≥3) – bit 12 à 0 – bit 13 indicateur de format IBM (type "bloc") utilisé pour les périphériques orientés bloc. Affecte l'appel de la fonction *BUILD BPB* – bit 14 bit IOCTL indiquant si le périphérique peut traiter des chaînes de commande (fonction 44H, INT 21H) – bit 15 = 1 si périphérique orienté caractère 0 si périphérique orienté bloc
6–7	Pointeur vers la routine "stratégie" (dans le segment utilisé pour pointer vers cet en-tête).
8–9	Pointeur vers routine d'interruption (dans le segment utilisé pour pointer vers cet en-tête).
10–17	Nom logique de l'élément périphérique (complété avec des "espaces") ou nombre d'unités pour un élément orienté bloc dans le premier octet.

Les *DD* de type "horloge" est appelé à partir des fonction 2AH et 2DH du DOS. Ils doivent mettre en œuvre les fonctions 0, 4 et 8.

II.4.2. L'installation et l'appel aux pilotes de périphériques

L'installation en mémoire des *DD* lus dans le fichier CONFIG.SYS est effectuée par le système. Une fois installé, MSDOS fait appel aux fonctions du *DD* en effectuant un appel long (`FAR CALL`) à la routine de stratégie en passant dans ES:BX l'adresse d'une structure contenant les informations relatives à la fonction demandée. Cette structure comporte une en-tête (*Request Header*) suivie des informations nécessaires à la fonction appelée. Il est de la responsabilité de la routine de traitement de préserver l'état de la machine.

Déplacement	Contenu
0	Longueur de cet en-tête.
1	Numéro d'unité adressée (périphérique "bloc").
2	Octet de commande.
3 à 4	Mot d'état.
5 à 0CH	Réservé.
0DH à …	Données supplémentaires (voir description des fonctions).

La fonction "stratégie" se contente de ranger l'adresse du bloc de paramètres constitué d'au moins 13 octets.

– un pilote d'élément périphérique possède la structure suivante :

```
     org    0                  ; il n'y a pas de PSP
; en-tête du DD
     dw     -1,-1              ; il n'y a qu'un seul DD dans cet exemple
     dw     10 ……              ; attribut (ici un DD orienté "caractère"
                               ; le bit 15 étant à 1))
     dw     offset strategie   ; pointeur vers la routine stratégie
     dw     offset traitement  ; pointeur vers la routine de traitement
     db     'PERIPH  '         ; nom du périphérique (complété avec des
                               ; "espaces"
     …
fonctions   dw offset init     ; table des adresses de traitement des
     dw     offset test        ; fonctions
     dw     fonction_n         ; …
;
BPB  dw (?),(?)                ; pour y mettre l'adresse du BPB transmise
                               ; dans ES:BX
strategie   proc far
     mov    cs:BPB,bx          ; on range l'adresse du BPB
     mov    cs:BPB+2,es        ;
     ret
strategie   endp
     …
traite      proc far
     push …
     …
```

```
     les    di,dword ptr BPB   ; on va relire l'adresse du BPB, puis
     mov    al,es:[di+2]       ; traiter la commande
     cbw                       ; …
     mov    si,offset[fonctions]
     add    si,ax
     add    si,ax

     jmp    word ptr[si]
fin:        …
     pop …
     ret
traite      endp
;
fonction_n:                    ; …
     …                         ; …
     …                         ; …
     or     ax,100H            ; on met le bit "terminé" à 1
     mov    es:[di+3],ax       ; et on le transmet
     jmp    fin                ; …
;
init                           proc near
…
init                           endp
test                           proc near
…
test                           endp
```

– Les commandes : le *DD* est activé à partir de commandes prédéfinies dans MS/DOS. Il faut évidemment respecter les codes et significations attachés à ces commandes lorsqu'on écrit un nouveau *DD*. Ces commandes sont les suivantes :

Code	Signification de la commande
0	Initialisation : utilisé par DOS au moment de l'installation du DD. Dans la plupart des cas cette fonction peut être écrasée. Elle se contente de définir l'adresse de fin du DD.
1	Test de support (périphérique "bloc").
2	Construction bloc de paramètres (périphérique "bloc").
3	Entrée IOCTL (appelable par la fonction 44H du DOS.
4	Lecture.
5	Lecture non destructive sans attente (périph. "caract.").
6	Lecture non destructive sans attente (périph. "caract.").
7	Lecture état (périph. "caract.").
8	Entrée *flush* (périph. "caract.").
9	Ecriture.
10	Ecriture avec vérification.
11	Sortie état (périph. "caract.").
12	Sortie *flush* (périph. "caract."). Sortie IOCTL.

– Le mot d'état :	bits 0 à 7	code d'erreur lorsque bit 15 = 1.
	bit 8	1 si périphérique occupé ou bit TERMINE.
	bit 9	1 si périphérique prêt.
	bit 15	0 si pas d'erreur, 1 si erreur.

Pour les périphériques de type "bloc" les codes d'erreur ont la signification suivante :

Code	Signification
0	Support protégé en écriture
1	Périphérique inconnu
2	Lecteur non prêt
3	Instruction non connue
4	Erreur de lecture
5	Le bloc de données fourni est incorrect
6	Erreur de recherche
7	Support inconnu
8	Secteur introuvable
9	Défaut de papier dans l'imprimante
10	Erreur d'écriture
11	Erreur de lecture
12	Erreur générale
15	Changement de support interdit

II.4.3. Les blocs de données pour les appels au DD

fonction 0

	Initialisation
2	0
12H à 15H	Adresse du caractère suivant le "=" dans la directive DEVICE de CONFIG.SYS.
16H	0 pour A, 1 pour B, … pour périphériques de type "bloc" (version DOS ≥ 3).
3 à 4	Mot d'état.
0DH	Nombre de périphériques de ce type (type "bloc").
0EH à 11H	Adresse de la première case mémoire libre suivant le *driver*
12 à 15H	Adresse du champ contenant les adresses de BPB (type "bloc").

Cette fonction est appelée lors du chargement du système (ne peut appeler que les fonctions 1 à 0CH ou 30H de INT 21H)

Un BPB ou Bloc de Paramètres BIOS est un ensemble de données attaché aux périphériques de type "bloc". Sa structure est la suivante :

octet	Contenu
0–1	Nombre d'octets par secteur.
2	Nombre de secteurs par granule.
3–4	Nombre de secteurs reservés, secteur d'amorce compris.
5	Nombre de FAT.
6–7	Nombre maximum d'entrées dans la racine.
8–9	Nombre total de secteurs dans l'unité logique.
0AH	Descripteur de support (se retrouve dans le premier octet de la FAT) : – F8 disque dur, – F9 disquette 5"1/4, 17 secteurs par piste, double face, – FC disquette 5"1/4, 9 secteurs par piste, simple face, – FD disquette 5"1/4, 9 secteurs par piste, double face, – FE disquette 5"1/4, 8 secteurs par piste, simple face, – FF disquette 5"1/4, 8 secteurs par piste, double face.
0BH–0CH	Nombre de secteurs par FAT.
0DH–0EH	Nombre de secteurs par piste.
11H–12H	Distance entre le premier secteur du volume et le premier secteur de l'unité de stockage dans le cas où les partitions font moins de 32 Mo.
	DOS≥V4.0 pour partitions ≥32Mo (*Extended BPB*)
13H–14H	Nombre de secteurs par piste.
15H–16H	Distance entre le premier secteur du volume et le premier secteur de l'unité de stockage.
17H–1AH	Nombre de secteurs du volume.

Remarque : toutes les fonctions, sans exception, doivent fixer le bit TERMINE.

fonction 1

	Test de support (type "bloc")
1	Numéro du périphérique.
2	1
0DH	Descripteur de support.
3 à 4	Mot d'état.
0EH	Indicateur de changement : – 0 peut-être – 1 non changé – 0FFH changé
0FH à 12H	Adresse du nom de volume s'il y a eu changement.
Si le support a été changé, il faut refermer tous les tampons utilisés puis recharger la table d'allocation et le répertoire racine.	

fonction 2

	Construction de BPB
1	Numéro du périphérique.
2	2
0DH	Descripteur de support.
0EH à 11H	Adresse du tampon devant recevoir le premier secteur de la FAT.

3 à 4	Mot d'état.
12H à 15H	Adresse du BPB.
Pour une unité de type "caractère", cette fonction fixe simplement le *bit* TERMINE.	

fonction 3
fonction 4
fonction 8
fonction 9
fonction 0CH

	Lecture IOCTL (fonction 44H de INT 21H) Lecture (appels par fonctions DOS de niveau supérieur) Ecriture Ecriture avec vérification Ecriture IOCTL (fonction 44H de INT 21H)
1	Numéro du périphérique (si type "bloc").
2	3, 4, 8, 9, ou 0CH.
0DH	Descripteur de support.
0EH à 11H	Adresse de tampon pour le transfert.
12H à 13H	Nombre de secteurs (type "bloc") ou de caractères.
14H à 15H	Premier secteur à (type "bloc").
3 à 4	Mot d'état.
12H à 13H	Nombre de secteurs ou caractères transférés.
Dans le cas des Lecture ou Ecriture IOCTL, le bit IOCTL de l'en-tête doit être positionné à 1.	

fonction 5

	Lecture de caractère
2	5
3 à 4	Mot d'état.
0EH	Caractère.

fonction 6
fonction 0AH

	Lecture état (type "caractère") Transmission de l'état à la fonction d'appel
2	6 ou 0AH
3 à 4	Mot d'état.

fonction 7

	Vide le tampon d'entrée
2	7
3 à 4	Mot d'état.

fonction 0BH

	Vide le tampon de sortie
2	0BH
3 à 4	Mot d'état.

fonction 0DH
fonction 0EH

	Ouverture périphérique Fermeture périphérique

1 2	Numéro de périphérique (type "bloc"). 0DH ou 0EH
3 à 4	Mot d'état.

fonction 0FH

	Changement de support
2	0FH
3 à 4	Mot d'état (si le bit WAIT est à mis à 0, cela signifie que le support peut être changé).

fonction 10H

	Sortie si non occupé (*spool*)
2 0EH à 11H 12H à 13H	10H Adresse tampon de lecture. Nombre de caractères à envoyer.
3 à 4 12H à 13H	Mot d'état. Mombre de caractères envoyés.

Exemple d'utilisation des appels IOCTL :

```
; DS:DX= adresse d'une chaine contenant le nom d'une unité periphérique
mov    ax,3D00H           ; fonction d'ouverture en lecture
int    21H                ; du DOS
jc     erreur             ; traitement d'erreur
mov    bx,ax              ; numéro logique (handle) transmis par la fonction
mov    ax,4402H           ; lecture d'une caractère IOCTL
mov    dx,offset [tampon] ; adresse du tampon recevant les caractères
int    21H
jc     erreur
…
mov    ax,3E00            ; fonction de fermeture
int    21H
test   ax,ax              ; …
```

II.4.4. Structure générale du bloc de données à partir de DOS 4.0

00	Longueur du bloc de données en octets
01	Numéro de périphérique
02	**Numéro de fonction**
03-04	Mot d'état
05–0CH	Réservés
0DH	Descripteur de support
0EH–11H	Adresse du tampon de transmission
12H–13H	Nombre de secteurs
14H–15H	Numéro du premier secteur
16H–19H	Numéro du premier secteur pour les partitions de plus de 32 Mo (à partir de V.4.0)

CHAPITRE III

Organisation des disques

III.1. Organisation générale

Le disques sont organisés en cylindres, pistes et secteurs. La numérotation des pistes et secteurs est effectuée lors de l'opération de *formattage*. Celle-ci consiste à exécuter des "écriture piste" destinées à inscrire sur chaque piste les secteurs et les *gaps* séparant ceux-ci. Les *gaps* contiennent des informations assurant la synchronisation des données transmises au contrôleur de disque, les numéros des secteurs et le numéro de la piste.

Les secteurs ne sont pas obligatoirement numérotés séquentiellement en fonction de leur position physique sur la piste. Cet *entrelacement* permet d'avoir un délai suffisant entre accès à des secteurs numérotés *n* et *n*+1 sur la piste. Ce paramètre a une grande influence sur les performances globales de la machine pour les applications faisant appel aux disques.

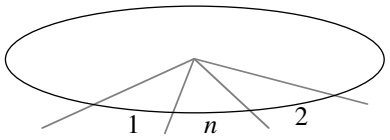


Fig.1 – *Organisation physique en pistes et secteurs*

On retrouve la notion de couche pour les appels aux fonctions d'accès disque :

– les fonctions du BIOS exigent de définir les numéros de cylindre, piste et secteur pour accéder à une information sur le disque,

– un niveau intermédiaire est constitué par les appels aux interruptions 25H et 26H du DOS, pour lesquelles les secteurs sont numérotés séquentiellement de 0 à *n* (ce mode de numérotation est utilisé par l'utilitaire DEBUG),

– un autre niveau, non visible pour l'utilisateur, consiste à organiser le disque en granules (*clusters*) ou ensembles de secteurs (généralement de même numéro physique sur un même cylindre),

– au niveau du DOS, l'accès aux informations est effectué par la spécification du nom d'un fichier. Selon les versions DOS, cet accès est plus ou moins aisé. Pour la version 1, on doit passer par un bloc de contrôle de fichier (FCB). A partir de la version 2, l'accès se fait d'une façon

ressemblant à ce qui est fait en langage évolué. Un *numéro logique* (*handle*) est associé à chaque fichier ouvert. Si on suppose les secteurs numérotés de 0 à *n*, l'organisation générale est la suivante :

Secteur	Contenu
0	Secteur amorce ou *Boot* (Il peut comporter plus d'un secteur, mais c'est rarement le cas)
1 à n_1	Table d'allocation FAT
n_1+1 à n_2	Copie de la FAT
n_2+1 à n_3	Répertoire principal
n_3 à n	Zone de données *Data Area*

III.2. La zone de données du secteur d'amorce

Le premier secteur logique du disque (secteur numéro 0) contient le programme de chargement du système d'exploitation ainsi que des informations relatives au disque et à l'agencement des informations enregistrées sur celui-ci :

Déplacement (hexa)	Contenu
0 à 2	Branchement vers le programme amorce (`JP XXxx`)
3 à A	Identification du système (Ex. IBM 3.30)
B à C	Nombre d'octets par secteur (généralement 512)
D	Nombre de secteurs par granule (*cluster*)
E à F	Nombre de secteurs réservés plus amorce (contient donc 1 au minimum)
10	Nombre de FAT (généralement 2)
11 à 12	Nombre maximum d'entrées dans le répertoire principal
13 à 14	Nombre total de secteurs
15	Identificateur de support
16 à 17	Taille de la FAT en secteurs
18	Nombre de secteurs par piste
1A	Nombre de têtes du lecteur (Nombre de pistes par cylindre)
1C à 1D	Nombre de secteurs cachés

Remarque : les disques durs supportent un partitionnement destiné à faire apparaître le volume physique comme constitué de plusieurs disques durs.

L'identificateur de support peut être le suivant (voir fonction 1CH de INT 21H) :

Code	Type
F0	disquette 3"1/2 (2 faces, 80 pistes, 18 secteurs/piste)
F8	disque dur
F9	disquette 5"1/4 1,2Mo (2 faces, 80 pistes, 15 secteurs/piste) ou 3"1/2, 2 faces, 80 pistes, 9 secteurs/piste
FA	disquette 5"1/4 ou 3"1/2 (1 face, 80 pistes, 8 secteurs/piste)

FB	disquette 5"1/4 ou 3"1/2 (2 faces, 80 pistes, 8 secteurs/piste)
FC	disquette 5"1/4 180 Ko (1 face, 40 pistes, 9 secteurs/piste)
FD	disquette 5"1/4 360 Ko (2 faces, 40 pistes, 9 secteurs/piste)
FE	disquette 5"1/4 160 Ko (1 face, 40 pistes, 8 secteurs/piste)
FF	disquette 5"1/4 320 Ko (2 face, 40 pistes, 8 secteurs/piste)

La table d'allocation contient des liens sur 16 bits (FAT16) lorsque le nombre de granules reste inférieur à 4079. Les informations du bloc de paramètres BIOS en fonction du type sont :

– disquettes 5"1/4 :

	FE	FF	FC	FD	F9
Secteurs/granule	1	2	1	2	1
Secteurs reservés	1	1	1	1	1
Secteurs/FAT	1	1	2	2	7
Nombre de FATs	2	2	2	2	2
Secteurs pour racine (\)	4	7	4	7	14
Entrées max dans racine (\)	64	112	64	112	224

– disquettes 3"1/2 :

	F9	F0
Secteurs/granule	2	1
Secteurs reservés	1	1
Secteurs/FAT	3	9
Nombre de FATs	2	2
Secteurs pour racine(\)	7	14
Entrées max dans racine (\)	112	224

III.3. Constitution d'une entrée du répertoire

A chaque fichier est associée une "entrée" (*entry*) de 32 octets constituée de la façon suivante :

Déplacement	Signification
0 à 7	Nom du fichier.
8 à 10	Extension.
11	Attributs.
12 à 21	Réservés.
22 à 23	Heure de la dernière modification : – bits 0 à 4 : secondes par pas de 2 – bits 5 à 10 : minutes – bits 11 à 15 : heure
24 à 25	Date de la dernière lecture : – bits 0 à 4 : jour – bits 5 à 8 : mois – bits 9 à 15 : année (par rapport à 1980)
26 à 27	Numéro de cluster du premier cluster du fichier.
28 à 31	Taille du fichier en octets.

Aux répertoires du disque sont associées des entrées de même nature que pour les fichiers. Par contre, les deux premières entrées de chaque répertoire sont réservées. Elles correspondent au répertoire courant noté • et au répertoire père ••.

Lorsque le premier caractère a pour code E5, ceci correspond à un fichier qui vient d'être effacé.

III.4. Les attributs des fichiers

Le mot d'attribut est constitué de la façon suivante :

bit	Signification
0	Protection écriture
1	Fichier caché
2	Fichier système
3	Nom de volume
4	Nom de répertoire
5	Indicatif d'archivage (lors d'une écriture ou création ce bit est mis à 1. Les programmes de *backup* le remettent à 0)
6 et 7	Réservés

III.5. La table d'allocation des fichiers FAT

L'emplacement de chaque fichier sur le disque est décrit par une liste chaînée contenue dans une table appelée FAT (*File Allocation Table*). Chaque élément de la FAT est constitué d'un mot de 12 bits (versions DOS 1 et 2 ou version 3 avec moins de 4085 granules) ou 16 bits (version 3 du DOS avec plus de 4085 granules).

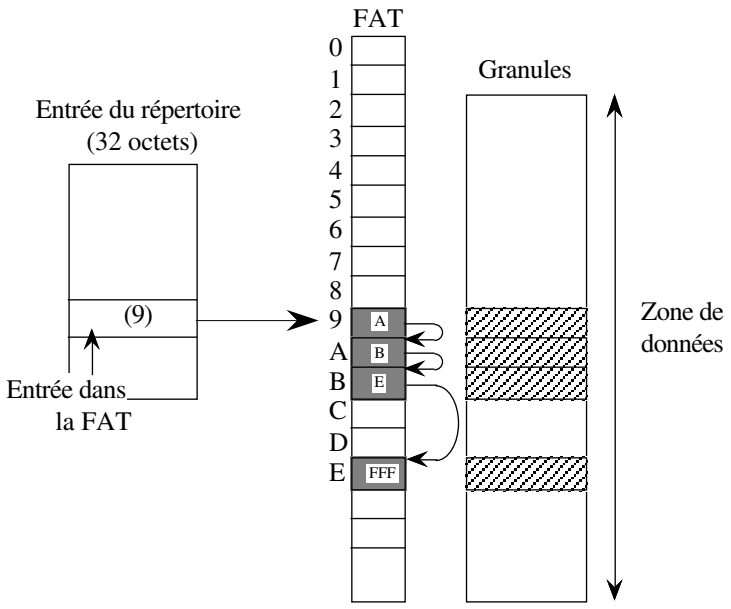


Fig.2 – *Organisation logique du disque*

Le premier octet de la FAT est un octet descripteur du support déjà vu dans la zone des données du *boot*. L'entrée dans la FAT peut être :

– 0 pour un granule libre,
– (F)FF0 à (F)FF6 pour un granule réservé,
– (F)FF7 pour un granule dont un secteur est défectueux,
– (F)FF8 à (F)FFF pour le dernier granule d'un fichier,
– tout autre numéro fournit un chaînage sur le granule suivant dans le fichier.

III.6. La gestion du partitionnement

Lorsqu'une machine est mise sous tension, le premier accès disque se fait sur le secteur 1 de la piste 0, tête 0. Lorsque le disque est un disque dur partitionné, ce secteur n'est pas le secteur d'amorce du système mais un secteur contenant un programme d'accès au *Bootstrap* du système. Ce secteur contient en outre une table de description des partitions du disque.

Considérons le secteur de partition suivant (voir l'exemple traité en fin de chapitre pour l'accès au secteur de partitionnement) :

`0000 FA 2B C0 8E D0 … …`	Début du programme de chargement du *Bootstrap* (il se charge en 0:7C00 puis se recopie en 0:7E00 pour laisser de la place pour le *Bootstrap*)
`0010 F0 …`	
`0020 …`	
…	
…	
`01B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00|80 01`	Table 80 01 01 … (16 octets)
`01C0 01 00 04 05 51 99 11 00-00 00 4B A3 00 00|00 00`	Table 00 00 41 … (16 octets)
`01D0 41 9A 51 05 D1 32 5C A3-00 00 F6 A2 00 00 00 00`	
`01E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00`	
`01F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA`	55 AA marque de fin de secteur

Chaque table de partition est constituée de 16 octets :

Offset	Signification
0	Etat de la partition (80H pour *Boot*, 0 sinon).
1	Tête de lecture de début de partition.
2 à 3	Secteur / cylindre de début de partition (voir INT 13H pour les règles de numérotation).
4	Type de la partition : – 00 entrée non utilisée – 01 DOS avec entrées de la table d'allocation sur 12 bits – 02 et 03 partition XENIX – 04 DOS avec entrées de la table d'allocation sur 16 bits – 05 DOS avec partition étendue – 06 DOS V4.X – DB Concurrent DOS

5	Tête de lecture de fin de partition.
6	Secteur / Cylindre de fin de partition.
8 à 0BH	Distance entre secteur du *Boot* et le premier secteur de la partition.
0CH à 0FH	Nombre de secteurs de la partition.

Ainsi dans cet exemple, on rencontre les information suivantes pour la première partition :

– 80 : la partition décrite ci-après est active,

– 01 : la partition commence tête 1 …

– 01 00 : … avec le secteur 1

– 04 : la FAT possède des entrées sur 16 bits,

– 05 : la partition se termine avec la tête 5,

– 51 99 : cylindre/secteur de fin de partition : 9951_{16} = 1001 1001 0101 0001_2

⇒ numéro de cylindre = 01 (2 bits 6 et 7 de l'octet de poids faible) 1001 1001 = 409_{10}

⇒ numéro de secteur = 01 0001 = 17_{10} (les secteurs d'une piste étant ici numérotés de 1 à 17, tous les secteurs de la dernière piste sont pris)

– distance = 11 00 00 00 : 0000 0011 = 17 : la première piste est réservée. On a seulement le secteur de partition.

– 4B A3 00 00 : la partition comporte 0000 A34B secteurs = 41 803_{10} secteurs (on retrouve bien 410 cylindres de 6 têtes, avec 17 secteurs / pistes, auxquels il faut enlever une piste ⇒ 41 803).

La deuxième partition est définie par :

– 00 : la partition n'est pas active,

– 00 : la partition commence tête 0,

– 41 9A : elle commence avec le secteur $9A41_{16}$ = 1001 1010 0100 0001_2

⇒ 01 1001 0100 = cylindre 410

⇒ 00 0001 = secteur 1

– 51 : la FAT est particulière,

– 05 : la partition se termine avec la tête 5,

– D1 32 : cylindre/secteur de fin de partition : $32D1_{16}$ = 0011 0010 1101 0001_2

⇒ numéro de cylindre = 11 0011 0010 = 818_{10}

⇒ numéro de secteur = 01 0001_2 = 17_{10}

– distance : 5C A3 00 00 : 0000 A35C = 41 820 (on retrouve 41 803 + 17)

– F6 A2 00 00 : la partition comporte 0000 A2F6 secteurs = 41 618 secteurs

III.7. Fonctions de contrôle pour disquettes (INT 13H)

Les fonctions d'accès aux disques utilisent l'interruption 13H. Lorsqu'une erreur se produit à la suite de l'appel d'une fonction un code d'erreur est transmis. Il faut cependant prendre la précaution de tester en priorité la retenue, certains BIOS ne respectant pas tout à fait les codes d'erreur.

Les fonction de gestion des disques utilisent des tables de paramètres dont l'adresse est donnée dans la zone des vecteurs d'interruption. Ces informations sont utilisées pour le BIOS pour initialiser les contrôleurs de disque présents sur la machine. Pour les disquettes l'adresse du champs des paramètres est donnée par INT 1EH, et pour les disques durs par INT 41H et, éventuellement, par INT 46H.

Table de paramètres disque (INT 1EH) :

Déplacement	Contenu
0 à 1	Octets pour la commande SPECIFY du contrôleur
2	Temps d'attente avant arrêt du moteur (en *ticks*)
3	Nombre d'octets par secteur
4	Nombre de secteurs par piste
5	Longueur GAP
6	DTL
7	Longueur GAP pour formatage
8	Octet de remplissage pour formatage
9	Temps de positionnement de tête
10	Temps de démarrage moteur

Codes erreurs pour le contrôle des disques souples :

AH	Signification
1	Erreur de commande contrôleur
2	Marque d'adresse non trouvée
3	Protection écriture
4	Secteur non trouvé
8	Erreur sur accès direct mémoire
9	Erreur de limite sur accès direct mémoire
10H	CRC erroné
20H	Défaillance contrôleur
40	Recherche avortée
80H	Erreur de fin de temporisation

Les appels à l'interruption 13H ne modifient pas BX, CX, DX, SI, DI, BP, CS, DS, SS.

INT 13H/0	Initialisation contrôleur disque
	AH 0

INT 13H/1	Obtention état système après la dernière opération
	AH 1
	AL état

INT 13H/2	Lecture de secteurs d'une piste
	AH 2 DL numéro d'unité (0 pour A, 1 pour B, etc.) DH numéro de tête (≥0) CH numéro de cylindre (≥0) CL numéro de secteur (≥1) AL nombre de secteurs ES:BX adresse de transfert
	AL nombre de secteurs correctement lus

INT 13H/3	Ecriture de secteurs d'une piste
	AH 3 DL numéro d'unité DH numéro de tête (≥0) CH numéro de cylindre (≥0) CL numéro de secteur (≥1) AL nombre de secteurs ES:BX adresse de transfert
	AL nombre de secteurs correctement écrits

INT 13H/4	Vérification CRC
	AH 4 DL numéro d'unité DH numéro de tête (≥0) CH numéro de cylindre (≥0) CL numéro de secteur (≥1) AL nombre de secteurs ES:BX adresse de transfert
	AL nombre de secteurs vérifiés correctement

INT 13H/5	Formatage piste
	AH 5 AL nombre de secteurs sur la piste DL numéro d'unité DH numéro de tête CH numéro de cylindre ES:BX adresse champ descripteur
	Chaque champ de descripteur est associé à un secteur. Il est composé de quatre octets : – numéro de cylindre (mot) – numéro de tête (mot) – numéro de secteur (mot) – nombre d'octets par secteur – 00 pour 128 octets – 01 pour 256 octets – 02 pour 512 octets – 03 pour 1024 octets

Le formattage de pistes avec un secteur de longueur différente de 512 octets a longtemps servi de méthode de protection pour les logiciels. Exemple de table de descripteurs pour une disquette dont on veut formatter la piste 25 de la face 0 avec 7 secteurs de 512 octets et le secteur 4 en 128 octets :

(25,0,1,2) (25,0,2,2) (25,0,3,2) (25,0,4,0) (25,0,5,2) (25,0,6,2) (25,0,7,2) (25,0,8,2)

Les numéros de secteurs peuvent ne pas être consécutifs (définition d'un entrelacement particulier) bien que cette mesure soit de peu d'utilité pour une disquette.

INT 13H/8	Lecture de paramètres
	AH 8 DL numéro d'unité (80H, 81H, …)
	AH code état CH nombre max. de cylindres (y compris les bits 6-7 de CL) CL bits 0-5 nombre de secteurs DL nombre d'unités utilisables DH nombre de têtes – 1

INT 13H/15H	Vérification type de lecteur (PC/AT)
	AH 15H DL numéro d'unité
	AH type du lecteur – 0 si pas de lecteur de disquette – 1 lecteur sans détection de changement de disquette – 2 lecteur avec détection de changement de disquette – 3 lecteur de disque dur

INT 13H/16H	Test changement de disquette (PC/AT)
	AH 16H DL numéro d'unité
	AH – 0 si pas de changement – 6 si changement

INT 13H/17H	Définition du type de formatage (PC/AT)
	AH 17H DL numéro d'unité AL – 1 formater en 320/360 sur lecteur 320/360 Ko – 2 formater en 320/360 sur lecteur 1,2 Mo – 3 formater en 1,2 Mo sur lecteur 1,2 Mo

III.8. Le contrôle des disques durs (INT 13H)

Table de paramètres disque dur (INT 41H ou INT 46H) :

Déplacement	Contenu
0 à 1	Nombre de cylindres
2	Nombre de têtes
3 à 4	Numéro de cylindre pour contrôle d'écriture réduite
5 à 6	numéro de cylindre pour précompensation
7	Nombre max. de bits dans la rafale d'erreurs que l'on peut corriger.

8	octet de contrôle du BIOS : – bits 0-1-2 : – 0,1,2,3,6,7 : pas de 3 ms – 4 : pas de 200 μs – 5 : pas de 70 μs – bits 3-4-5 à 0 – bit 6 contrôle de traitement pour ECC – bit 7 : – 0 nouvelle tentative permise – 1 pas de nouvelle tentative permise
9 à F	réservé

Codes d'erreur pour disques durs :

AH	Signification
1	Erreur de commande contrôleur
2	Marque d'adresse non trouvée
3	Protection écriture
4	Secteur non trouvé
5	Erreur lors du RESET
7	Erreur lors de l'initialisation
9	Erreur de limite sur accès direct mémoire
0BH	Track flag erroné
10H	ECC erroné
11H	ECC erroné mais corrigé
20H	Défaillance contrôleur
40H	Recherche avortée
80H	Erreur de fin de temporisation (lecteur non prêt)
0BBH	Erreur généralisée
0CCH	Pas prêt pour une demande d'état, chargement ou de retour
0FFH	Erreur en lecture de l'octet d'état du contrôleur

INT 13H/0	Initialisation pour disque dur
	AH 0 DL numéro d'unité (80H, 81H, …)
	AH code état
	Le vecteur d'interruption 41H pointe vers une sous-table d'adresses des tables de paramètres disques

INT 13H/1	Lecture d'état
	AH 1 DL numéro d'unité (80H, 81H, …)
	AH code état AL code état de la commande pour l'opération précédente

INT 13H fonction 2 fonction 3	Lecture de secteurs Ecriture de secteurs
	AH 2 (lecture) ou 3 (écriture) AL nombre de secteurs à lire DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (≥1) ES:BX adresse tampon de transfert
	AH code état AL Pour une lecture : longueur de la "rafale" d'erreur s'il y a eu une erreur corrigée
	Le premier secteur d'une piste a le numéro 1.

INT 13H/4	Vérification de secteurs
	AH 4 AL nombre de secteurs à lire (de 1 à 128) DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (≥1)
	AH code état AL Longueur de la "rafale" d'erreur s'il y a eu une erreur corrigée

INT 13H/5	Formattage de piste
	AH 5 AL facteur d'entrelacement DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (inutilisé) ES:BX adresse tampon
	AH code état
	Le tampon a une taille de 512 octets. Seuls les 34 premiers sont utilisés pour spécifier le numéro logique de chaque secteur.

INT 13H/6	Formattage de piste erronée
	AH 6 AL facteur d'entrelacement DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (≥1)
	AH code état

INT 13H/7	Formattage d'unité
	AH 7 AL facteur d'entrelacement DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (inutilisé)
	AH code état
	Le formatage est effectué du cylindre spécifié à la fin du disque

INT 13H/8	Lecture de paramètres
	AH 8 DL numéro d'unité (80H, 81H, …)
	AH code état CH nombre max. de cylindres (y compris les bits 6-7 de CL) CL bits 0-5 nombre de secteurs DL nombre d'unités utilisables DH nombre de têtes – 1

INT 13H/9	Positionnement des paramètres disque
	AH 9 DL numéro d'unité (80H, 81H, …)
	AH code état
	INT 41H pour unité 80H, ou INT 46H pour unité 81H, doivent pointer vers les tables de descripteur.

INT 13H fonction 0AH fonction 0BH	Lecture longue Ecriture longue
	AH 0AH ou 0BH AL nombre de secteurs DL numéro d'unité (80H, 81H, …) DH numéro de tête (≥0) CH numéro de cylindre (y compris les deux bits 6-7 de CL) (≥0) CL bits 0-5 numéro du premier secteur (≥1) ES:BX adresse de tampon de transfert
	AH code état
	Ces commandes sont utilisées essentiellement à des fins de diagnostic. 4 octets d'ECC sont rajoutés aux 512 octets du secteur.

INT 13H/0DH	Recalibrage disque dur
	AH 0DH DL numéro d'unité (80H, 81H, …)
	AH code état

INT 13H/14H	Diagnostic contrôleur
	AH 14H
	AH code état

INT 13H/15H	Détermination type de disque
	AH 15H DL numéro d'unité (80H, 81H, …)
	AH code résultat : – 0 pas de disque – 1 ou 2 lecteur de disquette – 3 disque dur : CX-DX contient le nombre de secteurs
	Le vecteur d'interruption 41H pointe vers une sous-table d'adresses vers les tables de paramètres disques

III.9. Exemple

La consultation du secteur de partitionnement peut être faite rapidement en écrivant un petit programme sous debug :

```
-a cs:100                          ; assemblage à partir de l'offset 100H
XXXX:0100 mov ax,201               ; fonction de lecture (ah=2), 1 seul secteur (al=1)
XXXX:0103 mov dx,80                ; dl=80H (premier disque dur), dh=0=tête de lecture
XXXX:0106 mov cx,1                 ; cx=1 (piste 0, secteur 1)
XXXX:0109 mov bx,1000              ; bx=1000H (tampon de lecture à l'offset 1000H)
XXXX:010C int 13                   ; appel de l'interruption
XXXX:010E int 3                    ; point d'arrêt
XXXX:010F                          ;
-g=100                             ; exécution
…
-d 1000 11ff                       ; et on regarde …
 …
```

CHAPITRE IV

Fonctions d'affichage

Les premiers *PC* sont apparus équipés de cartes d'affichage MDA (*Monochrome Display Adapter*), autorisant l'affichage en mode texte de 25 lignes de 80 caractères, suivis aussitôt de l'introduction d'un mode d'affichage dit "haute résolution" (CGA *Color Graphics Adapter*) graphique (640 par 200 pixels). Parallèlement ont été introduites les cartes Hercules graphiques (348 lignes de 720 pixels) qui n'ont jamais été officiellement soutenues par IBM.

L'évolution suivante, avec l'introduction des *PC/AT*, a vu l'apparition des cartes EGA (*Enhanced Graphics Adapter*), texte et graphique de 350 lignes de 640 pixels, et plus éphémèrement des cartes PGA (P*rofessionnal Graphics Adapter*).

La gamme des PS/2 est apparue équipée de cartes MCGA (*MultiColor Graphics Array*) ou VGA (*Video Graphics Array*). Ce dernier mode d'affichage est maintenant proposé de façon systématique sur tous les *PC* et compatibles.

Un groupe de constructeurs de cartes d'affichage (VESA *Video Electronics Standards Association*) a introduit un nouveau standard nommé *Super VGA*. Ce standard définit un ensemble matériel + logiciel (architecture et définition d'appels au BIOS).

Les définitions les plus hautes (1024 x 768) sont fournies par les modes d'affichage IBM 8514/A et XGA, et TIGA (*Texas Instruments Graphics Architecture*).

N°	Mode	Coul.	Texte	Segment	Définition	Pages	Supporté par les modes d'affichage :
0/1	AN	16	40x25	B800	320x200	8	CGA, MCGA, EGA, VGA
					320x350	8	EGA, MCGA, VGA
					320x400	8	MCGA
					360x400	8	VGA
2/3	AN	16	80x25	B800	640x200	4	CGA
					640x200	8	EGA, MCGA, VGA
					640x350	8	EGA, MCGA, VGA
					640x400	8	MCGA
					720x400	8	MCGA, VGA

4/5	GR	4	40x25	B800	320x200	1	CGA, EGA, MCGA, VGA
6	GR	2	80x25	B800	640x200	1	CGA, EGA, MCGA, VGA
7	AN	mono	80x25	B000	720x350	1	MDA
					720x350	8	EGA, MCGA, VGA
					720x400	8	MCGA, VGA
					640x200	4	
8	GR	16	20x25	B800	160x200	1	PC-Junior
9	GR	16	40x25	B800	320x200	1	PC-Junior
A	GR	4	80x25	B800	640x200	1	PC-Junior
D	GR	16	40x25	A000	320x200	8	EGA, MCGA, VGA
E	GR	16	80x25	A000	640x200	4	EGA, MCGA, VGA
F	GR	mono	80x25	A000	640x350	4	EGA, MCGA, VGA
10	GR	16	80x25	A000	640x350	2	EGA, MCGA, VGA
11	GR	2	80x30	A000	640x480	2	MCGA, VGA
12	GR	16	80x30	A000	640x480	1	VGA
13	GR	256	40x25	A000	320x200	1	MCGA, VGA
14	GR	16			640x200		EGA/VGA
15	GR	mono			640x350		EGA/VGA
16	GR	4/16			640x350		EGA/VGA
17	GR	2			640x480		VGA
18	GR	16			640x480		VGA
19	GR	256			320x200		VGA (1 octet par pixel)
30					720x350		3270
6A	GR	16			800x600		Super VGA (≈mode 12)
100	GR	256			640x400		Super VGA 1.0
101	GR	256			640x480		Super VGA 1.0
102	GR	16			800x600		Super VGA 1.0
103	GR	256			800x600		Super VGA 1.0
104	GR	16			1024x768		Super VGA 1.0
105	GR	256			1024x768		Super VGA 1.0
106	GR	16			1280x1024		Super VGA 1.0
107	GR	256			1280x1024		Super VGA 1.0
108	AN		80x60				Super VGA 1.1
109	AN		132x25				Super VGA 1.1
10A	AN		132x43				Super VGA 1.1
10B	AN		132x50				Super VGA 1.1
10C	AN		132x60				Super VGA 1.1

6A	GR	16			800x600		VESA
100	GR	256			640x400		VESA
101	GR	256			640x480		VESA
102	GR	16			800x600		VESA
103	GR	256			800x600		VESA
104	GR	16			1024x768		VESA
105	GR	256			1024x768		VESA
106	GR	16			1280x1024		VESA
107	GR	256			1280x1024		VESA

L'interruption 10H du BIOS permet de gérer ces différents modes d'affichage.

IV.1. Les fonctions de l'interruption INT 10H

IV.1.1. Les fonctions standards

INT 10H/0	Positionnement mode vidéo
	AH 0 AL mode vidéo
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. BIOS EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS ne sont pas modifiés. Si le bit 7 du mode vidéo est mis à 1, la mémoire n'est pas effacée.

INT 10H/1	Positionnement attributs curseur
	AH 1 CH ligne de trame de début pour le curseur : – bit 7 : curseur visible ou invisible – bits 6-0 à 0 si curseur normal – bits 6-0 à 1 si clignotement modifié CL ligne de trame de fin pour le curseur
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/2	Positionnement curseur
	AH 2 DH ligne DL colonne BH numéro de page d'affichage
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/3	Acquisition position et attributs du curseur
	AH 3 BH numéro de page d'affichage
	DH ligne DL colonne CH,CL attributs du curseur
	BX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/5	Sélection page active (page de texte seulement)
	AH 5 AL nouvelle page
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/6	Défilement vers le haut
	AH 6 CH,CL ligne, colonne coin haut gauche DH,DL ligne, colonne coin bas droit BH attribut lignes insérées AL nombre de lignes (0 pour effacer)
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés. Lorsque AL=0, la zone est remplie d'espaces.

INT 10H/7	Défilement vers le bas
	AH 7 CH,CL ligne, colonne coin haut gauche DH,DL ligne, colonne coin bas droit BH attribut lignes insérées AL nombre de lignes (0 pour effacer)
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés. Lorsque AL=0, la zone est remplie d'espaces.

INT 10H/8	Lecture caractère et attribut
	AH 8 BH page de texte
	AL code caractère AH attribut (en mode texte)
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/9	Ecriture caractère et attribut
	AH 9 AL code caractère BL attribut (si bit 7 à 1 en mode graphique il se produit un XOR avec le contenu de l'écran) BH numéro de page de texte CX nombre de caractères à écrire
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/0AH	Ecriture caractère
	AH 0AH AL code caractère BH page de texte EGA/VGA : BL = couleur du caractère en mode graphique CX nombre de caractères à écrire (répétition)
	(En EGA/VGA, si bit 7 de BL est à 1, il se produit, en mode graphique, un XOR avec le contenu de l'écran) BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/0BH	Définition palette/couleur cadre et fond
	AH 0BH BH 0 pour définir la couleur du cadre et du fond (en mode texte le cadre seulement) (à n'utiliser qu'en définition 320x200 ou 640x200) 1 pour définir la palette (à n'utiliser qu'en définition 320x200 ou 640x200) BL si BH=0 : couleur à utiliser (bit 0 à 3 + bit 4 pour la brillance) si BH=1 : numéro de palette
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/0CH	Ecriture pixel
	AH 0CH AL couleur CX,DX colonne,ligne EGA/VGA : BH = page écran
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/0DH	Lecture pixel
	AH 0DH CX,DX colonne,ligne EGA/VGA : BH = page écran
	AL couleur
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés.

INT 10H/0EH	Ecriture en mode télétype
	AH 0EH AL code du caractère BL couleur du caractère en mode graphique
	BX, CX, DX, CS, SS, DS ne sont pas modifiés. EGA/VGA : BX, CX, DX, SI, DI, BP, CS, SS, DS non modifiés. Cette fonction d'écriture comprend le test et l'exécution de la sonnerie.

INT 10H/0FH	Obtention état vidéo
	AH 0FH
	AL mode vidéo courant AH nombre de colonnes BH numéro de page active affichée
	BX, CX, DX, CS, SS, DS ne sont pas modifiés.

INT 10H/10H	Contrôle de couleur et palette (EGA/MCGA/VGA)
	AH 10H AL = 0 fixe le contenu d'un registre BH = registre à adresser (de 0 à 15) BL = contenu AL = 1 fixe la couleur du cadre de l 'écran BH = couleur (rangée dans le registre de palette 16) AL = 2 fixe registre de contrôle des attributs ES:DX pointe sur table des couleurs (17 registres de palette) AL = 3 bascule d'intensité/clignotement en alphanumérique BL = 0 si couleur fond intense BL = 1 si clignotement AL = 7 lecture registre de palette (VGA) entrée : BL = numéro de registre de palette sortie : BH = contenu du registre de palette AL = 8 lecture registre de "bordure" (*OverScan*) (VGA) sortie : BH = contenu du registre de "bordure" AL = 9 lecture de tous registres palettes et bordure (VGA) sortie : ES:DX pointe sur le tampon résultat (17 octets)

AL = 10H écriture reg. couleur (convertisseur NA) (VGA)
BX = numéro de registre de couleur (0 à 255)
CH = vert
CL = Bleu
DH = rouge
AL = 12H écriture bloc de reg. couleur (convertisseur NA) (VGA)
BX = numéro de registre de couleur (0 à 255)
CX = nombre de registres à définir
ES:DX = adresse du tampon de définition (CX×3 octets)
AL = 13H sélect. mode couleur (convertisseur NA) (VGA)
BL = 0 : bit 0 de BH recopié dans bit 7 du reg de contrôle de mode (sélection en 4 ou 16 groupes de 64 ou 16 couleurs)
BL = 1 : BH recopié dans registre de sélection de couleur (sélection d'un groupe de couleurs)
AL = 15H lecture d'un reg. couleur (convertisseur NA)
entrée : BX = numéro de registre de couleur (0 à 255)
sortie : CH = vert
sortie : CL = Bleu
sortie : DH = rouge
AL = 17H lecture d'un bloc de reg. couleur (convertisseur NA)
BX = numéro du premier registre de couleur (0 à 255)
CX = nombre de registres à définir
ES:DX = adresse du tampon de définition
AL = 18H défintion du registre de masque (VGA)
BL = valeur du registre de masque
AL = 19H lecture du registre de masque (VGA)
sortie : BL = valeur du registre de masque
AL = 1AH lecture mode de sélection couleur (VGA)
BL = bit 7 du registre de contrôle de mode
BH = valeur du registre de sélection de couleur
AL = 1BH contrôle de sommation de niveaux de gris (VGA)
pour le convertisseur : BX = numéro du registre à convertir
CX = nombre de registres à convertir

BX, CX, DX, SI, DI, BP, CS, SS, DS ne sont pas modifiés.
(voir notes sur l'affichage en fin de chapitre)

INT 10H/11H	Interface du générateur de caractères (EGA/MCGA/VGA)

AH 11H

AL = 0 charge le jeu de caract. défini
BL = numéro de table (0/1)
BH = nombre de lignes par caract.
CX = nombre de caract. dans la table
DX = code ASCII du premier de ces caractères
ES:BP = adresse de la table

AL = 1 charge le jeu de caract. 8x14 (de la ROM vers la RAM)
BL = numéro de table (0/1)

AL = 2 charge le jeu de caract. 8x8 (de la ROM vers la RAM)
BL = numéro de table (0/1)

AL = 3 active un jeu de caractères
BL = numéro de jeu (0/1)

AL = 4 charge des caractères alphanumériques 8x16 (VGA)
BL = numéro de table (0/1)

AL = 10H charge et active le jeu de caract. défini
(idem 0 avec reprog. du contrôleur CRTC)
BL = numéro de table (0/1)
BH = nombre de lignes par caract.
CX = nombre de caract. dans la table
DX = code ASCII du premier de ces caractères
ES:BP = adresse de la table

AL = 11H charge et active le jeu de caract. 8x14
(idem 1 avec reprog. du contrôleur CRTC)
BL = numéro de table (0/1)

AL = 12H charge et active le jeu de caract. 8x8
(idem 2 avec reprog. du contrôleur CRTC)
BL = numéro de table (0/1)

AL = 14H charge et active le jeu de caract. 8x16 (VGA)
BL = numéro de table (0/1)

AL = 20H modifie le vecteur INT 1FH
ES:BP = nouvelle adresse table de caractères (128 à 255)

AL = 21H modifie le vecteur INT 43H
ES:BP = nouvelle adresse de la table de caractères (0 à 255)
CX = nombre de lignes par caractère
si BL = 0 DL indique le nombre de lignes de texte à l'écran
(1 pour 14 lignes, 2 pour 25, 3 pour 43)

AL = 22H fixe l'adresse INT 1FH vers table du BIOS

AL = 23H fixe l'adresse INT 43H vers table du BIOS

AL = 30H lire les informations d'affichage
entrée :
BH = 0 contenu de INT 1FH
BH = 1 contenu de INT 43H
BH = 2 adresse table 8x14
BH = 3 adresse table 8x8
BH = 4 adresse table 8x8 (seconde moitié)
BH = 5 adresse table 9x14 (supplémentaire)
BH = adresse table 8x16 (supplémentaire)
BH = adresse table 9x16 (supplémentaire)
sortie :
CX = hauteur caractères
DL = nombre de colonnes−1
ES:BP = adresse cherchée

BX, CX, DX, SI, DI, BP, CS, SS, DS ne sont pas modifiés sauf : sous-fonction 30H : BX, DH, SI, DI, CS, SS, DS ne sont pas modifiés.

INT 10H/12H	Contrôle des fonctions (EGA/MCGA/VGA)

AH 12H
BL = 10H lire configuration
BL = 20H active recopie d'écran
BL = 30H sélection résolution vert. en alphanum. (EGA/VGA)
0 (200) 1(350) 2 (400 pour VGA)
BL = 31H valid./inhib. chargement par défaut de la palette (VGA)
AL = 0 (valide) / 1 (inhibe)
BL = 32H valid./inhib. adressage de la RAM vidéo (VGA)
AL = 0 (valide) / 1 (inhibe)
BL = 33H valid./inhib. sommation niveaux de gris (VGA)
AL = 0 (valide) / 1 (inhibe)
BL = 34H valid./inhib. émulation curseur (VGA)
AL = 0 (valide) / 1 (inhibe)
BL = 35H valide/inhibe temporairement la sortie écran (VGA)
AL = 0 (valide) / 1 (inhibe)

sortie (lecture de configuration) :
BH = 0 moniteur couleur / haute résolution
BH = 1 moniteur mono.
BL = 0 à 3 pour taille RAM 64 à 256 Ko
CL = 8 moniteur couleur connecté
CL = 9 moniteur EGA /MultiSync.
CL = 0BH moniteur mono.

BX, CX, DX, SI, DI, BP, CS, SS, DS ne sont pas modifiés

INT 10H fonction 13H	Ecriture de chaîne de caractères (PC/AT)

	AH 13H AL 0 attribut dans BL, position du curseur non modifiée 1 attribut dans BL, position du curseur actualisée 2 attribut dans tampon, position du curseur non modifiée 3 attribut dans tampon, position du curseur actualisée BL octet d'attribut (AL=0 ou 1) CX nombre de caractères dans la chaîne DH numéro de ligne DL numéro de colonne BH numéro de page ES:BP : adresse du tampon
	Lorsqu'on utilise AL=1 ou 2, le tampon contient contient les caractères avec leur attribut. Si AL=0 ou 1, le tampon ne contient que le code des caractères et ceux-ci sont affichés avec l'attribut défini par BL. BX, CX, DX, CS, SS, DS ne sont pas modifiés.

INT 10H/1AH	Contrôle de type d'écran (MCGA/VGA)
	AH 1AH AL = 0 lecture du code d'affichage AL = 1 écriture du code d'affichage
	En lecture si AL ≠ 1AH il n'y a pas de BIOS VGA BL = code carte vidéo active BH = code carte vidéo non active FF = carte inconnue 00 = pas de carte 01 = MDA / écran monochrome 02 = CGA / écran CGA 04 = EGA / écran EGA/MultiSync 05 = EGA / écran monochrome 07 = VGA / écran monochrome analogique 08 = MDA / écran couleur analogique

INT 10H/1BH	Lecture d'état du système d'affichage (MCGA/VGA)
	AH 1BH BX 0 ES:DI pointeur sur tampon
	AL 1BH

offset	Contenu du tampon

0 à 3	Adresse de la table des informations BIOS (16 octets)
4	Mode vidéo courant
5 à 6	Nombre de colonnes ou points
7 à 8	Taille page écran dans la RAM vidéo
9 à A	Adresse de la page écran active dans la RAM vidéo
B à 1A	Positions curseur dans les pages écran
1B	Ligne de fin du curseur
1C	Ligne de début du curseur
1D	Numéro de la page écran active
1E à 1F	Adresse du contrôleur vidéo
20	
21	
22	Nombre de lignes affichables
23 à 24	Hauteur des caractères en points
25	
26	
27	Nombre de couleurs affichables
29	Nombre de pages écran
2A	
2B	
2C	
2D	
2E à 2F	Réservé
31	Taille de la RAM vidéo
32 à 3F	Réservé

INT 10H/1CH	Sauvegarde et restauration état affichage (VGA)
	AH 1CH AL = 0 donne la taille du tampon d'état AL = 1 sauvegarde de l'état d'affichage AL = 2 restaure l'état d'affichage

IV.1.2. Les fonctions "Super VGA"

Ces fonctions sont appelées avec AH=4FH et un numéro de fonction dans AL.

INT 10H fonction 4F00H	Lecture état (Super VGA)
	AX 4F00H ES:DI : adresse d'un tampon de 256 octets

INT 10H fonction 4F01H	Lecture mode et informations sur le mode (Super VGA)
	AX 4F01H ES:DI : adresse d'un tampon de 256 octets
	CX numéro du mode Informations permettant de traiter des caractéristiques non standard de l'affichage.

INT 10H fonction 4F02H fonction 4F03H	Fixer mode (Super VGA) (2) Lire mode (Super VGA) (3)
	AX 4F02H ou 4F03H BX mode (sous-fonction 2)
	BX mode (sous-fonction 3)

INT 10H fonction 4F04H	Sauvegarde état vidéo(Super VGA)
	AX 4F04H
	Le contenu de tous les registres du contrôleur ainsi que des informations BIOS sont sauvegardées.

En mode 1024×768 l'espace mémoire nécessaire avec une architecture de type VGA est supérieur à 64K. La technique utilisée ressemble à celle qui est mise en œuvre dans la mémoire paginée :

– en mode normal (*Single Window Bank Switching*), le banc 0 permet d'accéder au premier segment d'affichage, le banc 1 au segment suivant, etc. En mode 1024×768, les 512 premières lignes sont accessibles dans le banc 0 et le banc 1 correspond aux 256 lignes suivantes.

– en mode "à fenêtres recouvrantes" (*Overlapping Windows*) on met un banc en *lecture seule*, et l'autre en mode *écriture seule*, pour pouvoir accéder "simultanément" aux deux bancs à travers les mêmes adresses mémoire.

– en mode "à fenêtres non recouvrantes" (*Dual Nonoverlapping Windows*), dans le mode 1024×768, on peut disposer de quatre bancs et les deux fenêtres d'affichage occupant au total trois granules de 32K peuvent être réparties à la demande sur les quatre bancs de 32K.

Le mode super-VGA laisse une certaine liberté aux fabricants de cartes quant au nombre et à la taille des bancs de mémoire sur la carte vidéo.

INT 10H fonction 4FH05	Contrôle des *bancs* vidéo (Super VGA)

	AX 4F05H BH : 0 définition du banc BL : numéro de fenêtre (0 pour A, 1 pour B) DX : numéro du banc BH : 1 lecture du banc

INT 10H fonction 4F06H	Définition largeur d'écran (Super VGA)
	AX 4F06H

INT 10H fonction 4F07H	Définition et lecture adresses d'affichage (Super VGA) (Contrôle d'affichage multi-écran)
	AX 4F07H

IV.1.3. Les fonctions VESA

INT 10H fonction 4F00H	Lire les caractéristiques de la carte (VESA)
	AX 4F00H ES:DI adresse d'un tampon de 256 octets
	AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

Le tampon contient les informations suivantes :

offset	contenu
0	marqueur : chaîne "VESA"
4	version, numéro primaire
5	version, numéro secondaire
6	pointeur (4 octets) sur une chaîne ASCIIZ contenant le nom du fabricant de la carte.
0AH	4 octets à 0
OEH	pointeur (4 octets) vers une liste de *mots* codant les modes vidéo soutenus par la carte (le dernier est 0FFFFH).

INT 10H fonction 4F01H	Lire les caractéristiques d'affichage (VESA)

AX	4F01H
CX	mode vidéo
ES:DI	adresse d'un tampon de 29 octets
AX	004FH si carte VESA et résultat OK
AX	014FH si carte VESA et résultat pas OK
Les registres ne sont pas modifiés.	

Le tampon contient les informations suivantes :

offset	contenu
0	mot d'état : bit 0 : 1 si compatibilité carte-moniteur bit 1 : 1 si les informations optionnelles sont bien disponibles bot 2 : 1 support des modes texte du BIOS bit 3 : 1 couleur bit 4 : 1 mode graphique
2	octet d'état première fenêtre d'accès : bit 0 : 1 fenêtre disponible bit 1 : 1 écriture en RAM vidéo activée bit 2 : 1 écriture en RAM vidéo activée
3	octet d'état deuxième fenêtre d'accès : bit 0 : 1 fenêtre disponible bit 1 : 1 écriture en RAM vidéo activée bit 2 : 1 écriture en RAM vidéo activée
4	Granularité pour les déplacements en Ko
6	Taille des fenêtres d'accès en Ko
8	Adresse de segment de la première fenêtre
0AH	Adresse de segment de la seconde fenêtre
0CH	Pointeur
10H	Nombre d'octets pour une ligne de pixels
12H	Nombre de points en X par caractère
14H	Nombre de points en Y par caractère
16H	Largeur de la matrice de points en X par caractère
17H	Hauteur de la matrice de points en X par caractère
18H	Nombre de plans de bits
19H	Nombre de bits / pixel
1AH	Nombre de blocs de mémoire
1BH	Organisation de la mémoire 0 texte, 1 CGA, 2 Hercules, 3 EGA/VGA 16 couleurs, 4 format compacté 2 à 4 bits/pixels, 5 EGA/VGA 256 couleurs
1CH	Taille des blocs de mémoire en Ko

INT 10H fonction 4F02H	Fixer le mode vidéo (VESA)
	AX 4F02H BX mode vidéo (si bit 15=1: pas d'effacement)
	AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

INT 10H fonction 4F03H	Lire le mode vidéo (VESA)
	AX 4F03H
	BX mode vidéo AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

INT 10H fonct.4F04H/0	Lecture/écriture de l'état de la carte (VESA) (Taille tampon nécessaire aux fonctions suivantes)
	AX 4F04H DL 0 CX bit 0 état de la partie vidéo bit 1 zone de données BIOS bit 2 contenu de la table des convertisseurs (DAC) bit 3 état VESA
	BX nombre de blocs de 64 octets nécessaires pour le tampon AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

INT 10H fonct.4F04H/1	Ecriture de l'état de la carte (VESA)
	AX 4F04H DL 1 CX même définition que précédemment ES:BX adresse tampon
	AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

INT 10H fonct.4F04H/2	Lecture de l'état de la carte (VESA)
	AX 4F04H DL 2 CX même définition que précédemment ES:BX adresse tampon

AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
Les registres ne sont pas modifiés.

INT 10H fonct.4F05H/0	Accès à la RAM vidéo (VESA) (fixer fenêtre d'accès sur la RAM vidéo)
	AX 4F05H BH 0 BL fenêtre d'accès (0/1) DX adresse de départ
	AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

INT 10H fonct.4F05H/1	Accès à la RAM vidéo (VESA) (lire numéro de fenêtre d'accès sur la RAM vidéo)
	AX 4F05H BH 1
	DX adresse de départ AX 004FH si carte VESA et résultat OK AX 014FH si carte VESA et résultat pas OK
	Les registres ne sont pas modifiés.

IV.1.4. Exemples

– Fonction de défilement vers le haut, appelée à partir d'un langage évolué par :

```
CALL DEFILH(ligneH,colonneG,ligneB,colonneB,nombreL)
```

Les paramètres correspondent respectivement aux numéros de ligne haute, de colonne gauche, de ligne basse, de colonne droite et au nombre de lignes de défilement. On suppose que les adresses (seuls les *offsets* sont passés dans la pile; cela correspond au modèle dit *medium*) des arguments sont passées dans la pile dans l'ordre où ceux-ci apparaissent dans l'appel de la procédure. Toutes les variables sont entières.

```
DEFILH procfar
  push      bp
  mov       bp,sp
  add       bp,0Eh
  push      ax
  push      bx
  push      cx
  push      dx
  pushf
  mov       bx,[bp]        ; on lit l'adresse du numéro de ligne haute
  mov       ch,[bx]        ; ch := numéro de ligne haute
```

```
  mov     bx,[bp-2]     ; on lit l'adresse du numéro de colonne gauche
  mov     cl,[bx]       ; ch := numéro de colonne gauche
  mov     bx,[bp-4]     ; on lit l'adresse du numéro de ligne basse
  mov     dh,[bx]       ; ch := numéro de ligne basse
  mov     bx,[bp-6]     ; on lit l'adresse du numéro de colonne droite
  mov     dl,[bx]       ; ch := numéro de colonne droite
  mov     bx,[bp-8]     ; on lit l'adresse du nombre de lignes
  mov     al,[bx]       ; ch := nombre de lignes
  mov     bh,1          ; on force l'attribut des lignes nouvelles à 1
  mov     ah,6          ; code de la fonction
  int     10H
  popf
  pop     dx
  pop     cx
  pop     bx
  pop     ax
  pop     bp
  retf    8
DEFILH    endp
```

– Programme de lecture de la configuration contrôleur écran/écran (MS/BASIC) :

```
DEFINT I-N
TYPE RegTypeX
     ax    AS INTEGER
     bx    AS INTEGER
     cx    AS INTEGER
     dx    AS INTEGER
     bp    AS INTEGER
     si    AS INTEGER
     di    AS INTEGER
     flags AS INTEGER
     ds    AS INTEGER
     es    AS INTEGER
END TYPE
DIM RegIn AS RegTypeX, RegOut AS RegTypeX
CLS
intNum = 16: RegIn.ax = &H1200: RegIn.bx = &H10
CALL interruptX(intNum, RegIn, RegOut)
Mon1 = RegOut.bx AND 255
Mon2 = RegOut.bx \ 256
Mon3 = RegOut.cx
PRINT "-------- contrôle BIOS standard ---------"
PRINT "Taille mémoire carte :"; 64 * (1 + Mon1); "K"
SELECT CASE Mon2
    CASE 0
```

```
        PRINT "Moniteur couleur/haute résolution"
    CASE 1
        PRINT "Moniteur N/B"
    CASE ELSE
        PRINT "?"
END SELECT
SELECT CASE Mon3
    CASE 8
        PRINT "Moniteur couleur connecté"
    CASE 9
        PRINT "Moniteur EGA/MultiSync"
    CASE 11
        PRINT "Moniteur Monochrome"
    CASE ELSE
        PRINT "?"
END SELECT
PRINT "-------- contrôle MCGA/VGA ---------"
'* contrôle type (MCGA/VGA)
RegIn.ax = &H1A00
CALL interruptX(intNum, RegIn, RegOut)
IF ((RegOut.ax AND 255) = &H1A) THEN
    iCodeCA = RegOut.bx AND 255
    PRINT "Code carte activée : "; iCodeCA
    SELECT CASE iCodeCA
        CASE 255
            PRINT "    Type carte inconnu"
        CASE 0
            PRINT "    Pas de carte"
        CASE 1
            PRINT "    MDA/Ecran Monochrome"
        CASE 2
            PRINT "    CGA/Ecran CGA"
        CASE 4
            PRINT "    EGA/Ecran EGA/MultiSync"
        CASE 5
            PRINT "    EGA/Ecran Monochrome"
        CASE 7
            PRINT "    VGA/Ecran Monochrome analogique"
        CASE 8
            PRINT "    VGA/Ecran couleur analogique"
        CASE ELSE
    END SELECT
    iCodeCA = RegOut.bx \ 256
    PRINT "Code carte non activée : "; iCodeCA
    SELECT CASE iCodeCA
        CASE 255
```

```
            PRINT "    Type carte inconnu"
        CASE 0
            PRINT "    Pas de carte"
        CASE 1
            PRINT "    MDA/Ecran Monochrome"
        CASE 2
            PRINT "    CGA/Ecran CGA"
        CASE 4
            PRINT "    EGA/Ecran EGA/MultiSync"
        CASE 5
            PRINT "    EGA/Ecran Monochrome"
        CASE 7
            PRINT "    VGA/Ecran Monochrome analogique"
        CASE 8
            PRINT "    VGA/Ecran couleur analogique"
        CASE ELSE
    END SELECT
ELSE
    PRINT "Il n'y a pas de BIOS VGA"
END IF
Mon1 = RegOut.bx AND 255
Mon2 = RegOut.bx \ 256
Mon3 = RegOut.cx
…
```

IV.2. Notes sur l'affichage

IV.2.1. Attributs et Palettes

En dehors de l'affichage MDA qui est un mode purement *texte*, les types d'affichage des PC disposent de deux modes de fonctionnement : le mode *texte* et le mode *graphique*.

– MDA ou *Hercules* texte : dans ce mode texte, chaque caractère est codé sur deux octets :

– son code ASCII,
– un octet d'attribut définissant l'apparance du caractère :

7	6	5	4	3	2	1	0
(Cl)	fond (F)			(I)	caractère (C)		

Le bit 7 (Cl) est l'*indicateur de clignotement*, le bit 3 (I) celui de *brillance*. Le champ des bits 0 à 2 (C) caractérise le caractère, et les bits 4 à 6 (F) le fond associé. Le premier champ peut prendre les valeur 0 (noir), 1 (souligné) ou 7 (blanc), tandis que le second peut prendre les valeurs 0 (noir) ou 7 (blanc).

– CGA texte 25x80 ou 25x40 : le codage est similaire au précédent. La différence provient des champs (C) et (F) qui contrôlent la couleur du caractère et du fond correspondant. En fait, le bit de brillance permet d'obtenir 8 couleurs supplémentaires pour le caractère, qui en compte ainsi 16, alors que le fonds n'en compte que 8 (à moins que l'on ne désactive le bit de clignotement - bit 5 du registre de sélection de mode 3D8H - celui-ci étant alors considéré comme un bit supplémentaire de couleur).

– CGA graphique 640x200 : chaque octet correspond à huit pixels. La première ligne est associée aux octets d'*offset* 0 à 79, la seconde aux octets 2000_{16} à 2000_{16} + 79, la troisième aux octets 80 à 159, etc … Le texte s'inscrit dans une matrice 25x80.

– CGA graphique 320x200 : chaque octet correspond à quatre pixels. A chaque pixel sont associés deux bits codant quatre couleurs ou quatre niveaux de gris dans la palette courante (la couleur 0 peut être choisie librement, tandis que les trois autres correspondent soit à "Turquoise, Violet, Blanc" soit à "Vert, Rouge, Jaune"). L'organisation mémoire est similaire à celle du mode précédent. Le texte s'inscrit dans une matrice 25x40.

– Hercules graphique 720x348 : chaque octet correspond à huit pixels. La première ligne est associée aux octets d'*offset* 0 à 79, la seconde aux octets 2000_{16} à 2000_{16} + 79, la troisième aux octets 4000_{16} à 4000_{16} + 79, la quatrième aux octets 6000_{16} à 6000_{16} + 79, la cinquième aux octets 80 à 159, la sixième aux octets 2000_{16} + 80 à 2000_{16} + 159, etc …

– EGA texte : le codage des caractères est similaire à celui du mode CGA texte. La seule différence est que les champs (C)+(I) et (F) ne codent pas directement la couleur mais donnent accès à 1 parmi 16 registres de *palette* dont la constitution est la suivante :

5	4	3	2	1	0
R	V	B	r	v	b

Chaque bit code une couleur, soit brillante (RVB) soit normale (rvb). On dispose ainsi de 16 couleurs parmi 64.

– VGA texte : la différence avec le mode EGA est que le contenu du registre de palette ne donne plus la couleur mais un index dans un groupe de 256 registres 18 bits (*Digital Analog Converter Color Table* en entrée du convertisseur Digital/Analogique). En principe ceci permet d'obtenir 64 couleurs parmi 256 K couleurs. Etant donnés les 256 registres de la table des couleurs, on peut commuter rapidement entre quatre jeux de 64 couleurs. Il existe un autre mode de fonctionnement dans lequel on n'utilise que les bits 0 à 3 des registres de palette. On dispose alors de 16 jeux de 16 couleurs parmi 256 K.

– EGA graphique 640x350 : chaque pixel est codé de manière interne sur 4 bits (4 *bits Planar Model*). Chaque bit appartient à un plan différent. Chaque écriture ou lecture d'un octet permet d'accéder à 8 bits dans un ou plusieurs plans :

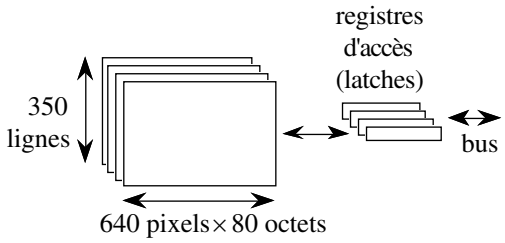


Fig.1 – *Mode EGA graphique*

– En mode VGA graphique, l'organisation de la mémoire vidéo est la même qu'en EGA. Il n'y a qu'un seul mode (mode 13H) dans lequel l'adressage se fait pixel par pixel, la couleur étant codée sur 8 bits. Lorsque le *bit* 7 du registre de contrôle de mode est à 0, on peut schématiser l'affichage selon le principe indiqué figure 2. Les quatre *bits* d'attribut donnent accès à une table de 16 registres 6 *bits*. Ces 6 *bits* fournissent à leur tour un *offset* dans un bloc de 64 registres 18 *bits* de la table des couleurs.

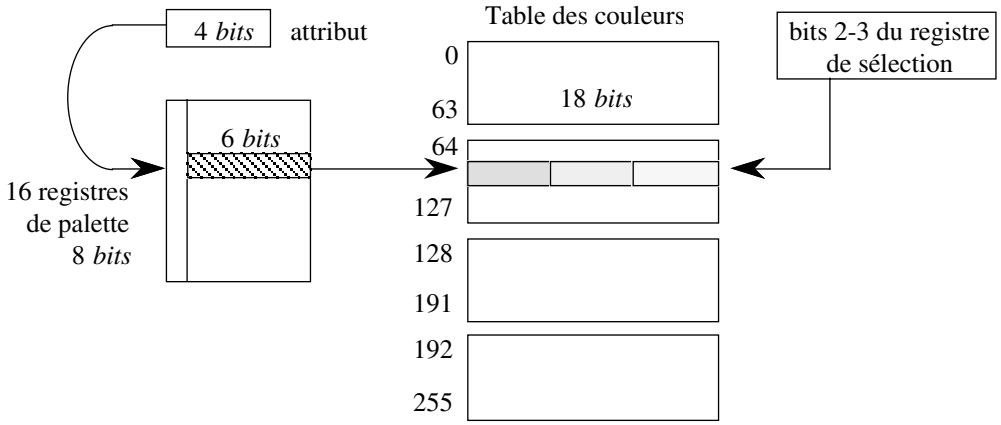


Fig.2 – *Affichage en mode VGA (1)*

La table des couleurs possède de 256 mots de 18 *bits* (6 *bits* pour chacune des couleurs Rouge, Vert et Bleu). Il y a quatre blocs sélectionnés par les *bits* 2 et 3 du registre de sélection de couleur. Si le *bit* 7 du registre de contrôle de mode est à 1, l'affichage est schématisé :

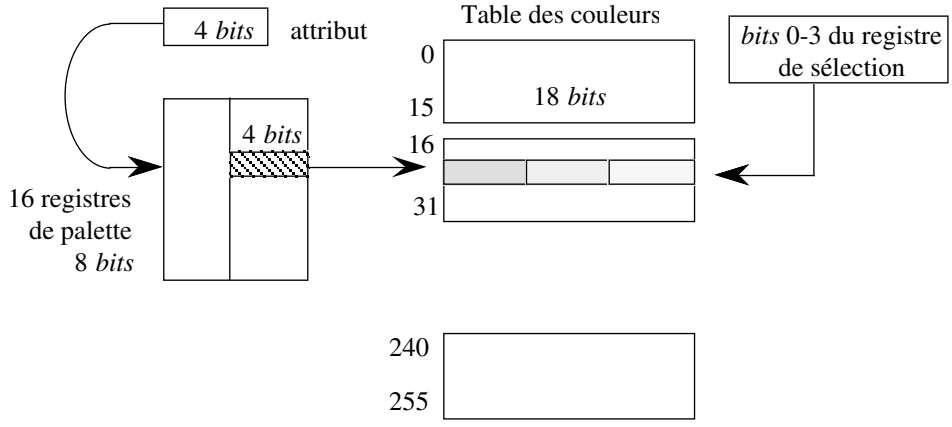


Fig.3 – *Affichage en mode VGA (2)*

Les quatre bits d'attribut donnent toujours accès à la table de palette 16 registres, mais seuls 4 bits seront utilisés. Ces 4 bits fournissent un *offset* dans un bloc de 16 mots de 18 bits. Il y a seize blocs sélectionnés à partir des bits 0, 1, 2 et 3 du registre de sélection de couleur.

IV.2.2. Les contrôleurs

Le contrôle de l'affichage est composé[1] :

– d'un contrôle du signal vidéo *CRTC* (signaux de synchronisation, nombre de lignes de balayage, nombre de points par ligne, gestion du curseur, …),
– d'un contrôle de séquencement (modes d'affichage texte/graphique, contrôle d'accès aux plans d'affichage, …
– d'un contrôle de l'accès à la mémoire d'affichage (modes d'écriture et lecture),
– d'un contrôle aux registres d'attributs (couleurs et palettes).

Dans le cas EGA/VGA, les contrôleurs associés ont respectivement pour adresse :

– 3B4-3B5 ou 3D4-3D5 pour le contrôleur *CRTC* (*Cathode Ray Tube Controller*). L'adresse 3Bx ou 3Dx est déterminée par programmation du registre d'adresse 3C2 (*Miscellaneous Output Register*),
– 3C4-3C5 pour le séquenceur,
– 3CE-3CF pour l'accès à la mémoire écran,
– 3C0 pour le contrôle des attributs.

En pratique, la programmation des modes vidéo et de l'accès aux registres de palette se fera en passant par les fonctions du BIOS. Le seul contrôleur auquel on peut avoir besoin d'accéder est le contrôleur d'accès à la mémoire écran lorsqu'on doit manipuler des images en mode graphique. Les fonctions du BIOS, en nombre trop limité, n'offrent, généralement pas une rapidité suffisante.

La plupart de ces contrôleurs possède plusieurs registres, mais seulement deux adresses. L'accès aux registres internes se fait en écrivant, à la première adresse, le numéro du registre auquel on veut accéder, puis, à la seconde, on lit ou écrit la valeur voulue.

Exemple : l'accès à la mémoire écran est fonction des *mode d'écriture et lecture*. Ceux-ci sont généralement égaux à zéro, sauf cas particulier (voir les deux exemples d'utilisation des modes d'écriture 1 et 2). Le registre numéro 5 permet de programmer ce mode :

ØØØØrØww

La programmation du mode *read* 0-*write* 2 (`r=0,w=2`) se fait de la façon suivante :

```
mov   dx,3CEH     ; adresse du registre recevant le numéro de
                  ; registre de contrôle
mov   al,5        ; registre de définition de mode = 5
out   dx,al       ;
inc   dx          ; on passe au registre suivant (3CFH)
```

[1]EGA : Mode d'emploi, Micro-Systèmes, Janvier 1989

```
mov   al,2        ; read 0, write 2
out   dx,al       ;
```

Les modes de lecture et écriture ont les fonctions suivantes[2] :

– *Read* 0 : on accède au contenu des quatre plans. Quatre octets sont recopiés simultanément dans les quatre *latches*. Les deux bits de poids faible du registre *Read Map Select* permet de sélectionner le plan dont l'octet sera ainsi recupéré par le programme de lecture. Pour accéder à huit pixels "dans le même octet", il faudra donc quatre lectures à la même adresse.

– *Read* 1 : sert à comparer la couleur définie par le contenu du registre *Color Compare* avec les couleurs des huit pixels à l'adresse définie par le programme. L'octet lu indique par des 1 lesquels des pixels ont la couleur désignée. Le registre *Color Don't Care* valide ou non les bits impliqués dans la comparaison.

– *Write* 0 : ce mode permet d'accéder aux quatre plans en écriture. Le registre *Bit Mask* indique par des 1 les pixels concernés par l'opération définie par les *bits* 4 et 3 du registre *Function Select*. Les quatre bits de poids faible du registre *Set/Reset* sont combinés respectivement avec les bits de chacun des quatre plans. Le registre *Enable Set/Reset* autorise ou non cette combinaison (le plus souvent ces quatre bits sont à 1).

Lorsque le registre *Enable Set/Reset* est à 0, c'est l'octet émis par le processeur qui est combiné simultanément avec les quatre octets des quatre plans selon la fonction indiquée.

– *Write* 1 : ce mode est destiné aux transferts de mémoire écran à mémoire écran (voir exemple).

– *Write* 2 : ce mode est destiné à manipuler la mémoire écran pixel par pixel. Le registre *Bit Mask* définit les bits qui seront modifiés simultanément dans l'octet adressé. On peut ainsi fixer la couleur de huit pixels en une opération d'écriture (voir exemple). On notera qu'en fait il faut effectuer une opération de lecture-écriture à chaque fois que l'on modifie un pixel pour recharger les *latches*.

Les registres sont les suivants :

	Registre	
0	*Set/Reset Register*	
1	*Enable Set/Reset Register*	
2	*Color Compare Register*	En mode *read*-1 spécifie la couleur que l'on compare aux pixels de l'octet adressé.
3	*Function Select Register*	Les bits 4 et 3 indiquent la fonction à réaliser : – 00 : on remplace, – 01 : ET logique, – 10 : OU logique, – 11 : OU exclusif
4	*Read Map Select Register*	
5	*Mode Register*	Définition des modes de lecture/écriture
6	*Miscellaneous Register*	
7	*Color Don't Care Register*	En mode *read*-1, indique quels pixels de l'octet adressé on exclue de la comparaison.
8	*Bit Mask Register*	Un 1 indique que le pixel correspondant de l'octet adressé sera impliqué dans l'opération d'écriture.

[2]La bible PC, Editions Micro Applications

IV.2.3. Exemples

Sous-programme de décalage vers la gauche d'une zone d'écran graphique EGA/VGA alignée horizontalement sur des limites d'octets. Les arguments d'appels sont les suivants :

– `iCoulF` : couleur de fond pour la partie décalée,
– `ipas` : nombre d'octets de décalage,
– `icol` : colonne gauche (c'est un multiple de huit),
– `indc` : nombre d'octets constituant la ligne,
– `ilmin` : ligne haute,
– `indMax` : nombre de lignes–1.
Le mode d'écriture utilisé est le mode 1.

```
;*----------------- appel -------------------------------*
;*  CALL decaleG(iCoulF,ipas, icol, indc, ilMin, indMax) *
;*-----------------------------------------------------*
         public   decaleG
decaleG proc    far
        jmp     debdecG
FarPtr  dd      ?
indl    dw      ?
indc    dw      ?
ilMin   dw      ?
colonne dw      ?
indice  dw      ?                              ; num. de ligne relatif
TabOr   db      0,1,2,4,8,16,32,64,128
cte     dw      80                             ; 80 colonnes
saveDI  dw      ?
ipas    dw      ?
ohg     dw      ?
interr  db      ?
iCoulF  db      ?

debdecG label   near
        push    bp
        mov     bp, sp
        pusha
        push    es
        pushf
        mov     bx,[bp+6]                      ; nbre de lignes
        mov     ax,[bx]
        mov     word ptr[indl],ax
        mov     bx,[bp+8]                      ; idem pour la ligne haute
        mov     ax,[bx]
        mov     word ptr[ilMin],ax
        mov     bx,[bp+10]                     ; idem pour la colonne max
        mov     ax,[bx]
```

```
        mov     word ptr[indc],ax
        mov     bx,[bp+12]                  ; idem pour la colonne gauche
        mov     ax,[bx]
        mov     word ptr[colonne],ax
        mov     bx,[bp+14]                  ; idem pour pas de décalage
        mov     ax,[bx]
        mov     word ptr[ipas],ax
        mov     bx,[bp+16]                  ; couleur du fond
        mov     al,[bx]
        mov     byte ptr[iCoulF],al

        mov     ax,[indc]                   ; nbre de décalages
        sub     ax,[ipas]                   ; = indc - ipas
        mov     [indc],ax

        mov     dx,3CEH                     ; registre d'adresse
        mov     ax,105H                     ; on met 1 dans reg5
        out     dx,ax                       ; read 0, write 1
        mov     ax,0A000H                   ; segment EGA/VGA
        mov     es,ax                       ; dans ES

        mov     ax,[ilMin]                  ; on initialise les registres
        mul     word ptr[cte]               ; ilMin*80+iCol
        add     ax,[colonne]
        mov     [ohg],ax
        mov     di,ax                       ; DI=offset octet haut gauche
        mov     [saveDI],ax
        mov     si,ax
        add     si,[ipas]                   ; octet suivant
        mov     cx,[indl]
        inc     cx                          ; on répète indl+1 fois

bclDecG :
        push    cx
        mov     cx,word ptr[indc]
        rep     movs es:byte ptr[di],es:[si]  ; décale indc caract.
        add     [saveDI],80                 ; on passe à la ligne
        mov     ax,[saveDI]                 ; suivante
        mov     di,ax
        mov     si,ax
        add     si,[ipas]
        pop     cx
        loop    bclDecG

        ;-------- on efface ce qui reste ------------------------
        ; bit mask register reg8=FF par défaut (tous les bits)
```

```
        ; function select   reg3=00 par défaut (preset)
        ; enable set/reset  reg1=00 par défaut (->F tous plans)
        ; set/reset         reg0=00 par défaut (-> code couleur)
        ; mode write 0      reg5=00
        ;-------------------------------------------------------
        mov     dx,3CEH                     ; registre d'adresse
        mov     ax,005H                     ; on met 0 dans reg5
        out     dx,ax                       ; mode write 0
        mov     dx,3CEH                     ; registre d'adresse
        mov     ax,0F01H                    ; on met F dans reg1
        out     dx,ax                       ;
        mov     dx,3CEH                     ; registre d'adresse
        xor     al,al                       ; on met iCoulF dans reg0
        mov     ah,[iCoulF]
        out     dx,ax                       ; (marron)

        mov     ax,[ohg]                    ; point haut gauche
        add     ax,[indc]                   ; on ajoute indc
        mov     [saveDI],ax
        mov     di,ax
        mov     cx,[indl]
        inc     cx                          ; on répète indl+1 fois
bclDecG2 :
        push    cx
        mov     cx,word ptr[ipas]
bclDecG3 :
        mov     es:byte ptr[di],ah
        inc     di
        loop    bclDecG3

        add     [saveDI],80                 ; on passe à la ligne
        mov     ax,[saveDI]                 ; suivante
        mov     di,ax
        pop     cx
        loop    bclDecG2

        popf
        pop     es
        popa
        pop     bp
        retf    12
decaleG endp
```

Exemple de sous-programme d'affichage en mode graphique EGA/VGA : un tampon contient les couleurs, sur huit bits, de chacun des pixels du rectangle à afficher. Les arguments d'appels sont

passés par adresse, ici des *adresses courtes*, c'est-à-dire uniquement les *offsets* des variables dans le segment de données courant (*model medium*) :

– `iColG` : colonne gauche,
– `iColD` : colonne droite,
– `iLigH` : ligne haute,
– `iLigB` : ligne basse,
– `SegTamp` : segment du tampon contenant la couleur des pixels,
– `PtrTamp` : *offset* du tampon.

Le code est du code 286. Le mode d'écriture 2 est utilisé le plus souvent pour dessiner un pixel dont la couleur est fournie par le programme. Le pixel allumé est défini par un masque (contenu du registre 8), un bit à 1 indiquant celui qui est concerné.

```
        .286
        .SALL
.MODEL MEDIUM
.CODE
;*-------------- Appel ----------------------------------------
;* CALL PlotTab(iColG,iColD,iLigH,iLigB,SegTamp,PtrTamp)
;*---------------------------------------------------------------
SegEGACO equ    0A000H
        public  PlotTab
PlotTab  proc      far
        jmp     debPlotTab
iColG   dw    ?       ; colonne gauche
iColD   dw    ?       ; colonne droite
iLigH   dw    ?       ; ligne haute
iLigB   dw    ?       ; ligne basse
kBitG   db    ?       ; position 1° pixel dans l'octet de début de ligne
kBitD   db    ?       ; position dernierpixel dans dernier octet ligne
iCGs8   dw    ?       ; offset 1° octet dans segment d'affichage
iCDs8L  dw    ?       ; offset dernier octet (sur la même ligne iLigH)
iCDs8   dw    ?       ; offset dernier octet (sur iLigB)
NbCol   dw    ?       ; nombre de colonnes
NbLig   dw    ?       ; nombre de colonnes
NbOct   dw    ?       ; nombre d'octets complets dans la ligne
;
debPlotTab   label   near
    push    bp
    mov     bp, sp
    pusha
    push    es
    push    ds
    pushf
    mov     bx,SegEGACO
    mov     es,bx
    ;*--------- traitement colonne gauche
```

```
    mov     bx,[bp+16]              ; Colonne gauche
    mov     ax,[bx]
    mov     word ptr cs:[iColG],ax
    mov     bx,ax
    and     ax,7                    ; iColG MOD 8
    mov     byte ptr cs:[kBitG],al
    shr     bx,3                    ; iColG \ 8
    mov     word ptr cs:[iCGs8],bx
    mov     word ptr cs:[NbOct],bx
    ;*--------- traitement colonne droite
    mov     bx,[bp+14]              ; Colonne droite
    mov     ax,[bx]
    mov     word ptr cs:[iColD],ax
    mov     word ptr cs:[NbCol],ax
    mov     bx,ax
    and     ax,7                    ; iColD MOD 8
    mov     byte ptr cs:[kBitD],al
    shr     bx,3                    ; iColD \ 8
    mov     word ptr cs:[iCDs8],bx
    mov     word ptr cs:[iCDs8L],bx
    dec     bx
    sub     bx,word ptr cs:[NbOct]
    jns     CalcNbCol
    xor     bx,bx
CalcNbCol:
    mov     word ptr cs:[NbOct],bx
    ;*--------- nBCol = (iColD - iColG + 1) --- nombre de colonnes
    mov     ax,word ptr cs:[iColG]
    sub     word ptr cs:[NbCol],ax
    inc     word ptr cs:[NbCol]
    ;*--------- traitement ligne haute
    mov     bx,[bp+12]
    mov     ax,[bx]                 ; Ligne haute
    mov     word ptr cs:[iLigH],ax
    mov     cx,80
    mul     cx                      ; calcul offset premier octet
    add     word ptr cs:[iCGs8],ax
    add     word ptr cs:[iCDs8L],ax ; es:[di] sera l'adresse en
    mov     di,word ptr cs:[iCGs8]  ; mémoire écran
    ;*--------- traitement ligne basse
    mov     bx,[bp+10]              ; Ligne basse
    mov     ax,[bx]
    mov     word ptr cs:[iLigB],ax
    mov     word ptr cs:[NbLig],ax
    mul     cx                      ; Calcul offset dernier octet
    add     word ptr cs:[iCDs8],ax
```

```
    ;*--------- NbLig = (iLigB - iLigH + 1) --- nombre de lignes
    mov     ax,word ptr cs:[iLigH]
    sub     word ptr cs:[NbLig],ax
    inc     word ptr cs:[NbLig]
    ;*--------- offset iTab(0)
    mov     bx,[bp+6]
    mov     ax,[bx]
    mov     si,ax
    ;*--------- segment iTab(0)
    mov     bx,[bp+8]
    mov     ax,[bx]
    mov     ds,ax            ; ds:[si] est l'adresse dans la table

;*-------------- on commence le traitement
    mov     dx,3CEH
    mov     ax,205H   ; 5=registre de mode, 2=(écriture 2, lecture 0)
    out     dx,ax     ; on écrit le mode
    mov     ax,300H   ; 3=reg."Function Select Reg." en mode "preset"
    out     dx,ax     ;
    mov     al,8      ; numéro du registre de masquage
    out     dx,al
    inc     dx        ; on accède maintenant au registre de masquage

    mov     ax,word ptr cs:[iCGs8]  ; on regarde si octets gauche et
    cmp     ax,word ptr cs:[iCDs8L] ; droit sont les mêmes
    mov     cx,word ptr cs:[NbLig]
    jne     bcl2PlotTab
;*---------------------------- un seul octet --------------------
    mov     bh,byte ptr cs:[kBitG]
    mov     bl,byte ptr cs:[kBitD]
bcl1PlotTab:
    call    PlotEGA
    add     di,80               ; on passe à la ligne suivante
    loop    bcl1PlotTab
    jmp     finPlotTab

;*---------------------------- plusieurs octets ----------------------
bcl2PlotTab:                        ;*---------------------------------
    mov     bh,byte ptr cs:[kBitG]
    mov     bl,7
    push    di
    call    PlotEGA                 ; premier octet --------
    inc     di

    push    cx
    mov     cx,word ptr cs:[NbOct]  ; octets entiers -------
```

```
    or      cx,cx
    jz      dernOct
bcl3PlotTab:
    mov     bx,7                    ; pixel gauche=0, droit=7
    call    PlotEGA
    inc     di
    loop    bcl3PlotTab
dernOct:
    pop     cx
    xor     bh,bh
    mov     bl,byte ptr cs:[kBitD]  ; dernier octet --------
    call    PlotEGA
    pop     di
    add     di,80           ; on passe à la ligne suivante
    cmp     si,8000H
    jbe     NoCorrection
    sub     si,1000H
    mov     ax,ds
    add     ax,100H
    mov     ds,ax
NoCorrection:
      loop    bcl2PlotTab         ;*--------------------------------

finPlotTab:
    mov     dx,3CEH
    mov     ax,0FF08H    ; masquage validé
    out     dx,ax       ;
    ;*--- puis on revient en mode Read 0, Write 0
    mov     ax,5        ; 5=reg. de mode, 0=(écriture 0, lecture 0)
    out     dx,ax       ; on écrit le mode

    popf
    pop     ds
    pop     es
    popa
    pop     bp
    retf    12
PlotTab endp

;*-----------------------
;* bh position premier pixel dans octet (0 = bit de gauche)
;* bl position dernier pixel dans octet (7 = bit de droite)
;* al sert d'octet de masquage
;* dx est l'adresse du port VGA=3CFH (on a déjà mis 8 en 3CEH
;* pour désigner le registre de masquage)
;*-----------------------
```

```
PlotEGA  proc  near
    push    cx
    ;*------ masque=2^(7-bh) -> AL
    mov     cl,bh
    inc     cl
    xor     al,al
    stc
    rcr     al,cl

    ; nombre de pixels concernés dans l'octet
    mov     cl,bl
    sub     cl,bh
    inc     cl
    xor     ch,ch       ; pour avoir CX=compteur de boucles

bclePlotEGA:                        ;*--------------------------------
    out     dx,al       ; on écrit le masque
    shr     al,1        ; on passe au pixel suivant
    mov     ah,byte ptr [si]    ; lecture d'un octet dans table
    inc     si
    xchg    ah,byte ptr es:[di]
    loop    bclePlotEGA         ;*--------------------------------

    pop     cx
    ret
PlotEGA endp

PLOT_TEXT ends
    end
```

Exemple de sous-programmes de lecture d'un plan complet d'écran :

```
            .286
            .SALL
.MODEL MEDIUM
.CODE
;
;*------------------------------------------------------------*
;*                     AFFICHAGE D'UN PLAN                    *
;*------------------------------------------------------------*
;* Format d'appel Basic :                                     *
;* CALL PlotPlan(BYVAL noplan,BYVAL nblig,BYVAL i1,BYVAL i2)  *
;*------------------------------------------------------------*
;* Définition des paramètres :                                *
;* noplan = 1,2,4,8 (n° du plan)                              *
;* nblig = nombre de lignes d'affichage (480 ou 350)          *
```

```
;* i1    = VARSEG(iTab(0))                                        *
;* i2    = VARPTR(iTab(0))                                        *
;*---------------------------------------------------------------*
        public  PlotPlan
PlotPlan proc    far
        push    bp
        mov     bp,sp
        pusha
        push    es
        push    ds
        push    ss
        pushf
        ;-------- on récupère l'adresse du tableau dans DS:SI
        mov     si,[bp+6]               ; offset iTab(0)
        mov     bx,[bp+8]               ;
        mov     ds,bx                   ; segment iTab(0)
        ;-------- nombre d'octets par ligne
        mov     cx,80
        ;-------- on récupère le nombre de lignes
        mov     ax,[bp+10]              ;
        mul     cx                      ;
        mov     cx,ax                   ; nbre d'octets dans CX

        mov     di,0                    ; adresse plan vidÇo
        mov     ax,0A000H               ;
        mov     es,ax                   ; dans ES:DI

        cld
        mov     dx,3CEH                 ; enable set/reset (n° du plan)
        mov     ax,1003h                ; fonction select 2 (ou logique)
        out     dx,ax
        mov     ax,0f00h                ; set/reset register
        out     dx,ax

        mov     al,1
        mov     ah,[bp+12]
        out     dx,ax
        mov     al,8                    ; mask register
        out     dx,al
        inc     dx
PlotPlanLab1:
        outsb
        mov     al,es:[di]
        mov     es:[di],al
        inc     di
        loop    PlotPlanLab1
```

```
        popf
        pop     ss
        pop     ds
        pop     es
        popa
        pop     bp
        retf    8
PlotPlan endp

;*-------------------------------------------------------------*
;*                     LECTURE D'UN PLAN                       *
;*-------------------------------------------------------------*
;* Format d'appel Basic :                                      *
;* CALL ReadPlan(BYVAL noplan,BYVAL nblig,BYVAL i1,BYVAL i2)   *
;*       (passage par valeur des paramètres)                   *
;*-------------------------------------------------------------*
;* Définition des paramètres :                                 *
;* noplan = 0,1,2,3 (n° du plan)                               *
;* nblig  = nombre de lignes d'affichage (480 ou 350)          *
;* i1     = VARSEG(iTab(0))                                    *
;* i2     = VARPTR(iTab(0))                                    *
;*-------------------------------------------------------------*
        public  ReadPlan
ReadPlan proc    far
        push    bp
        mov     bp, sp
        pusha
        push    es
        push    ds
        push    ss
        pushf
        ;-------- on récupère l'adresse du tableau dans ES:DI
        mov     di,[bp+6]                   ; offset iTab(0)
        mov     bx,[bp+8]                   ;
        mov     es,bx                       ; segment iTab(0)
        ;-------- nombre d'octets par ligne
        mov     cx,80
        ;-------- on récupère le nombre de lignes
        mov     ax,[bp+10]                  ;
        mul     cx                          ;
        mov     cx,ax                       ; nbre d'octets dans CX

        mov     si,0                        ; adresse plan vidéo
        mov     ax,0A000H                   ;
        mov     ds,ax                       ; dans ES:DI
```

```
        cld
        mov     dx,3CEH                 ;
        mov     al,4
        mov     ah,[bp+12]              ; n° de plan à sauvegarder
        out     dx,ax
        rep     movsb                   ; transfert

        popf
        pop     ss
        pop     ds
        pop     es
        popa
        pop     bp
        retf    8
ReadPlan endp

PLOTPLAN_TEXT ends
        end
```

CHAPITRE V

Les fonctions du BIOS

Les fonctions décrites ici ne comprennent pas les interruptions 10H (chapitre sur l'affichage) et 13H (chapitre sur les disques).

V.1. Lecture du vecteur d'équipement (INT 11H)

INT 11H	Lecture vecteur d'équipement (PC/XT)
	AX vecteur d'équipement (0040:0010) *bit* – 15-14 nombre d'imprimantes – 12 présence de l'adaptateur de jeux – 11-10-9 nombre de liaisons série – 8 présence de canal DMA – 7-6 nombre de lecteurs de disques souples (si bit 0=1) – 5-4 type vidéo au démarrage – 3-2 taille mémoire sur carte mère (par blocs de 16K) – 0 existence de lecteur de disquette
	Seul AX est modifié.

INT 11H	Lecture vecteur d'équipement (AT)
	AX vecteur d'équipement (0040:0010) – 15-14 nombre d'imprimantes – 11-10-9 nombre de liaisons série – 7-6 nombre de lecteurs de disques souples – 5-4 type vidéo au démarrage) – 1 (à 1 si coprocesseur) – 0 existence de lecteur de disquette
	Seul AX est modifié.

V.2. Lecture de la taille mémoire (INT 12H)

INT 12H	Obtention taille mémoire
	AX nombre de Ko

V.3. Fonctions de contrôle des ports série (INT 14H)

Octet d'état du canal :

bit	Signification
0	Caractère reçu
1	Caractère écrit dans le registre de données
2	Erreur de parité
3	Procole non conservé
4	Connexion interrompue
5	Registre de données vide
6	Registre à décalage vide
7	*Time-out*

Octet d'état du modem :

bit	Signification
0	Modem prêt à émettre (*)
1	Modem activé (*)
2	Détection front arrière de sonnerie (*)
3	Liaison modem établie (*)
4	Modem prêt à émettre
5	Modem activé
6	Détection front arrière de sonnerie
7	Liaison modem établie

(*) 1 indique qu'il y a eu un changement d'état

INT 14H/0	Initialisation du port de communication
	AH 0 AL paramètres de transmission – *bits* 1-0 : 2 = 7 *bits*, 3 = 8 *bits* – *bits* 2 : 0 = 1 *bit* de STOP, 1 = 2 *bits* de STOP – *bits* 5-4 : 0 pas de parité, 1 parité impaire, 2 parité paire – *bits* 7-6-5 : de 0 à 7 : 110, 150, 300, 600, 1200, 2400, 4800, 9600
	AH état de la ligne AL état du modem

INT 14H/1	Envoi de caractère
	AH 1 AL caractère
	AH – *bit* 7 à 0 caractère transmis – *bit* 7 à 1 caractère non transmis Les autres *bits* contiennent l'état du canal

INT 14H/2	Réception de caractère
	AH 2
	AL caractère reçu AH 0 pas d'erreur sinon état du canal

INT 14H/3	Etat chaîne de communication
	AH 3
	AL état modem AH état canal

V.4. L'interruption INT 15H

L'interruption 15H n'est pas utilisée de la même façon sur tous les modèles *XT*, *AT* ou *PS*/2. Seules quelques fonctions sont décrites ci-après. Certaines autres servent au système d'exploitation et non au BIOS. La touche SYS (PC-AT), en particulier, appelle l'interruption 15H (appelée aussi *interruption cassette*) en transmettant AH=85H. Si AL vaut 0 c'est que la touche est activée. Si AL vaut 1, c'est qu'elle a été relâchée.

INT 15H	Interruption cassette ou touche SYS
	La touche SYS peut facilement être utilisée pour des applications particulières en déroutant celle-ci.

Autres fonctions de INT 15H (PC-AT) :

INT 15H/83H	Horloge temps réel
	AH 83H CX:DX nombre de μs du délai ES:BX drapeau
	Le drapeau ne peut être testé qu'après qu'après fin du délai.

INT 15H/86H	Fonction d'attente
	AH 86H CX:DX nombre de μs du délai
	Le programme ne reprend la main qu'après fin du délai.

INT 15H/87H	Transfert de mémoire (passage au mode protégé du 80286)
	AH 87H CX nombre de mots à déplacer ES:SI adresse de la GDT

AH	code état	– 0 pas d'erreur – 1 erreur de parité sur la mémoire – 2 GDT incorrecte – 3 le mode protégé n'a pu être initialisé

INT 15H/88H	Lecture de la taille mémoire étendue
AH	88H
Taille mémoire étendue en Ko	

INT 15H/89H	Passage en mode protégé
AH	89H

INT 15H/0C0H	Acquisition adresse des paramètres de configuration
AH	0C0H
Table des paramètres (octets 6 à 9 : réservés) : – octets 0-1 : longueur de la table qui suit (8) – octet 2 : identification du modèle – octet 3 : type – octet 4 : version BIOS – octet 5 : mot de configuration	

V.5. Fonctions de contrôle clavier (INT 16H)

INT 16H/0	Lecture de caractère
AH	0
AX	code ASCII / code géographique
L'en-tête du tampon clavier avance dans la file d'attente du clavier.	

INT 16H/1	Etat du tampon clavier	
AH	1	
Z	0	AX = *scan code* / code ASCII
	1	aucun code disponible
L'en-tête du clavier n'est pas modifiée		

INT 16H/2	Drapeaux clavier	
AH	2	
AL	drapeaux clavier	– *bit* 0 SHIFT droit enfoncé – *bit* 1 SHIFT gauche enfoncé – *bit* 2 CONTROL enfoncé – *bit* 3 ALT enfoncé – *bit* 4 état du verrouillage du défilement – *bit* 5 état du verrouillage des touches num. – *bit* 6 état du verrouillage des majuscules – *bit* 7 état touche INSERTION

INT 16H/3	Réglage pour la répétition automatique des touches
	AX 0305H BH délai jusqu'à répétition de 1/4*s* à 1*s* BL vitesse de répétition
	Dépend du BIOS

INT 16H/5	Met un code dans le tampon clavier
	AH 05H CX *scan code* / code ASCII
	AL 0 opération effectuée 1 tampon saturé
	Dépend du BIOS

INT 16H/10H	Test de clavier étendu
	AH 10H
	AX *scan code* / code ASCII
	Comme pour la fonction 0 mais pour les claviers étendus du type AT, Dépend du BIOS

INT 16H/11H	Test de clavier étendu
	AH 11H
	Z 0 AX = *scan code* / code ASCII (Si AL=0, AH=code ASCII pour clavier étendu), 1 aucun code disponible
	Dépend du BIOS

V.6. Fonctions de contrôle port parallèle (INT 17H)

INT 17H/0	Impression de caractère
	AH 0 AL code du caractère DX numéro d'imprimante
	AH état

INT 17H/1	Initialisation du port imprimante
	AH 1 DX numéro d'imprimante
	AH état

INT 17H/2	Etat imprimante
	AH 2 DX numéro d'imprimante

AH état	– *bit* 0 fin de temporisation (*time out*) – *bit* 3 erreur d'E/S – *bit* 4 sélection – *bit* 5 fin de papier – *bit* 6 ACK – *bit* 7 imprimante occupée (0 si BUSY)

V.7. Autres fonctions

V.7.1. L'appel au moniteur (INT 18H)

INT 18H	Appel moniteur

V.7.2. Le pré-chargement (INT 19H)

L'appel à cette interruption est décrit dans le chapitre II et se produit notamment lorsqu'on appuie simultanément sur les touches <CTRL–ALT–DEL> pour provoquer un redémarrage de la machine.

INT 19H	Pré-amorce

V.7.3. Gestion de l'heure et de la date (INT 1AH)

INT 1AH/0	Lecture de l'heure
	AH 0
	CX:DX temps écoulé depuis 0h en unités de 55 *ms*. AL nombre de fois où le compteur est passé à 0 depuis le dernier appel à cette fonction.
	BX, SI, DI, BP, CS, DS, SS, ES non modifiés

INT 1AH/1	Mise à l'heure
	AH 1 CX:DX temps écoulé depuis 0h en unités de 55 *ms*
	AX, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES non modifiés.

INT 1AH/2	Lecture de l'horloge (AT)
	AH 2
	CY=0 CH = heure CL = minutes DH = secondes CY=1 défaut de pile
	Données au format BCD BX, SI, DI, BP, CS, DS, SS, ES non modifiés

INT 1AH/3	Mise à jour de l'horloge (AT)
	AH 3 CH heure CL minutes DH secondes DL 0 (heure d'hiver) ou 1 (heure d'été)
	Données au format BCD BX, SI, DI, BP, CS, DS, SS, ES non modifiés

INT 1AH/4	Lecture de la date sur l'horloge (AT)
	AH 4
	CY=0 CH = siècle (19 ou 20) CL = année DH = mois DL = jour CY=1 défaut de pile

INT 1AH/5	Mise à jour de la date sur l'horloge (AT)
	AH 5 CH siècle (19 ou 20) CL année DH mois DL jour
	Données au format BCD BX, CX, SI, DI, BP, CS, DS, SS, ES non modifiés

INT 1AH/6	Fixer l'heure d'alarme (AT)
	AH 6 CH heure CL minutes DH secondes
	CY 1 : problème de piles ou une alarme a déjà été programmée (il faut la désactiver par la fonction 7)
	Les données sont au format BCD. Lorsque l'heure d'alarme est atteinte, un appel à l'interruption 70H est réalisé. Ensuite est effectué un INT 4AH qui est à la libre disposition des utilisateurs. BX, CX, SI, DI, BP, CS, DS, SS, ES non modifiés

INT 1AH/7	Désactiver une alarme (AT)
	AH 7
	BX, CX, SI, DI, BP, CS, DS, SS, ES non modifiés

V.7.4. La touche <CTRL BREAK> (INT 1BH)

Cette touche, lorsqu'elle est utilisée, appelle l'interruption 1BH à disposition des utilisateurs.

INT 1BH	Interruption <CTRL BREAK>

V.7.5. La fonction INT 1CH

A chaque *top* d'horloge (18,2 *ticks/s* = 65536/3600), l'interruption 1CH est appelée. Cette interruption est laissée à la disposition des utilisateurs.

INT 1CH	Traitement horloge utilisateur

V.7.6. Contrôle de disquette (INT 40H)

L'interruption 40H, utilisée pour le contrôle des disquettes, est destinée à remplacer INT 13H lorsque cette dernière est chargée de gérer les disques durs.

INT 40H	voir INT 13H

CHAPITRE VI

Les fonctions du DOS

Les fonctions compatibles XENIX sont indiquées par (*). Les fonctions non documentées dans le DOS sont décrites dans un cadre grisé. Les fonctions qui remplacent les appels documentés sont notées en caractères gras.

VI.1. Les fonctions de bas niveau

INT 20H	1 Fin d'exécution
	CS Segment PSP
	Cet appel est utilisé dans les programmes .COM développés dans des versions antérieures à 2.0 de MSDOS (utiliser plutôt fonction 4CH de INT 21H). Tous les *Handles* sont fermés. Tous les tampons disque sont sauvegardés sur disque. Les adresses rangées aux *offset* 0AH (*Terminate Address*), 0EH (traitement du <CTRL-C>) et 12H (*Critical Error*) sont récupérées par le programme *père*.

Les interruptions 25H et 26H permettent d'accéder aux secteurs d'un disque par un numéro logique. Le secteur de *boot* possède le numéro 0. A partir de la version 4.0 du DOS il y a un changement si les partitions sont de taille supérieure à 32 Mo. Dans ce cas CX doit contenir FFFFH et DS:BX l'adresse d'un bloc de données contenant :

offset	contenu
0-3	numéro du premier secteur
4-5	nombre de secteurs
6-9	pointeur sur le tampon de transfert

INT 25H	1 Lecture absolue disque	
	AL	numéro d'unité disque (0 pour A, 1 pour B, 2 pour C, etc …)
	DS:BX	adresse de transfert disque (DTA)
	CX	nombre de secteurs
	DX	numéro du premier secteur relatif (le premier est 0)

	AL code d'erreur si CF=1 flag CF indicateur d'erreur
	Tous les registres exceptés ceux de segments sont modifiés. !!! Le système sauvegarde les drapeaux sur la pile au moment de l'appel de cette fonction. Il est donc indispensable de faire un POPF après appel à cette fonction.

INT 26H	1 Ecriture absolue disque
	AL numéro d'unité disque DS:BX adresse de transfert disque (DTA) CX nombre de secteurs DX numéro du premier secteur relatif (le premier est 0)
	AL code d'erreur flag CF indicateur d'erreur
	Tous les registres sauf ceux de segments sont modifiés. !!! le système sauvegarde les drapeaux sur la pile au moment de l'appel de cette fonction. Il est donc indispensable de faire un POPF après appel à cette fonction.

INT 27H	1 Fin d'exécution en restant résident
	CS:DX Adresse du premier octet suivant le code du programme appelant.
	Cet appel est utilisé dans les programmes .COM développés dans des versions antérieures à 2.0 de MSDOS (voir fonction 31H de INT 21H).

INT 28H	Inoccupation DOS (→ **fonction 1680H** de **INT 2FH**)
	Keyboard Busy Loop Utilisé lorsqu'on est sous contrôle du clavier pour savoir si l'on peut faire appel au DOS (sert essentiellement lors de l'écriture de programme résidents faisant appel au DOS).

INT 2EH	
	DS:SI adresse d'une chaîne de commande
	Execute Command. La chaîne de commande est transmise à la partie résidente de COMMAND.COM. Tous les registres sont modifiés.

L'interruption 2FH dite *interruption multiplex* permet aux divers programmes résidents de fournir leurs services aux autres programmes (*TSR* et *device drivers*). Comme pour les autres interruptions, le registre AH doit contenir l'identificateur de la fonction demandée. Un certain nombre de numéros sont réservés à des utilitaires du DOS :

AH (hexa)	Utilitaire
1	PRINT.EXE
6	ASSIGN.COM
10	SHARE.EXE
11	(réseau)
14	NLSFUNC.EXE
1A	ANSI.SYS
43	HIMEM.SYS
48	DOSKEY.COM
4B	(gestion des tâches)
AD	KEYB.COM
AE	APPEND.EXE
B0	GRAFTABL.COM
B7	APPEND.EXE

Les adresses indiquées aux "adresses" INT 22H, INT 23H, INT 24H fournissent :

– INT 22H : donne l'adresse de fin de traitement. Cette adresse est copiée à l'*offset* 0AH du PSP lorsque celui-ci est créé.

– INT 23H : donne l'adresse de traitement du <CTRL-C>. Cette adresse est copiée à l'*offset* 0EH du PSP. Cette adresse de traitement est utilisée si l'on tape sur <CTRL-C> lors d'une entrée de caractères sur le clavier, ou pendant un affichage.

– INT 24H : donne l'adresse de traitement d'erreur *fatale*. Celle-ci est copiée à l'*offset* 12H du PSP. Cette interruption n'est pas utilisée par INT 25H et INT 26H. COMMAND.COM possède en effet son propre traitement (envoi du message : *Retry, Abort, Ignore ?*).

Lorsqu'une erreur se produit, les registres AX et DI contiennent une information concernant la cause de l'erreur. Si le *bit* 7 de AH est à 1 l'erreur est dûe à une image erronnée de la FAT ou une erreur sur un *character device*. Le bloc de données (*Device Header*) peut, dans ce dernier cas, être examiné à l'adresse BP:SI (le bit de poids fort de l'octet d'attribut indique de quel type d'unité il s'agit. On peut en déduire s'il y a erreur sur une FAT ou à cause d'une unité d'E/S de type "caractère").

Code erreur	Signification
0	Tentative d'écriture sur disque protégé en écriture.
1	Unité inconnue.
2	Lecteur pas prêt.
3	Commande inconnue.
4	Erreur sur les données.
5	Longueur de la structure de données de requête incorrecte.
6	Erreur de recherche (commande *Seek*).
7	Type de support inconnu.
8	Secteur non trouvé.
9	Plus de papier sur l'imprimante.
0AH	Erreur d'écriture.
0BH	Erreur en lecture.
0CH	Autre erreur (*General Failure*).

Contenu	Contenu de la pile utilisateur
IP, CS Drapeaux	Sauvegarde du contexte lors de l'appel à INT 24H.
AX, BX, CX, DX SI, DI, BP, DS, ES	Registres utilisateur sauvegardés par le DOS après appel au DOS par INT 21H.
IP, CS Drapeaux	Sauvegarde du contexte lors de l'appel au DOS par le programme utilisateur.

Si une instruction IRET est exécutée, MSDOS réagit en fonction de la valeur du registre AL : 0 (*Ignore*), 1 (*Retry*), 2 (*Terminate* à l'aide de INT 23H). Le traitement d'erreur doit s'abstenir de faire appel au DOS via INT 21H (les fonctions 1 à 0CH sont cependant utilisables). Si l'on désire effectuer ce traitement d'erreur, le programme doit "vider" la pile en y laissant le contexte permettant d'effectuer une instruction `IRET` ramenant au programme ayant appelé le DOS. L'état du DOS est dit *unstable*, jusqu'à ce qu'un appel à une fonction de numéro supérieur à 0CH soit effectué.

VI.2. Les entrées/sorties de caractères

INT 21H fonction 1	1.0 Lecture clavier avec écho (→ **3FH**)
	AH 1
	AL code caractère
	S'il s'agit de <CTRL C>, INT 23H est exécuté

INT 21H fonction 2	1.0 Affichage caractère (→ **40H**)
	AH 2 DL code caractère
	S'il s'agit de <CTRL C> INT 23H est exécuté.

INT 21H fonction 3	1.0 Lecture caractère sur entrée auxiliaire (→ **3FH**)
	AH 3
	AL code caractère
	S'il s'agit de <CTRL C> INT 23H est exécuté.

INT 21H fonction 4	1.0 Envoi caractère sur sortie auxiliaire (→ **40H**)
	AH 4 DL code caractère
	S'il s'agit de <CTRL C> INT 23H est exécuté.

INT 21H fonction 5	
	1.0 Envoi caractère sur imprimante (→ **40H**)
	AH 5 DL code caractère
	Si <CTRL C> sur la console, INT 23H est exécuté.

INT 21H fonction 6	
	1.0 Entrée/sortie directe sur console
	AH 6 DL – si FFH positionne le flag Z – si ≠ FFH le caractère dans DL est affiché
	ZF si DL valait FFH, ZF=0 signifie qu'un caractère a été tapé et son code est fourni par AL. Si ZF=1, aucun caractère n'a été tapé. AL code du caractère
	Si <CTRL C> sur la console, INT 23H est exécuté.

INT 21H fonction 7	
	1.0 Entrée directe sur console
	AH 7
	AL code du caractère
	Attente de caractère, sans d'écho ni test de <CTRL C>.

INT 21H fonction 8	
	1.0 Lecture clavier
	AH 8
	AL code du caractère
	Attente de caractère avec test de <CTRL C> et pas d'écho.

INT 21H fonction 9	
	1.0 Affichage chaîne de caractères (→ **40H**)
	AH 9 DS:DX adresse de la chaîne
	La chaîne doit se terminer avec un caractère "$".

INT 21H fonction 0AH	
	1.0 (→ **3FH**)
	AH 0AH DS:DX adresse du tampon
	Le <CTRL C> est traité et le tampon contient : – octet 1 : nombre max. de caractères dans le tampon (CR inclus), – octet 2 : nombre de caractères tapés (sans le CR), – puis les caractères.

INT 21H fonction 0BH		
	1.0	Test de l'état du clavier
	AH	0BH
	AL	FF s'il y a des caractères dans le *buffer* 0 si le *buffer* est vide
	<CTRL C> est traité.	

INT 21H fonction 0CH		
	1.0	Vide le *buffer* clavier et lecture clavier
	AH	0CH
	AL	fonction de lecture MSDOS 1, 6, 7, 8, A si AL prend une autre valeur le buffer est simplement vidé.
	AL	si 0, la valeur en entrée était ≠ 1, 6, 7, 8, A

VI.3. Les accès disque

INT 21H fonction 0DH		
	1.0	Mise à jour disques
	AH	0DH
	Cette fonction est utilisée pour s'assurer que les tampons disque et le contenu des disques sont bien identiques. Tous les tampons modifiés sont reécrits, mais les répertoires ne sont pas mis à jour (il faut fermer les fichiers pour que cela soit fait). Les programmes traitant <CTRL C> doivent utiliser cette fonction.	

INT 21H fonction 0EH		
	1.0	Sélection disque
	AH	0EH
	DL	numéro de disque (0 pour A, 1 pour B, …)
	AL	nombre de lecteurs logiques

INT 21H fonction 19H		
	1.0	Trouver le disque courant
	AH	19H
	AL	numéro du disque courant (0 pour A, 1 pour B, …)
	Au lancement d'un programme, l'adresse de DTA est fixée à l'adresse 80H du PSP.	

INT 21H fonction 1AH		
	1.0	Définir l'adresse de transfert disque
	AH	1AH
	DS:DX	adresse de transfert
	Les transferts ne peuvent s'étendre d'un segment à un autre.	

INT 21H fonction 1BH	2.0 Obtention informations disque courant (→ **36H**)
	AH 1BH
	AL nombre de secteurs par granule DS:BX adresse du descripteur de support (F8H, …) DX nombre de granules
	Les transferts ne peuvent s'étendre d'un segment à un autre. Dans la V.1, DS:BX donnait l'adresse de la copie de la FAT en mémoire.

INT 21H fonction 1CH	2.0 Obtention informations disque (→ **36H**)
	AH 1CH DL unité de disque (0 pour le lecteur courant, 1 pour A, …)
	AL nombre de secteurs par granule DS:BX adresse du descripteur de support DX nombre de granules
	Les transferts ne peuvent s'étendre d'un segment à un autre.

INT 21H fonction 1FH	5.0 Lecture DBP du disque par défaut
	AH 1FH
	AL 0FFH lecteur invalide ou erreur disque AL 0 DS:BX = adresse du tampon d'information disque

INT 21H fonction 2FH	2.0 Obtention de l'adresse de transfert disque
	AH 2FH
	ES:BX adresse de transfert disque

INT 21H fonction 32H	5.0 Lire adresse du tampon des paramètres disque
	AH 32H DL numéro de disque (0=défaut, 1=A, …)
	DS:BX adresse du bloc des paramètres disque

INT 21H fonction 36H	2.0 Obtention espace disque disponible
	AH 36H DL numéro de lecteur (0 pour le lecteur courant, 1 pour A, …)
	BX nombre de granules libres DX nombre de granules du support désigné CX nombre d'octets par secteur AX nombre de secteurs par granule ou 0FFFFH si le numéro de lecteur est invalide

Exemple d'utilisation des informations sur l'occupation du disque : on exécute les fonctions 1CH et 36H sur un disque dur dont on examine aussi le secteur de *boot*. On obtient les informations suivantes :

– fonction 1CH :

```
al = 4 (4 secteurs par granule)
DS:BX = 135:598 (pointeur vers l'information F8)
DX = 281D (10269) granules
```

– fonction 36H :

```
ax = 4 (4 secteurs par granule)
BX = 16F ((367) granules libres
CX = 200 octets par secteur
DX = 281D (10269) granules
```

– examen du secteur de boot :

```
EB 3C 90 4D 53 44 4F 53-35 2E 30 00 02 04 01 00
02 00 02 E7 A0 F8 29 00-11 00 18 00 11 00 00 00
…
```

Le secteur de boot indique qu'il y a A0E7 (41191_{10}) secteurs au total. Ce nombre de secteurs inclue le secteur de *boot*, les FATs' et le répertoire racine. Le nombre de secteurs utiles est de :

A0E7-1-(2*29)-20=A074 (41076_{10})

soit en nombre de 281D=10269_{10} granules de 4 secteurs.

VI.4. L'accès aux fichiers

INT 21H fonction 0FH	1.0 Ouverture fichier (→ **3DH**)
	AH 0FH DS:DX adresse d'un FCB non ouvert
	AL 0 si fichier trouvé FFH si fichier non trouvé
	– Si le numéro de lecteur est 0 cela indique le disque par défaut. – Le champ *bloc courant* est mis à 0. – La taille de l'enregistrement est mise à 128 par défaut. – La taille du fichier, les date et heure de dernière modification sont mises à jour dans le FCB.

INT 21H fonction 10H	1.0 Fermeture fichier (→ **3EH**)
	AH 10H DS:DX adresse d'un FCB ouvert
	AL 0 si fichier trouvé ou FFH si fichier non trouvé

INT 21H fonction 13H	1.0 Effacement de fichier (→ **41H**)

	AH 13H DS:DX adresse d'un FCB non ouvert
	AL 0 si fichier trouvé ou FFH si fichier non trouvé

INT 21H fonction 14H	1.0 Lecture séquentielle (→ **3FH**)
	AH 14H DS:DX adresse d'un FCB ouvert
	AL 0 lecture effectuée avec succès 1 fin de fichier atteinte 2 zone DTA trop petite 3 fin de fichier atteinte avec un enregistrement incomplet
	L'enregistrement défini par "numéro de bloc courant" et "numéro d'enregistrement courant" est chargé dans la zone de transfert DTA.

INT 21H fonction 15H	1.0 Ecriture séquentielle (→ **40H**)
	AH 15H DS:DX adresse d'un FCB ouvert
	AL 0 écriture effectuée avec succès 1 disque plein 2 zone DTA trop petite
	L'enregistrement "numéro de bloc courant"/"numéro d"enregistrement courant" est écrit à partir du contenu de la zone de transfert DTA.

INT 21H fonction 16H	1.0 Création fichier (→ **3CH**)
	AH 16H DS:DX adresse d'un FCB non ouvert
	AL 0 le répertoire est vide FFH pas de répertoire vide disponible

INT 21H fonction 17H	1.0 Renommer un fichier (→ **56H**)
	AH 17H DS:DX adresse du FCB modifié
	AL 0 fichier trouvé FFH fichier non trouvé ou existe déjà

INT 21H fonction 21H	1.0 Lecture en accès direct (→ **3FH**)
	AH 21H DS:DX adresse d'un FCB ouvert

	AL 0 lecture effectuée avec succès 1 fin de fichier atteinte 2 zone DTA trop petite 3 fin de fichier atteinte avec un enregistrement incomplet
	Le numéro de bloc courant et le numéro d'enregistrement courant sont mis à jour en accord avec le champ *numéro d'enregistrement relatif*.

INT 21H fonction 22H	1.0 Ecriture en accès direct (→ **40H**)
	AH 22H DS:DX adresse d'un FCB ouvert
	AL 0 écriture effectuée avec succès 1 disque plein 2 zone DTA trop petite
	Le numéro de bloc courant et le numéro d'enregistrement courant sont mis jour en accord avec le champ *numéro d'enregistrement relatif*.

INT 21H fonction 23H	1.0 Taille de fichier (→ **42H**)
	AH 23H DS:DX adresse d'un FCB non ouvert
	AL 0 fichier trouvé FFH fichier non trouvé
	Champ *numéro d'enregistrement relatif* = nombre d'enregistrements du fichier.

INT 21H fonction 24H	1.0 Fixe le numéro d'enregistrement relatif (→ **42H**)
	AH 24H DS:DX adresse d'un FCB non ouvert
	Le champ *numéro d'enregistrement relatif* est calculé à partir du numéro de bloc courant et d'enregistrement courant.

INT 21H fonction 27H	1.0 Lecture de blocs en accès direct (→ **3FH** et **42H**)
	AH 27H DS:DX adresse d'un FCB ouvert CX nombre de blocs à lire
	AL 0 lecture effectuée avec succès 1 fin de fichier atteinte 2 fin de segment atteinte 3 fin de fichier avec enregistrement partiel CX nombre de blocs effectivement lus
	Si AL = 3 l'enregistrement lu est complété avec des caractères NUL.

INT 21H fonction 28H	1.0 Ecriture de blocs en accès direct (→ **3FH** et **42H**)
	AH 28H DS:DX adresse d'un FCB ouvert CX nombre de blocs à lire
	AL 0 écriture effectuée avec succès 1 fin de fichier atteinte 2 fin de segment atteinte CX nombre de blocs effectivement écrits

INT 21H fonction 29H	1.0 Analyse nom de fichier
	AH 29H AL contrôle d'analyse *bit* 0 à 0 analyse arrêtée dès qu'on trouve un séparateur 1 les séparateurs sont ignorés *bit* 1 à 0 : si pas de lecteur, N° de lecteur mis à 0 dans le FCB 1 : si pas de lecteur, numéro inchangé dans le FCB *bit* 2 à 0 : si pas de nom de fichier, le nom de fichier n'est pas changé dans le FCB 1 : si pas de nom de fichier, on met 8 "espaces" dans le nom du fichier dans le FCB *bit* 3 à 0 : si pas d'extension, l'extension inchangée dans FCB 1 : si pas d'extension, on met 3 "espaces" à la place DS:SI adresse de la chaîne à analyser ES:DI adresse FCB non ouvert
	AL 0 pas de caractères de remplacement 1 existence de caractères de remplacement FFH identificateur non valide de disque DS:SI adresse du premier octet suivant la chaîne à analyser ES:DI adresse FCB non ouvert
	La chaîne est analysée pour trouver un nom de fichier. Un FCB non ouvert est créé si l'on en trouve un.

VI.5. Les fonctions de l'horloge/calendrier

INT 21H fonction 2AH	1 Obtention date
	AH 2AH
	CX année (1980-2099) DH mois (1-12) DL jour (1-31) AL jour de la semaine (0 pour dimanche, …, 6 pour samedi)

INT 21H fonction 2BH	1	Définition date
	AH	2BH
	CX	année (1980-2099)
	DH	mois (1-12)
	DL	jour (1-31)
	AL	0 pour date valide 0FFH pour date invalide

INT 21H fonction 2CH	1	Obtention heure
	AH	2CH
	CH	heure (0-23)
	CL	minutes (0-59)
	DH	secondes (0-59)
	DL	centièmes de seconde (0-99)

INT 21H fonction 2DH	1	Définition heure
	AH	2DH
	CH	heure (0-23)
	CL	minutes (0-59)
	DH	secondes (0-59)
	DL	centièmes de seconde (0-99)
	AL	0 pour heure valide 0FFH pour heure invalide

VI.6. Accès aux indicateurs du DOS

Un certain nombre de fonctions d'accès aux variables internes du DOS ne sont pas documentées. La seule garantie que l'on a repose sur le nombre et l'importance des éditeurs de logiciels qui ont pu les utiliser. Nous avons déjà cité le cas de la gestion de l'espace mémoire à l'aide d'une liste chaînée.

INT 21H fonction 2EH	1.0	Définition du drapeau de vérification
	AH	2EH
	AL	0 pas de vérification 1 vérification

INT 21H fonction 3300H	2.0	Lecture du drapeau de contrôle de <CTRL C>
	AH	3300H
	DL	état du test : 0 si OFF, 1 si ON
	Normalement le test de <CTRL C> n'est effectué que sur les fonctions 1 à 0CH. La fonction 33H permet d'étendre ce test aux autres fonctions.	

INT 21H fonction 3301H		
	2.0	Initialisation du drapeau de vérification
	AX	3301H
	DL	0 test inhibé 1 test validé
	Normalement le test de <CTRL C> n'est effectué que sur les fonctions 1 à 0CH. La fonction 33H permet d'étendre ce test aux autres fonctions.	

INT 21H fonction 3305H		
	2.0	Numéro du disque de démarrage
	AX	3305H
	DL	numéro de disque de lancement (1=A, 2=B, …)

INT 21H fonction 3306H		
	5.0	Lire version DOS
	AX	3306H
	BL	Code majeur
	BH	Code mineur
	DL	Numéro de version de révision (3 bits de poids faible)
	DH	Drapeaux (8 pour MSDOS en ROM, 10H pour MSDOS en HMA)

INT 21H fonction 3701H	
AX	3701H
Set Switch-AR	

INT 21H fonction 52H		
		Lire adresse de la table des adresses des listes chaînées
	AH	52H
	ES:BX	adresse table des adresses des listes chaînées (MCB, DD, …)
	Les fonction SUBST et JOIN, notamment, utilisent cette fonction. Les pointeurs permettent d'accéder aux tables : – *Memory Control Blocks*, – Chaîne des *Device Drivers*, – Structure de répertoire courant (*Current Directory Structure*), – *System File Table*, …	

INT 21H fonction 53H	
AH	53H

	Translate BIOS Parameters Block

INT 21H fonction 54H	2 Donne l'état du drapeau de vérification
	AH 54H
	AL drapeau de vérification

INT 21H fonction 55H	
	AH 55H
	Create New PSP

INT 21H fonction 5D00H	
	AX 5D00H
	Server Function Call

INT 21H fonction 5D06H	
	AX 5D06H
	Get DOS Swappable Disk Area

INT 21H fonction 5D0AH	
	AX 5D0AH
	Set Extended-Error Information

VI.7. Le contrôle de fin d'exécution de programme

INT 21H fonction 0	1.0 Fin d'exécution (→ **4CH**)
	AH 0 CS segment du PSP courant
	Cet appel est équivalent à INT 20H. Tous les fichiers dont la taille a changé doivent être fermés avant d'appeler cette fonction.

INT 21H fonction 31H	2.0 Conserve le processus courant en mémoire

AH	31H
AL	code de sortie récupérable par le processus père
DX	taille mémoire à réserver en nombre de paragraphes

INT 21H fonction 34H

2.0	Lire adresse du drapeau InDOS
AH	34H
DS:BX	adresse d'un "sémaphore" (octet)
Il s'agit plus d'un compteur que d'un sémaphore. Lorsque ce drapeau est différent de 0, MSDOS est en train d'exécuter une fonction.	

INT 21H fonction 4B00H

2.0	Charge et exécute un programme (*)
AX	4B00H
DS:DX	adresse "chemin d'accès/nom de programme" (terminée par 0)
ES:BX	adresse d'un bloc de paramètres
AL	0 chargement et exécution de programme
CY=1	AX = 1 fonction incorrecte AX = 10 environnement incorrect AX = 11 format incorrect AX = 2 fichier non trouvé AX = 8 taille mémoire insuffisante
Les *handles* des fichiers ouverts sont accessibles dans le processus fils (à moins de le demander explicitement à l'ouverture du fichier-fonction 3DH). Le processus père peut garder tout contrôle sur ceux-ci.	

En version 2.X, le système ne sauvegarde que les registres CS et IP. Les fichiers désignés sont de type .EXE ou .COM. Pour un fichier de type .BAT ou pour exécuter des commandes internes, il faut appeler COMMAND.COM par la chaîne de commande (terminée par un <retour chariot>) :

```
'COMMAND.COM /c <nom_de_fichier.BAT>',13 ou
'COMMAND.COM /c <commande_interne>',13
```

INT 21H fonction 4B01H

2.0	Charge un programme en mémoire
AX	4B01H
Le programme chargé n'est pas lancé.	

INT 21H fonction 4B03H

2.0	Charge *Overlay*
AX	4B03H
DS:DX	adresse nom de programme (terminé par NUL)
ES:BX	adresse d'une structure de chargement d'*overlay*
CY=1	AX conteint un code d'erreur (1 à 5, 8 ou A)

Le programme chargé n'est pas lancé. La structure d'*overlay* contient : – mot : segment de l'*overlay* – mot : facteur de relocation

INT 21H fonction 4B05H		
	5.0	Prépare un programme pour exécution
	AX	4B05H
	DS:DX	adresse d'une structure d'exécution
	Cette fonction prépare en particulier le numéro de version pour SETVER. La structure d'exécution contient : – mot réservé – mot pour drapeaux de type – adresse du nom du programme (terminé par NUL) – segment du PSP du nouveau programme – CS:IP du nouveau programme – double mot : taille du programme, PSP compris Cette fonction désactive la ligne A20. Si ce programme, qui est utilisé lorsqu'on déroute la fonction 'charge et exécute un programme', utilise la HMA, il doit revalider la ligne A20.	

INT 21H fonction 4CH		
	2	Fin d'exécution de processus (*)
	AH	4CH
	AL	code de retour exploitable par le processus père (fonction 04DH)

VI.8. L'accès aux vecteurs d'interruption

INT 21H fonction 25H		
	1.0	Définition adresse de traitement d'interruption
	AH	25H
	AL	numéro d'interruption
	DS:DX	adresse du programme de traitement

INT 21H fonction 35H		
	2.0	Récupération d'adresse de traitement d'interruption
	AH	35H
	AL	numéro d'interruption
	ES:BX	adresse du programme de traitement de l'interruption

VI.9. Fonctions d'accès aux répertoires

INT 21H fonction 39H		
	2.0	Création sous-répertoire (*)

	AH	39H
	DS:DX	adresse chaîne de caractères (terminée par NUL) du chemin d'accès
	CY=1	AX=3 chemin non valide
		AX=5 le répertoire ne peut être créé

INT 21H fonction 3AH	2.0	Effacement de répertoire (*)
	AH	3AH
	DS:DX	adresse du chemin d'accès (terminé par NUL)
	CY=1	AX=3 chemin non valide
		AX=5 le répertoire ne peut être créé
		AX=16 répertoire courant

INT 21H fonction 3BH	2.0	Changement de répertoire (*)
	AH	3BH
	DS:DX	adresse chaîne de caractères (terminée par NUL) du chemin d'accès
	CY=1	AX=3 chemin non valide

INT 21H fonction 47H	2.0	Fournit le répertoire courant
	AH	47H
	DS:SI	adresse d'une zone mémoire de 64 octets
	DL	numéro de lecteur (1 pour A, 2 pour B, …)
	CY=1	AX = 15 numéro de lecteur non correct

VI.10. Accès aux fichiers par numéros logiques

La méthode d'accès aux fichiers utilisant des numéros logiques devra être préférée à celle utilisant les blocs de contrôle de fichiers (*FCB*).

INT 21H fonction 3CH	2.0	Création de fichier (*)
	AH	3CH
	DS:DX	adresse de la chaîne (terminée par NUL) du chemin d'accès
	CX	bit 0 à 1 : lecture seulement
		bit 1 à 1 : fichier caché
		bit 2 à 1 : fichier système
	CY=1	AX=3 chemin non valide
		AX=4 trop de fichiers ouverts
		AX=5 le fichier ne peut être créé
	CY=0	AX=numéro logique (*handle*)
	Si le fichier existait déjà, il est écrasé, sinon il est créé.	

INT 21H fonction 3DH

2.0	Ouverture de fichier (*handle*) (*)
AH	3DH
AL	*bits* 0-2 autorisation en lecture/écriture : – 000 *read Only* – 001 *write Only* – 010 *read/Write* *bits* 3 à 0 *bits* 4-6 : – 000 seul le prog. actuel peut accéder au fichier en mode FCB – 001 seul le programme actuel peut accéder au fichier – 010 un autre prog. peut accéder au fichier en lecture seulement – 011 un autre prog. peut accéder au fichier en écriture seulement – 100 un autre prog. peut accéder au fichier bit 7 0 le prog. fils peut accéder au fichier 1 seul le programme actuel peut accéder au fichier
DS:DX	adresse du nom de fichier (terminé par un octet à 0)
CY=1	AX=1 pas de logiciel de partage de fichier AX=2 fichier non trouvé AX=3 chemin non trouvé ou fichier n'existe pas AX=4 trop de fichiers ouverts AX=5 accès refusé (violation de type ou de protection) AX=12 accès non valide
CY=0	AX=numéro logique (*handle*)

INT 21H fonction 3EH

2.0	Fermeture de fichier (*handle*)
AH	3EH
BX	numéro logique (*handle*) de fichier
CY=1	AX=6 numéro non valide

INT 21H fonction 3FH

2.0	Lecture généralisée sur fichier (*)
AH	3FH
DS:DX	adresse d'un tampon
CX	nombre d'octets à lire
BX	numéro logique du fichier ou périphérique (*handle*)
CY=1	AX = 5 accès refusé AX = 6 numéro logique invalide
CY=0	AX = nombre d'octets lus

INT 21H fonction 40H

2.0	Ecriture généralisée sur fichier (*)

	AH	40H
	DS:DX	adresse d'un tampon
	CX	nombre d'octets à écrire
	BX	numéro logique du fichier ou périphérique (*handle*)
	CY=1	AX = 5 accès refusé
		AX = 6 numéro logique invalide
	CY=0	AX = nombre d'octets écrits

INT 21H fonction 41H	2.0	Effacement fichier (*)
	AH	41H
	DS:DX	adresse de la chaîne "nom de fichier" (le caractère de fin a pour code 0)
	CY=1	AX = 5 accès refusé
		AX = 2 fichier non trouvé

INT 21H fonction 42H	2.0	Modification de pointeur dans un fichier (*)
	AH	42H
	CX-DX	déplacement en octets (sur 32 bits)
	AL	0 déplacement à partir du début du fichier
		1 déplacement à partir de la position courante
		2 déplacement à partir de la fin du fichier
	BX	numéro logique (*handle*)
	CY=1	AX = 6 numéro logique invalide
		AX = 1 fonction (AL) non valide
	CY=0	DX-AX nouveau pointeur
	Lorsque AL est égal à 0, on a un mode d'accès direct. Le numéro du premier octet du fichier est 0.	

INT 21H fonction 45H	2.0	Duplication de pointeur sur fichier (*)
	AH	45H
	BX	numéro logique (*handle*)
	CY=1	AX = 6 numéro logique non correct
		AX = 4 trop de fichiers ouverts
	CY=0	AX = nouveau numéro logique

INT 21H fonction 46H	2.0	Duplication forcée de pointeur sur fichier (*)
	AH	46H
	BX	numéro logique (*handle*)
	CX	nouveau numéro logique
	CY=1	AX = 6 numéro logique non correct
		AX = 4 trop de fichiers ouverts

INT 21H fonction 56H	2.0 Renomme dans un autre répertoire
	AH 56H DS:DX adresse d'une chaîne "chemin d'accès" valide ES:DI adresse d'une chaîne "nouveau chemin d'accès"
	CY=1 AX = 2 fichier non trouvé AX = 17 support différent AX = 5 accès refusé
	Le support doit être le même

INT 21H fonction 5AH	3.0 Création de fichier temporaire
	AH 5AH CX attribut fichier temporaire DS:DX adresse répertoire
	CY=0 AX=*handle* DS:DX adresse nom de fichier complet CY=1 AX=code erreur (3 = chemin non trouvé, 5 = accès refusé)

INT 21H fonction 67H	3.3 Augmente le nombre de *Handles*
	AH 67H
	Increase File Handle (de 20 à 65535).

INT 21H fonction 68H	3.3 Sauve les *Buffers* sur disque
	AH 68H BX *Handle*
	CY=1 AX = code erreur
	Force l'écriture des tampons dans les fichiers correspondants.

INT 21H fonction 6CH	4.0 Création, ouverture et mise à jour
	AX 6C00H BX mode d'accès *bits* 0-2 000: fichier *Read-Only* 001: fichier *Write-Only* 010: fichier accessible en lecture/écriture *bits* 3 0 *bits* 4-6 000: accès réservé au process courant 001: accès réservé au process courant 010: accès aurorisé en lecture aux autres programmes 011: accès aurorisé en écriture aux autres programmes 100: accès aurorisé en lecture/écriture aux autres programmes

	bits 7 0: les programmes enfants peuvent accéder au fichier *bits* 8-12 0 *bits* 13 0: si erreur appeler INT 24H 1: si erreur sortir avec CY et AX *bits* 14 1: mise à jour répertoire à chaque écriture *bits* 15 0
	CX attribut (*bits* 0:*Read-only*, 1:caché, 2:fichier système, 5:archive)
	DX mode opératoire du DOS (DL=xxxxyyyy) yyyy=0000 sortir avec code erreur si fichier n'existe pas encore yyyy=0001 créer le fichier xxxx=0000 sortir avec code erreur si fichier existe déjà xxxx=0001 ouvrir le fichier existant
	DS:SI adresse nom du fichier (terminé par 0)
	CY=1 AX = code erreur (1, 2, 3, 4, 5 ou 12) CY=0 AX = *handle* et CX état
	Extended Create/Open file

VI.11. L'accès aux informations sur les fichiers

INT 21H fonction 4300H	2.0 Lecture attributs fichier (*)
	AX 4300H DS:DX adresse chaîne "nom de fichier" (terminée par un octet à 0) CX attributs du fichier (si AL = 1)
	CY=1 AX = 3 chemin d'accès incorrect AX = 1 fonction (AL) invalide AX = 2 fichier non trouvé AX = 5 accès refusé CY=0 CX = attributs si fonction (AL) = 0

INT 21H fonction 4301H	2.0 Modification attributs fichier (*)
	AX 4301H DS:DX adresse chaîne "nom de fichier" (terminée par un octet à 0) CX attributs du fichier (si AL = 1)
	CY=1 AX = 3 chemin d'accès incorrect AX = 1 fonction (AL) invalide AX = 2 fichier non trouvé AX = 5 accès refusé

INT 21H fonction 5700H	2.0 Lire date/heure fichier
	AX 5700H BX numéro logique

CY=1	AX = 1 fonction incorrecte AX = 6 numéro logique incorrect
CY=0	CX = heure (sous-fonction 0) DX = date (sous-fonction 0)

INT 21H fonction 5701H	
2.0	Ecrire date/heure fichier
AX 5701H BX=numéro logique, CX=heure, DX=date	
CY=1	AX = 1 fonction incorrecte AX = 6 numéro logique incorrect

VI.12. L'accès aux *Device Drivers*

INT 21H fonction 44H	
2.0	Contrôle sur entrées/sorties généralisées (*)
AH 44H AL=fonction – 0 lecture d'informations sur unité d'entrée/sortie – 1 mise à jour informations sur unité d'entrée/sortie – 2 lecture de CX caractères sur unité d'entrée/sortie – 3 écriture de CX caractères sur unité d'entrée/sortie – 4 lecture de CX caractères sur unité de disque (unité BL) – 5 écriture de CX caractères sur unité de disque (unité BL) – 6 lecture d'état en lecture – 7 lecture d'état en écriture BX numéro logique (*handle*) BL numéro de lecteur (si AL=4,5) – 0 pour lecteur courant, – 1 pour A, 2 pour B, … DS:DX adresse des données ou d'un tampon (fonctions 2, 3, 4, 5) DX informations sur unité d'entrée/sortie (fonction 1) CX nombre d'octets à lire ou écrire	
CY=1	AX = 1 fonction (AL) invalide AX = 5 accès refusé AX = 13 données incorrectes AX = 6 numéro logique incorrect
CY=0	AX = nombre d'octets transmis (fonctions 2, 3, 4, 5) AL = 0 (fonctions 6 et 7) si unité ou fichier prêt AL = 0FFH (6 et 7) si unité ou fichier non prêt DX informations sur unité d'entrée/sortie (fonction 0)

INT 21H fonction 44H	
3.0	Contrôle sur entrées/sorties généralisées (*)

AH	44H
AL	– 8 test d'amovabilité support – 0BH fixer le nombre de tentatives d'accès réseau
BX	numéro logique (*handle*) (si AL=0AH) nombre de répétitions (si AL=0BH)
BL	numéro de lecteur (si AL=8) – 0 pour lecter courant, – 1 pour A, 2 pour B, …
CX	délai entre deux tentatives d'accès (si AL=0BH)
CY=1	AX = 1 fonction (AL) invalide AX = 15 périphérique inconnu AX = 6 numéro logique incorrect
CY=0	AX = 0 support non amovible (fonctions 8) AX = 1 support amovible (fonctions 8) – *bit* 12 = 0 périphérique local – *bit* 12 = 0 périphérique distant

INT 21H
fonction 44H

3.1	Contrôle sur entrées/sorties généralisées (*)
AH	44H
AL	code fonction – 9 test de périphérique distant – 0AH test *handle* distant
BX	numéro logique (*handle*) (si AL=0AH)
BL	numéro de lecteur (si AL=9), 0lecteur courant, 1=A, 2=B, …
CY=1	AX = 1 fonction (AL) invalide AX = 15 périphérique inconnu AX = 6 numéro logique incorrect
CY=0	DX = attribut du périphérique (fonction 9) – *bit* 12 = 0 périphérique local – *bit* 12 = 0 périphérique distant DX = code IOCTL (fonction 0AH) – *bit* 15 = 0 périphérique local – *bit* 15 = 0 périphérique distant

Constitution de DX pour les sous-fonctions 0 et 1 :

bit	
0	1 si l'unité est la console (en entrée)
1	1 si l'unité est la console (en sortie)
2	1 si l'unité est NUL
3	1 si l'unité est l'unité d'horloge
4	1 si l'unité est une unité spéciale
5	1 si l'unité est une unité en mode "caractères" 0 si l'unité est en mode "blocs"

6	0 si l'unité possède une signalisation de fin de fichier
7	1 si l'unité est une unité d'entrée/sortie (bits 0 à 6 utilisés) 0 si l'unité est un fichier disque (les bits 8 à 15 sont à 0)
8 à 13	réservés
14	0 l'unité ne traite pas les chaînes de contrôle lors du traitement des sous-fonctions 2 et 3 1 si l'unité peut traîter des chaînes de contrôle (AL=2 ou 3)
15	réservé

INT 21H fonction 440CH	IOCTL générique (périphériques caractères)
	CL 45H (V3.3) initialise compteur d'itérations 4AH (V3.3) sélectionne code de page 4CH (V3.3) début préparation de code de page 4DH (V3.3) fin préparation de code de page 5FH (V4.0) initialise mode d'affichage 65H (V3.3) lit compteur d'itérations 6AH (V3.3) lit code de page sélectionné 6BH (V3.3) demande liste des codes de page 7FH (V4.0) lit le mode d'affichage

INT 21H fonction 440DH	IOCTL générique (périphériques blocs)
	CL 40H (V3.2) initialise paramètres de périphériques 41H (V3.2) écrit piste sur lecteur logique 42H (V3.2) formatte piste sur lecteur logique 46H (V4.0) initialise identificateur de support 60H (V3.2) lit paramètres de périphériques 61H (V3.2) lit piste sur disque logique 62H (V3.2) vérifie piste sur disque logique 66H (V4.0) lit identificateur de support 68H (V5.0) lit le type de support

INT 21H fonction 440EH	3.2 Renvoie correspondance des lecteurs logiques
	BL numéro de lecteur (0=lecteur par défaut, 1=A, …)
	CY=0 AL=numéro du lecteur actif du lecteur physique correspondant CY=1 numéro d'erreur dans AX : 0001 numéro de fonction erroné 0005 accès refusé 000F numéro de lecteur erroné

INT 21H fonction 440FH	3.2 Initialise correspondance des lecteurs logiques
	BL numéro de lecteur (0=lecteur par défaut, 1=A, …)
	CY=0 AL=numéro du lecteur actif du lecteur physique correspondant CY=1 numéro d'erreur dans AX : 0001 numéro de fonction erroné 0005 accès refusé 000F numéro de lecteur erroné

INT 21H fonction 4410H	5.0 Vérification fonction *Handle* IOCTL
	BX *handle de périphérique* CH catégorie de périphérique 01 ligne série 03 écran 05 sortie parallèle CL fonction à vérifier (45H ou 65H)
	CY=1 AX=0001 numéro de fonction erroné AX=0005 accès refusé
	Cette fonction permet de savoir si une fonction IOCTL est bien mise en œuvre par le *driver* de périphérique indiqué par *handle*.

INT 21H fonction 4411H	5.0 Vérification fonction périphérique IOCTL
	BL numéro de lecteur (0=lecteur par défaut, 1=A, …) CH 8 (catégorie) CL à vérifier : (40H,41H,42H,46H,60H,61H,62H,66H,68H)
	CY=1 AX = 1 numéro de fonction erroné AX = 5 accès refusé AX = F numéro de lecteur erroné
	Cette fonction permet de savoir si une fonction IOCTL est bien mise en œuvre pour le lecteur indiqué.

VI.13. Les fonctions de gestion de la mémoire

INT 21H fonction 48H	2 Allocation de mémoire
	AH 48H BX taille mémoire requise (en paragraphes)
	CY=1 AX = 8 pas assez de mémoire AX = 7 violation d'espace mémoire BX = taille mémoire que l'on peut allouer CY=0 AX:0 adresse de la mémoire allouée

INT 21H fonction 49H	2	Libération d'espace mémoire
	AH 49H BX nombre de paragraphes à libérer ES segment de la zone à libérer	
	CY=1	AX = 9 le segment donné dans ES n'est pas correct AX = 7 violation d'espace mémoire
	Libération de ce qui a été alloué par la fonction 48H.	

INT 21H fonction 4AH	2	Modification d'allocation de mémoire
	AH 4AH ES segment de la zone BX taille mémoire requise (en paragraphes)	
	CY=1	AX = 9 le segment donné dans ES n'est pas correct AX = 7 violation d'espace mémoire AX = 8 pas assez de mémoire BX = taille mémoire qui peut être allouée

INT 21H fonction 5800H	3.0	Lire principe de répartition mémoire
	AX 5800H	
	CY=1	AX=1 code fonction inconnu
	CY=0	AX=0 recherche en partant des adresses basses AX=1 recherche selon meilleure méthode AX=2 recherche en partant des adresses hautes

INT 21H fonction 5801H	3.0	Fixer principe de répartition mémoire
	AX 5801H BX principe :	– 0 en partant des adresses basses – 1 bloc de taille la plus proche – 2 en partant des adresses hautes – 80H premier bloc en mémoire haute (sinon la recherche s'effectue en mémoire normale) – 81H bloc de taille la plus proche en mémoire haute … – 82H dernier bloc en mémoire haute … – 40H premier bloc en mémoire haute – 41H bloc de taille la plus proche en mémoire haute – 42H dernier bloc en mémoire haute

INT 21H fonction 5802H	5.0	Lire lien vers mémoire haute

	AX 5802H
	AL 1 si lien, 0 sinon
	Permet de savoir si un prog. peut allouer des blocs en mémoire haute.

INT 21H fonction 5803H	5.0 Initialise lien vers mémoire haute
	AX 5803H BX 1 si lien, 0 sinon
	CY=1 AX=1 fonction non valide (si DOS=UMB n'a pas été précisée) AX = 7 zone mémoire corrompue

VI.14. Fonctions diverses

INT 21H fonction 30H	2.0 Version DOS
	AH 30H
	AL numéro principal de version AH numéro secondaire de version

INT 21H fonction 38H	2.0 Lire informations liées au pays
	AH 38H DS:DX adresse d'un tampon de 32 octets au moins AL 0 lecture paramètres pour pays courant, ≠ 0 code pays
	si DX=0FFFFH change le code courant par le code donné dans AL

INT 21H fonction 38H	3 Déterminer le pays
	AH 38H DS:DX adresse d'un tampon de 32 octets au moins (si DX=0FFFFH la fonction change le code courant par lecode donné dans AL) AL 1-254 numéro de pays 255 : alors BX doit contenir le code pays
	CY=1 code pays incorrect

Zone de données d'informations sur le pays (les chaînes de caractères (CC) sont toutes terminées par le caractère NUL) :

Déplacement	
0-1	format pour la date et l'heure
2-6	CC symbole monétaire
7-8	CC séparateur pour les milliers
9-10	CC séparateur décimal
11-12	CC séparateur pour la date

13-14	CC séparateur pour l'heure
15	0 symbole monétaire précède le montant sans espace 1 symbole monétaire suit le montant sans espace 2 symbole monétaire précéde le montant avec un espace 3 symbole monétaire suit le montant avec un espace
16	nombre de décimales dans la représentation monétaire
17	format pour l'heure (0-1 /1-24)
18-21	adresse de traitement pour les caractères de code ≥ 80H (particularités nationales : conversion minuscules --> majuscules)
22-23	CC séparateur de listes de données

Le bloc de paramètres a le format suivant :

octet	Contenu (sous fonction 0)
0-1	segment de l'environnement
2-5	adresse complète de la ligne de commande (80H dans le PSP)
6-9	adresse complète du FCB (5CH dans le PSP)
0AH-0DH	adresse complète du FCB (6CH dans le PSP)
	Lorsque l'adresse de l'environnement est 0, le processus fils hérite de l'environnement du processus père

octet	Contenu (sous fonction 3)
0-1	segment de chargement du fichier
2-3	facteur de réimplantation appliqué à l'"image"

INT 21H fonction 4DH	2.0 Récupération état d'un processus fils (*)
	AH 4DH
	AL code de sortie AH 0 fin sur "abort" 1 fin sur <CTRL C> 2 fin sur erreur matérielle 3 fin mais reste résident

INT 21H fonction 6501H	3.3 Lire informations étendues sur pays
	AX 6501H BX code de page (si 0FFFH pays courant) CX taille du tampon DX code du pays ES:DI adresse d'une structure d'information sur le pays
	CY=1 AX = code d'erreur 1 ou 2

INT 21H fonction 6502H	3.3 Lire table des majuscules

AX	6502H
BX	code de page (si 0FFFH pays courant)
CX=5	taille du lien du tampon
DX	code de pays
ES:DI	adresse du tampon
CY=1	AX = code d'erreur (1 ou 2)
Cette table fait correspondre les codes > 128 à leur équivalents en majuscule. Le lien de tampon contient : – octet d'identification (=2) – adresse (4 octets) de la table	

INT 21H fonction 6504H

3.3	Lire table des majuscules pour noms de fichiers
AX	6504H
BX	code de page (si 0FFFH pays courant)
CX=5	taille du lien du tampon
DX	code de pays
ES:DI	adresse du tampon
CY=1	AX = code d'erreur (1 ou 2)

INT 21H fonction 6505H

3.3	Lire table des caractères pour noms de fichiers
AX	6505H
BX	code de page (si 0FFFH pays courant)
CX=5	taille du lien du tampon
DX	code de pays
ES:DI	adresse du tampon
CY=1	AX = code d'erreur (1 ou 2)

INT 21H fonction 6506H

3.3	Lire table de correspondance
AX	6506H
BX	code de page (si 0FFFH pays courant)
CX=5	taille du lien du tampon
DX	code de pays
ES:DI	adresse du tampon
CY=1	AX = code d'erreur (1 ou 2)
La table contient 256 éléments indiquant l'ordre alphabétique.	

INT 21H fonction 6507H

3.3	Lire table des caractères (sur deux octets)
AX	6507H
BX	code de page (si 0FFFH pays courant)
CX=5	taille du lien du tampon
DX	code de pays
ES:DI	adresse du tampon

	CY=1 AX = code d'erreur (1 ou 2)

INT 21H fonction 6520H	3.3 Conversion en majuscules
	AX 6520H DL code du caractère
	DL caractère converti en utilisant la table des majuscules

INT 21H fonction 6521H	3.3 Conversion de chaîne
	AX 6521H CX longueur de la chaîne DS:DX adresse de la chaîne

INT 21H fonction 6522H	3.3 Conversion de chaîne ASCIIZ
	AX 6522H DS:DX adresse de la chaîne

INT 21H fonction 6601H	3.3 Lire code de page global
	AX 6601H
	BX code de page utilisateur DX code de page système
	CY=1 AX=1

INT 21H fonction 6602H	3.3 Initialise code de page global
	AX 6602H BX code de page

VI.15. L'accès aux répertoires

INT 21H fonction 11H	1.0 Recherche d'un fichier dans le répertoire (recherche par FCB, première occurence) (→ **4EH**)
	AH 11H DS:DX adresse d'un FCB non ouvert
	AL 0 si fichier trouvé FFH si fichier non trouvé

INT 21H fonction 12H	1.0 Recherche de la prochaine occurence dans le répertoire (→ **4FH**) (recherche par FCB)

AH	12H
DS:DX	adresse d'un FCB non ouvert
AL	0 si fichier trouvé FFH si fichier non trouvé
Cette fonction est appelée après la fonction 11H	

INT 21H fonction 4EH	
2	Recherche de fichier d'un attribut donné (première occurence)
AH	4EH
DS:DX	adresse de chaîne "chemin d'accès"
CX	attribut de recherche : – 0 recherche des fichiers "normaux" au sens du DOS – ≠0 recherche de tous les fichiers dont l'attribut est donné.
CY=1	AX = 2 chemin non trouvé AX = 18 fichier non trouvé
Cette fonction accepte un chemin d'accès contenant des *wildcards*. A l'adresse de transfert, on trouve un bloc d'information (tableau ci-après). L'adresse de transfert doit avoir été définie au préalable (fonction 1AH).	

octet	Contenu du bloc d'information
0-20	réservés
21	attribut trouvé
22-23	heure
24-25	date
26-27/28-29	taille (poids faible) / (poids fort)
30-42	nom

INT 21H fonction 4FH	
2.0	Suite de la recherche
AH	4FH
CY=1	AX = 18 plus de fichiers
Cette fonction fait suite à la fonction 4EH lorsqu'on a des caractères de remplacement dans la nom du fichier. La DTA ne doit pas être modifiée entre le premier appel à la fonction 4EH et les appels à la fonction 4FH.	

VI.16. L'accès au préfixe de segment programme

INT 21H fonction 26H	
1.0	Création de PSP (→ **4B00H**)
AH	26H
DS	adresse de segment du nouveau PSP

INT 21H fonction 50H	
2.0	Définition d'une adresse de PSP
AH	50H

Set PSP Address (accompagné du changement de la DTA, on peut éventuellement commuter d'une tâche à l'autre).

INT 21H fonction 51H	
	2.0 Lecture adresse de PSP
	AH 51H
	Get PSP address (remplace la fonction 62H)

INT 21H fonction 59H	
	3.0 Lire code étendu d'erreur
	AH 59H BX 0
	AX erreur BH cause erreur BL conseil CH origine erreur

INT 21H fonction 5D0AH	
	4.0 Etablit les classes d'erreur
	AX 5D0AH DS:SI adresse d'une structure d'erreur
	La structure d'erreur contient : – 8 mots pour les registres AX,BX,CX,DX,SI,DI,DS,ES, – mot réservé – mot : identificateur utilisateur – mot : identificateur de processus

VI.17. Complément sur les fonctions de gestion de réseau

NetBios (*NETwork Basic IO System*) constitue une couche logicielle qui vient intercepter les appels standard au DOS. Ce logiciel met en œuvre (à peu près …) les couches 2 à 5 du modèle ISO (le niveau *liaison* ne respecte pas la normalisation ISO car il n'y a pas de contrôle d'erreur, lequel est laissé à la charge de la couche *transport*).

INT	AH	AL	fonction
21H	3DH		ouverture de fichier avec attributs de partage
	44H	9	IOCTL, unité d'E/S redirigée ?
		0AH	IOCTL, numéro logique local ou distant ?
		0BH	IOCTL, modification du compteur d'accès partagé
	59H		lecture code d'erreur étendu
	5AH		création de fichier temporaire
	5BH		création de nouveau fichier

21H	5CH	0	blocage
		1	déblocage
	5EH	0	lecture nom de machine
		2	définition de chaîne de contrôle imprimante
	5FH	2	lecture adresse de liste d'assignation
		3	redirection unité d'E/S vers le réseau
		4	annulation de redirection
2AH	0		test d'installation NETBIOS
	3		lecture d'état d'unité partagée
	4		exécution NETBIOS
	5		lecture des informations ressources réseau
	6		contrôle d'impression sur le réseau
2FH	87	0	ajoût sur test d'installation
		2	ajoût sur test de version
	88	0	test de commande réseau
		3	lecture d'adresse du serveur de messagerie
		4	écriture d'adresse du serveur de messagerie
		9	test de version de réseau

VI.18. Exemples

Exemple de recherche de fichiers et affectation automatique de nom de fichier à partir d'un numéro : le nom du fichier `Nfich$` possède le chemin d'accès `PathImp$` et a pour extension `EXT`. Il commence par les trois caractères `FIC` et est complété par un numéro de trois chiffres.

```
'* Toutes les variables dont le nom commence par i, j, …, n
'* sont des entiers
DEFINT I-N
Nerr = 0: Nfich = 0
WHILE Nerr = 0
    Nb$ = LTRIM$(STR$(Nfich))
    WHILE LEN(Nb$) < 3: Nb$ = "0" + Nb$: WEND
    Nfich$ = PathImp$ + "\FIC" + Nb$ + ".EXT"
    CALL FileSearch(Nfich$, Nerr)
    Nfich = Nfich + 1
WEND
```

Le programme de recherche de fichier utilisant la fonction 4EH fait appel au DOS à l'aide de la procédure `interruptX` à laquelle on transmet le numéro d'interruption et deux structures pour les registres. Ces deux structures ont pour type `RegTypeX` :

```
TYPE RegTypeX
     ax    AS INTEGER
     bx    AS INTEGER
     cx    AS INTEGER
     dx    AS INTEGER
     bp    AS INTEGER
```

```
      si    AS INTEGER
      di    AS INTEGER
      flags AS INTEGER
      ds    AS INTEGER
      es    AS INTEGER
END TYPE

'*-----------------------------------*
'* Recherche d'un fichier            *
'*    Nom$  = nom du fichier         *
'*    NoErr = numéro d'erreur si <> 0 *
'*-----------------------------------*
SUB FileSearch (Nom$, NoErr)
DIM iRegIn AS RegTypeX, iRegOut AS RegTypeX

'* Le tableau d'entiers iBidon() sert de DTA
' $DYNAMIC
DIM iBidon(64)
'* Définition de la DTA. On transmet à la fonction 1AH
'* l'adresse du tableau
iRegIn.ax = &H1A00: IntNum = &H21
iRegIn.ds = VARSEG(iBidon(0))
iRegIn.dx = VARPTR(iBidon(0))
CALL interruptX(IntNum, iRegIn, iRegOut)

'* Recherche à l'aide de la fonction 4EH
NoErr = 0: IntNum = &H21: iRegIn.ax = &H4E00: iRegIn.cx = 0
'* on complète le nom du fichier avec un caractère NUL
Name$ = Nom$ + CHR$(0): iRegIn.ds = VARSEG(Name$)
iRegIn.dx = SADD(Name$)
CALL interruptX(IntNum, iRegIn, iRegOut)
'* on récupère le drapeau de retenue CY dans l'entier iCarry
iCarry = iRegOut.flags AND 1
IF iCarry = 1 THEN NoErr = iRegOut.ax

ERASE iBidon
END SUB
```

CHAPITRE VII

La gestion de la souris

Le *driver* de gestion de la souris est fourni, soit sous forme d'un *Device Driver* MOUSE.SYS, soit sous forme d'un utilitaire résident MOUSE.COM.

VII.1. Les masques de définition du curseur graphique

Le curseur associé à la souris peut être soit un curseur de type *texte*, soit un curseur de type *graphique.*

En mode graphique, la forme du curseur est définie par un *masque d'écran* et un *masque de curseur*. Chaque masque est constitué de 16 mots de 16 *bits*. Les masques sont utilisés de la façon suivante :

– un ET logique est effectué entre le masque d'écran et l'état des pixels de l'écran,
– puis un OU EXCLUSIF est effectué entre le résultat de l'opération précédente et le masque de curseur.

En moyenne résolution, à chaque pixel est associé soit 2 *bits*, soit 4 *bits* selon le nombre de couleurs du mode d'affichage choisi.

Il existe deux types de curseurs en mode texte :

– un curseur dit "système", qui est constitué d'un bloc clignotant dont on définit la première ligne et la dernière ligne active (fonction 10),
– un curseur dit "logique", qui affecte la forme et l'attribut du caractère qu'il recouvre en fonction de deux masques de 16 *bits* dits "masque d'écran" et "masque de curseur". Le logiciel effectue d'abord un ET logique entre les 16 *bits* du caractère affiché (code + attribut) et le masque d'écran, puis un OU exclusif entre le résultat de cette opération et le masque de curseur.

Au curseur est associé la notion de point actif, ou point *chaud*, qui définit la position du curseur sur l'écran. Pour un curseur de type "texte", la position est donnée sous forme ligne-colonne de texte. Pour un curseur "graphique", la position est donnée sous forme ligne-colonne "pixel".

VII.2. Les fonctions de gestion de la souris

Les fonctions de gestion de la souris sont appelées à partir de l'interruption INT 33H.

INT 33H/0	Initialisation de l'état de la souris
	AX 0
	AX –1 souris installée, 0 souris non installée BX nombre de boutons de la souris

INT 33H/1	Affichage curseur
	AX 1

INT 33H/2	Effacement curseur
	AX 2

INT 33H/3	Acquisition état et position du curseur
	AX 3
	BX état boutons : *bit* 0 bouton gauche (souris à 2 ou 3 boutons) *bit* 1 bouton droit (souris à 2 ou 3 boutons) *bit* 2 bouton milieu (souris à 3 boutons) CX position horizontale DX position verticale
	Le bit d'état est à 0 pour un bouton relevé, à 1 sinon. En mode texte, la position est donnée en pixels, l'écran étant considéré comme étant un écran graphique avec des caractères 8×8, quelque soit le type d'affichage.

INT 33H/4	Positionnement curseur
	AX 4 CX position horizontale DX position verticale

INT 33H/5	Lecture état d'enfoncement des boutons
	AX 5 BX choix du bouton (voir fonction 3)
	AX état du bouton (voir fonction 3) BX nombre de fois où le bouton a été appuyé depuis le dernier appel. CX position horizontale lors de la dernière pression DX position verticale lors de la dernière pression
	Le compteur est remis à zéro après appel de cette fonction.

INT 33H/6	Lecture état de relèvement des boutons
	AX 6 BX choix du bouton (voir fonction 3)

AX,BX,CX,DX : mêmes défintions que pour la fonction 5 de gestion de la pression sur le bouton
Le compteur est remis à zéro après appel de cette fonction

INT 33H/7	Définition abscisses min. et max. pour le curseur
	AX 7 CX, DX abscisse minima, maxima
	Les valeurs de CX et DX sont permutées si DX<CX

INT 33H/8	Définition ordonnées min. et max. pour le curseur
	AX 8 CX, DX ordonnées minima, maxima
	Les valeurs de CX et DX sont permutées si DX<CX

INT 33H/9	Définition de la forme du curseur en mode graphique
	AX 9 ES:DX adresse de la table des masques BX,CX abscisse, ordonnée du point de saisie

INT 33H/0AH	Définition de la forme du curseur en mode texte
	AX 0AH BX 1 si curseur matériel 0 si curseur logiciel CX ligne de départ curseur (si BX=1) attribut et code du caractère (si BX=0) pour le masque d'écran : CH, CL = attribut, code DX ligne de fin curseur (si BX=1) attribut et code du caractère (si BX=0) pour le masque de curseur : DH, DL = attribut, code

INT 33H/0BH	Lecture du déplacement du curseur en nombre de déplacements élémentaires du curseur
	AX 0BH
	CX déplacement horizontal depuis le dernier appel de la fonction DX déplacement vertical depuis le dernier appel de la fonction
	Le compteur est remis à zéro après appel de cette fonction

INT 33H/0CH	Définition de l'adresse de traitement des évènements
	AX 0CH CX masque d'appel des évènements : *bit* 0 modification de position du curseur *bit* 1 bouton gauche appuyé

	bit 2 bouton gauche relaché *bit* 3 bouton droit appuyé *bit* 4 bouton droit relaché *bit* 5 bouton milieu appuyé *bit* 6 bouton milieu relaché *bit* 7 à 15 non utilisés
ES:DX	adresse du programme de traitement de l'évènement

INT 33H fonct. 0DH/0EH	Contrôle d'émulation *light pen*
AX	0DH : validation émulation 0EH : inhibition émulation
Lors de l'initialisation, une validation est effectuée.	

INT 33H/0FH	Définition de la sensibilité de la souris
AX	0FH
CX	rapport déplacement horizontal / nombre de pixels (défaut 8)
DX	rapport déplacement vertical / nombre de pixels (défaut 16)

INT 33H/10H	Effacement automatique de curseur
AX	10H
CX,DX	abscisse, ordonnée point haut-gauche
SI, DI	abscisse, ordonnée point bas-droit
Si le curseur disparaît, il faut le réafficher par la fonction 1. Certains *drivers* fournissent les coordonnées dans un tampon dont l'adresse est passée en paramètre dans ES:DX.	

INT 33H/13H	Valeur du seuil de doublement de la vitesse du curseur
AX	13H
DX	seuil au-delà duquel la vitesse est doublée (par défaut 64 déplacements élémentaires de la souris par seconde)
Si on donne une valeur irréalisable, cette option est désactivée.	

INT 33H/14H	Changement de routine de traitement
AX	14H
ES:DX	adresse de la nouvelle routine
CX	masque du nouvel appel
ES:DX	adresse de l'ancienne routine
CX	masque de l'ancien appel

Bits du masque : – 0 le curseur change de position – 1 le bouton gauche est enfoncé – 2 le bouton gauche est relâché – 3 le bouton droit est enfoncé – 4 le bouton droit est relâché
Lorsque la routine est appelée, elle passe les informations suivantes : AX masque de condition (le bit correspondant à l'évènement mis à 1) BX état du bouton CX abscisse DX ordonnée DI abscisse en nombre de déplacements élémentaires SI ordonnée en nombre de déplacements élémentaires DS contient la valeur du segment du driver de la souris. La routine doit mettre à jour DS.

INT 33H/15H	Informations sur le tampon du *driver* de la souris
	AX 15H
	BX taille du tampon utilisé
	Utilisé en liaison avec les fonctions 16H et 17H.

INT 33H/16H	Sauver l'état du *driver* de la souris
	AX 16H ES:DX pointeur vers le tampon utilisé

INT 33H/17H	Restituer l'état du *driver* de la souris
	AX 17H ES:DX pointeur vers le tampon utilisé

INT 33H/1DH	Modifie le numéro de page écran
	AX 1DH BX numéro de page utilisé

INT 33H/1EH	Lecture du numéro de page écran
	AX 1EH
	BX numéro de page utilisé

ANNEXE A

Jeu d'instructions

Le jeu d'instruction a fortement évolué ces dernières années avec les changements intervenus dans l'architecture des processeurs. On présente dans ce qui suit les instructions de base des architectures 86, 286 et 386. Les extensions dites MMX introduites pour accélérer les traitements mis en œuvre pour les jeux et applications multimédia (algorithmes de compression/décompression de sons et images) ne sont pas décrites.

VIII.1. Codage des instructions

Drapeaux	
O	*Overflow*
D	Direction (1=adresses hautes vers adresses basses)
I	Interruptions
T	*Trap*
S	Signe
Z	Zéro
A	Retenue auxiliaire
P	Parité
C	Retenue

Les instructions notées * sont relatives au 286.

Abréviation		Abréviation	
reg	registre	cond	code condition
reg8	registre 8 *bits*	sreg	registre de segment
reg16	registre 16 *bits*	imm	valeur immédiate
mem	adresse mémoire		

Abréviation	
d	Direction (0=mémoire à registre, 1=registre à mémoire)
w	Indicateur mot (1)/octet (0)
s	Signe (indicateur à 1 pour extension de 8 à 16 *bits*)

mod	Indicateur de mode : – 00 si r/m=110 : adressage direct, sinon disp=0 et on a indirection. L'opérande doit être basé, indexé ou basé/indexé. – 01 accès indirect avec disp sur 8 *bits* – 10 accès indirect avec disp sur 16 *bits* – 11 instruction à deux registres (reg pour destination et r/m pour source)
reg	Registres : – 000 AX AL – 001 CX CL – 010 DX DL – 011 BX BL – 100 SP AH – 101 BP CH – 110 SI DH – 111 DI BH
sreg	Registres de segment – 00 ES – 01 CS – 10 SS – 11 DS
r/m	Registre/mémoire (mode d'accès). Si le champ mod est à 11, r/m désigne le registre source et reg la destination. Sinon le mode d'adressage est codé par r/m de la façon suivante : – 000 DS:[BX+SI+disp] – 001 DS:[BX+DI+disp] – 010 SS:[BP+SI+disp] – 011 SS:[BP+DI+disp] – 100 DS:[SI+disp] – 101 DS:[DI+disp] – 110 DS:[BP+disp] – 111 DS:[BX+disp]
disp	*Offset* des opérandes
data	Constantes

VIII.2. Transferts

VIII.2.1. Données simples

MOV	MOVE Data
MOV reg,reg	100010dw mod reg r/m disp(0/2)
MOV mem,reg	
MOV reg,mem	
MOV mem,imm	1100011w mod 000 r/m disp(0/2) data(1/2)
MOV reg,imm	1011w reg data(1/2)
MOV mem,acc	101000dw disp(0/2)
MOV acc,mem	

MOV sreg,reg16
MOV sreg,mem16
MOV reg16,mem
MOV mem16,sreg
	100011d0 | mod sreg r/m | disp(0/2)

MOVS MOVSB/MOVSW	MOVE String Data DS:SI → ES:DI SI et DI mis à jour selon le flag de direction D Utilisé avec REP, CX doit contenir le nombre d'éléments
MOVS [ES:]dest,[sreg:]source MOVSB MOVSW	1010010w

XCHG	Exchange
XCHG reg,reg XCHG reg,mem XCHG mem,reg	1000011w | mod reg r/m | disp(0/2)
XCHG acc,reg16 XCHG reg16,acc	10010 reg

LODS LODSB/LODSW	LOAD String Operand DS:SI → acc AL ou AX SI mis à jour selon le flag de direction D
LODS [sreg:]source LODSB LODSW	1010110w

STOS STOSB/STOSW	STORE String Operand acc AL ou AX → ES:DI DI mis à jour selon le flag de direction D Utilisé avec REP, CX doit contenir le nombre d'éléments
STOS [ES:]destination STOSB STOSW	1010101w

LEA	LOAD Effective Address
LEA reg,mem	10001101w | mod reg r/m | disp(2)

LDS/LES	LOAD Far Pointer
LDS reg,mem	11000101 | mod reg r/m | disp(2)
LES reg,mem	11000100 | mod reg r/m | disp(2)

XLAT/XLATB	TRANSLATE BX pointe vers la table, AL donne l'indice de l'élément dans la table. Après exécution AL contient l'élément.
XLAT [[sreg:]mem]	10001101w | mod reg r/m | disp(2)

VIII.2.2. Manipulation de chaînes

Les manipulations de chaînes MOVS, LODS et STOS sont documentées par ailleurs.

SCAS OSZAPC SCASB/SCASW	SCAN String Flags Chaîne pointée par ES:DI L'élément de comparaison est mis dans l'accumulateur. DI mis à jour selon le flag de direction D. Utilisé avec REP, CX doit contenir le nombre d'éléments. Utilisé avec REPE ou REPNE, SI et DI pointent vers le premier élément suivant l'élément trouvé.
SCAS [ES:]dest SCASB SCASW	1010111w
CMPS OSZAPC CMPSB/CMPSW	COMPARE String Chaînes pointées par DS:SI et ES:DI Mêmes remarques qu'avec SCAS
CMPS [sreg:]source,[ES:]dest CMPSB CMPSW	1010011w
INS* INSB*/INSW*	INPUT from Port to String Chaîne pointée par ES:DI DI mis à jour selon le flag de direction D. CX éléments lorsqu'utilisé avec REP
INS [ES:]dest,DX INSB INSW	0110110w
OUTS* OUTSB*/OUTSW*	OUTPUT String to Port Chaîne pointée par DS:SI SI mis à jour selon le flag de direction D. CX éléments lorsqu'utilisé avec REP
OUTS DX,[sreg:]source INSB INSW	0110111w

REP	REPEAT String
REPE/REPZ	REPEAT while zero flag is set
REPNE/REPNZ	REPEAT while zero flag is cleared
REP	11110010
REPE/REPZ	11110011
REPNE/REPNZ	11110010

VIII.3. Instructions arithmétiques et logiques

VIII.3.1. Arithmétiques

ADD OSZAPC	Add

ADD reg,reg
ADD mem,reg
ADD reg,mem
	000000dw mod reg r/m disp(0/2)

ADD reg,imm ADD mem,imm	100000sw mod 000 r/m disp(0/2) data (1/2)
ADD acc,imm	0000010w data (1/2)

ADC OSZAPC	Add with Carry
ADC reg,reg ADC mem,reg ADC reg,mem	000100dw mod reg r/m disp(0/2)
ADC reg,imm ADC mem,imm	100000sw mod 010 r/m disp(0/2) data (1/2)
ADC acc,imm	0001010w data (1/2)

INC OSZAP	Increment
INC reg8 INC mem	1111111w mod 000 r/m disp(0/2)
INC reg16	01000 reg

SUB OSZAPC	Substract

SUB reg,reg
SUB mem,reg
SUB reg,mem
	001010dw mod reg r/m disp(0/2)

SUB reg,imm SUB mem,imm	100000sw mod 101 r/m disp(0/2) data (1/2)
SUB acc,imm	0010110w data (1/2)

SBB OSZAPC	Substract with Borrow
SBB reg,reg SBB mem,reg SBB reg,mem	000110dw mod reg r/m disp(0/2)

SBB reg,imm SBB mem,imm	100000sw | mod 011 r/m | disp(0/2) data (1/2)
SBB acc,imm	0001110w | data (1/2)

DEC | OSZAP |	Decrement
DEC reg8 DEC mem	1111111w | mod 001 r/m | disp(0/2)
DEC reg16	01001 reg

NEG | OSZAPC |	Two's Complement Negation
NEG reg NEG mem	1111011w | mod 011 r/m | disp(0/2)

MUL | OC |	Unsigned Multiply – opér16. * AX → DX:AX – opér8. * AL → AX
MUL reg MUL mem	1111011w | mod 100 r/m | disp(0/2)

IMUL | OC |	Signed Multiply – opér16. * AX → DX:AX – opér8. * AL → AX
IMUL reg IMUL mem	1111011w | mod 101 r/m | disp(0/2)

DIV	Unsigned Divide – DX:AX / opér16. → AX + reste DX – AX / opér8. → AL + reste AH
DIV reg DIV mem	1111011w | mod 110 r/m | disp(0/2)

IDIV	Signed Divide – DX:AX / opér16. → AX + reste DX – AX / opér8. → AL + reste AH
IDIV reg IDIV mem	1111011w | mod 111 r/m | disp(0/2)

VIII.3.2. Logiques

AND | O(0)SZPC(0) |	Logical AND
AND reg,reg AND mem,reg AND reg,mem	001000dw | mod reg r/m | disp(0/2)
AND reg,imm AND mem,imm	100000sw | mod 100 r/m | disp(0/2) data (1/2)

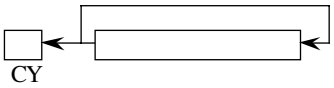
AND acc,imm	0010010w data (1/2)

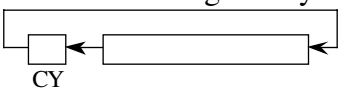
OR O(0)SZPC(0)	Inclusive OR
OR reg,reg OR mem,reg OR reg,mem	000010dw mod reg r/m disp(0/2)
OR reg,imm OR mem,imm	100000sw mod 001 r/m disp(0/2) data (1/2)
OR acc,imm	0000110w data (1/2)

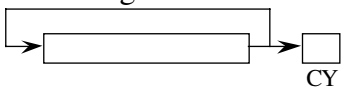
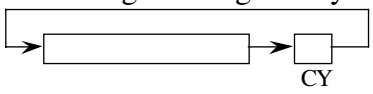
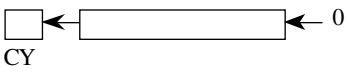
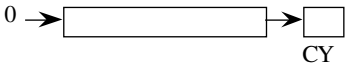
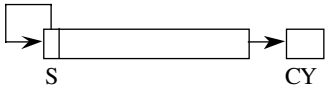
XOR O(0)SZPC(0)	Exclusive OR
XOR reg,reg XOR mem,reg XOR reg,mem	001100dw mod reg r/m disp(0/2)
XOR reg,imm XOR mem,imm	100000sw mod 110 r/m disp(0/2) data (1/2)
XOR acc,imm	0011010w data (1/2)

NOT	One's Complement Negation
NOT reg NOT mem	1111011w mod 010 r/m disp (0/2)

VIII.3.3. Décalages

ROL OC	Rotate left 
ROL reg,1 ROL mem,1	1101000w mod 000 r/m disp (0/2)
ROL reg,CL ROL mem,CL	1101001w mod 000 r/m disp (0/2)
ROL reg,imm8* ROL mem,imm8*	1100000w mod 000 r/m disp (0/2) data (1)

RCL OC	Rotate left through Carry 
RCL reg,1 RCL mem,1	1101000w mod 010 r/m disp (0/2)
RCL reg,CL RCL mem,CL	1101001w mod 010 r/m disp (0/2)
RCL reg,imm8* RCL mem,imm8*	1100000w mod 010 r/m disp (0/2) data (1)

ROR [OC]	Rotate Right 
ROR reg,1 ROR mem,1	1101000w mod 001 r/m disp (0/2)
ROR reg,CL ROR mem,CL	1101001w mod 001 r/m disp (0/2)
ROR reg,imm8* ROR mem,imm8*	1100000w mod 001 r/m disp (0/2) data (1)
RCR [OC]	Rotate Right through Carry 
RCR reg,1 RCR mem,1	1101000w mod 011 r/m disp (0/2)
RCR reg,CL RCR mem,CL	1101001w mod 011 r/m disp (0/2)
RCR reg,imm8* RCR mem,imm8*	1100000w mod 011 r/m disp (0/2) data (1)
SHL/SAL [OSZPC]	Shift Left 
SHL reg,1 SHL mem,1	1101000w mod 100 r/m disp (0/2)
SHL reg,CL SHL mem,CL	1101001w mod 100 r/m disp (0/2)
SHL reg,imm8* SHL mem,imm8*	1100000w mod 100 r/m disp (0/2) data (1)
SHR [OSZPC]	Shift Right 
SHR reg,1 SHR mem,1	1101000w mod 101 r/m disp (0/2)
SHR reg,CL SHR mem,CL	1101001w mod 101 r/m disp (0/2)
SHR reg,imm8* SHR mem,imm8*	1100000w mod 101 r/m disp (0/2) data (1)
SAR [OSZPC]	Shift Right Arithmetic 
SAR reg,1 SAR mem,1	1101000w mod 111 r/m disp (0/2)

SAR reg,CL SAR mem,CL	1101001w | mod 111 r/m | disp (0/2)
SAR reg,imm8* SAR mem,imm8*	1100000w | mod 111 r/m | disp (0/2) data (1)

VIII.3.4. Comparaisons

CMP OSZAPC	COMPARE Two Operands

CMP reg,reg
CMP mem,reg
CMP reg,mem
	001110dw | mod reg r/m | disp (0/2)

CMP reg,imm CMP mem,imm	1000000sw | mod 111 r/m | disp (0/2) data (1/2)
CMP acc,imm	0011110w | data (1/2)

CMPS OSZAPC CMPSB/CMPSW	COMPARE String chaîne pointée par DS:SI comparée à chaîne pointée par ES:DI. SI et DI mis à jour selon le flag de direction D. Utilisé avec REPE ou REPNE, CX doit contenir le nombre d'éléments. Après exécution, SI et DI pointent vers le premier élément suivant l'élément trouvé.
CMP [sreg:]source,[ES:]dest CMPSB CMPSW	1010011w

TEST O(0)SZPC(0)	Logical Compare
TEST reg,reg TEST mem,reg TEST reg,mem	1000011w | mod reg r/m | disp (0/2)
TEST reg,imm TEST mem,imm	1111011w | mod 000 r/m | disp (0/2) data (1/2)
TEST acc,imm	1010100w | data (1/2)

VIII.4. Branchements

VIII.4.1. Inconditionnels

CALL	Call Procedure
CALL etiq	11101000 | disp (2)
CALL etiq	10011010 | disp (4)
CALL reg	11111111 | mod 010 r/m
CALL mem16 CALL mem32	11111111 | mod 011 r/m

INT	Interrupt
INT imm8	11001101 data(1)
INT 3 (Breakpoint)	11001100

IRET	Interrupt Return
	11001111

RET	Return from Procedure
RET	11000011
RETN	
RET imm8	11000010 data(2)
RETN imm8	
RET	11001101
RETF	
RET imm16	11001010 data(2)
RETF imm16	

JMP	JUMP Unconditionally
JMP etiq	11101011 disp(1)
JMP etiq	11101001 disp(2)
JMP etiq	11101010 disp(4)
JMP reg16	11111111 mod 100 r/m
JMP mem16	
JMP mem32	11111111 mod 101 r/m

ENTER*	Make Stack Frame
ENTER imm16,0	11001000 data(2) data(1)
ENTER imm16,1	11001000 data(2) data(1)
ENTER imm16,imm8	11001000 data(2) data(1)

LEAVE*	High Level Procedure Exit
	11001001

VIII.4.2. Conditionnels

Jcondition	Jump Conditionally (S) indique le résultat d'une comparaison signée
	0111cond
JB/JNAE (0010) <	
JAE/JNB (0011) ≥	
JBE/JNA (0110) ≤	
JA/JNBE (0111) >	

JE/JZ	(0100)	=	
JNE/JNZ	(0101)	≠	
JL/JNGE	(1100)	< (S)	
JGE/JNL	(1101)	≥ (S)	
JLE/JNG	(1110)	≤ (S)	
JG/JNLE	(1111)	> (S)	
JS	(1000)	<0	bit de signe = 1
JNS	(1001)	≥0	bit de signe = 0
JC	(0010)	CY=1	Retenue
JNC	(0011)	CY=0	
JO	(0000)	OV=1	*Overflow*
JNO	(0001)	OV=0	
JP/JPE	(1010)	P=1	parité paire (*Even*)
JNP/JPO	(1011)	P=0	parité impaire (*Odd*)

JCXZ	Jump if CX is Zero
JCXZ etiq	11100011 disp(1)

INTO IT	Interrupt on *Overflow* (INT 4)
	11001110

BOUND*	Check Array Bounds (INT 5 en cas de violation de limites, avec une adresse de retour pointant sur l'instruction BOUND)
BOUND reg16,mem32	01100010 mod reg r/m disp(2)

VIII.4.3. Boucles

LOOP	LOOP
LOOP etiq	11100010 disp(1)

LOOPE/LOOPZ	LOOP condition
LOOPE etiq	11100001 disp(1)

LOOPNE/LOOPNZ	LOOP condition
LOOPNE etiq	11100000 disp(1)

JCXZ	

VIII.5. Divers

VIII.5.1. Entrées-sorties

IN	*Input from Port*
IN acc,imm	1110010w data(1)
IN acc,DX	1110110w

OUT	*Output to Port*
OUT imm8,acc	1110011w data(1)
OUT DX,acc	1110111w

Les instructions INS et OUTS sont documentées avec les instructions de manipulation de chaînes.

VIII.5.2. Conversions de type

CBW	*Convert Byte to Word* (extension de signe de AL dans AH)
	10011000

CWD	*Convert Word to Double* (Extension de signe de AX dans DX)
	10011001

VIII.5.3. Conversions BCD

AAA AC	*ASCII Adjust after Addition* (si AL>9 alors AH:=AH+1)
	00110111

AAS AC	*ASCII Adjust after Substraction* (si AL>9 alors AH:=AH−1)
	00111111

AAM SZP	*ASCII Adjust after Multiply* (conversion AL vers BCD non condensé dans AH et AL)
	11010100 00001010

AAD SZP	*ASCII Adjust before Division* (Conversion BCD non condensé dans AH et AL vers binaire dans AX)
	11010101 00001010

DAA SZAPC	*Decimal Adjust after Addition* (Conversion binaire dans AL vers BCD condensé dans AL)
	00100111

DAS SZAPC	*Decimal Adjust after Substraction* (Conversion binaire dans AL vers BCD condensé dans AL)

	00101111

VIII.5.4. Modification des drapeaux

		Clear Flag	
CLC	C(0)	11111000	
CLD	D(0)	11111100	
CLI	I(0)	11111010	
CMC	C	11110101	
CLTS*	(Clear Task Switched Flag)	00001111	00000110
STC	C(1)	11111001	
STD	D(1)	11111101	
STI	I(1)	11111011	

Les instructions POPF et PUSHF sont documentées avec les instructions de manipulation de chaînes.

LAHF	LOAD Flags into AH
	10011111

SAHF	STORE AH into Flags
SAHF SZAPC	10011110

VIII.5.5. Manipulation de la pile

PUSH	PUSH
PUSH reg16	01010 reg
PUSH mem16	11111111 mod 110 r/m disp(2)
PUSH sreg	00 sreg 110
PUSH imm*	011010s0 data(1/2)

PUSHF	PUSH Flags
	10011100

PUSHA*	PUSH all (PUSH AX, CX, DX, BX, SP, BP, SI, DI)
	01100000

POP	POP
POP reg16	01011 reg
POP mem16	10001111 mod 000 r/m disp(2)
POP sreg	00 sreg 111

POPF	POP Flags

POPF ODITSZAPC	10011101
POPA*	POP all
	01100001

VIII.5.6. Contrôle du processeur

NOP	No Operation
	10010000
ESC	Escape Le premier opérande est un code instr. coprocesseur sur 6 bits HHHLLL. Le second est un opérande pour ce dernier.
ESC imm,reg ESC imm,mem	11011HHH mod LLL r/m
WAIT	Wait Suspension d'activité du processeur en attente d'une fin d'exécution coprocesseur
	10011011
LOCK	*Lock the Bus* Blocage des autres processeurs. Précède une instruction ayant accès à une zone mémoire commune.
	11110000
HLT	Halt Suspension d'activité du processeur en attente d'une interruption
	11110100

VIII.5.7. Contrôle des tâches (mode privilégié)

ARPL* Z	Adjust Requested Privilege Level Vérification des privilèges d'accès
ARPL reg,reg ARPL mem,reg	01100011 mod reg r/m disp(0/2)
CLTS*	Clear Task Switched Flag
CLTS	00001111 00000110
LAR* Z	Load Access Rights L'opérande doit contenir un sélecteur.
LAR reg16,reg16 LAR reg16,mem16	00001111 00000010 mod reg r/m disp(0/2/4)

LGDT/LIDT/LLDT*	Load Descriptor Table			
LGDT mem64 (table globale)	00001111	00000001	mod 010 r/m	disp(2)
LIDT mem64 (table des interruptions)	00001111	00000001	mod 011 r/m	disp(2)
LLDT reg16 LLDT mem16 (table locale)	00001111	00000000	mod 010 r/m	disp(0/2)

LMSW*	Load Machine Status Word			
LMSW reg16 LMSW mem16	00001111	00000001	mod 110 r/m	disp(0/2)

LSL* Z	Load Segment Limit			
LSL reg16,reg16 LSL mem16,mem16	00001111	00000011	mod reg r/m	disp(0/2)

LTR*	Load Task Register			
LTR reg16 LTR mem16	00001111	00000000	mod 001 r/m	disp(0/2)

SGDT/SIDT/SLDT*	Store Descriptor Table			
SGDT mem64 (table globale)	00001111	00000001	mod 000 r/m	disp(2)
SIDT mem64 (table des interruptions)	00001111	00000001	mod 001 r/m	disp(2)
SLDT reg16 SLDT mem16 (table locale)	00001111	00000000	mod 000 r/m	disp(0/2)

SMSW*	Store Machine Status Word			
SMSW reg16 SMSW mem16	00001111	00000001	mod 100 r/m	disp(0/2)

STR*	Store Task Register			
STR reg16 STR mem16	00001111	00000000	mod 001 reg	disp(0/2)

VERR* VERW*	Verify Read Verify Write			
VERW reg16 VERW mem16	00001111	00000000	mod 100 r/m	disp(0/2)
VERW reg16 VERW mem16	00001111	00000000	mod 101 r/m	disp(0/2)

ANNEXE B

Coprocesseurs 80X87

Les coprocesseurs de la famille Intel[1], Cyrix, Weitek … apportent un gain non négligeable en puissance de calcul au processeur. Ces coprocesseurs sont dotés d'un jeu d'instruction particulièrement riche : fonctions arithmétiques, transcendentales, etc. Le coprocesseur apparaît comme une extension du processeur qui gagne de nouvelles instructions et huit registres 80 *bits* supplémentaires. Les échanges entre processeur et coprocesseur sont complètement transparents au programmeur.

Les huit registres 80 *bits* sont accessibles par un mécanisme de pile ou directement par leur numéro. Les registres sont désignés par ST pour le sommet de pile et ST(1) jusqu'à ST(7).

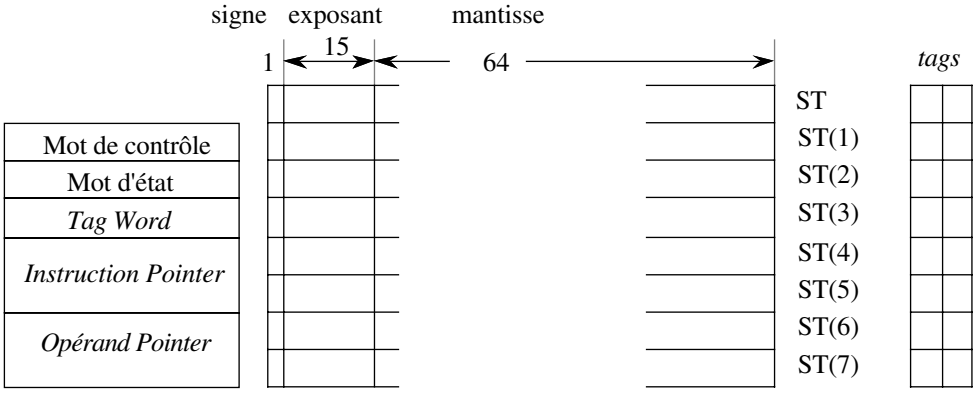


Fig.1 – *Architecture du coprocesseur*

La taille des instructions est de deux octets lorsqu'on ne fait pas appel à la mémoire. Dans le cas contraire cette taille est de quatre octets pour le 8087, le 80287 et le 80387 en mode 16 *bits*, et de six octets pour le 80387 en mode 32 *bits*. Sur le 8087 chaque instruction doit être précédée d'une inctruction FWAIT.

En interne le coprocesseur travaille sur une représentation flottante étendue de 80 *bits*. Cette dernière utilise 64 *bits* pour la mantisse, 15 *bits* pour l'exposant et un *bit* de signe. La particularité de

[1]80X87 Numeric Processor Extension, Documentation technique, INTEL Corp.

ce codage provient du fait que le premier *bit* de la mantisse (bit *caché* dans les autres représentations) apparaît dans la représentation.

Les formats traités, donc soumis à conversion lorsqu'ils sont chargés dans le coprocesseur, sont les suivants :

– entiers 16, 32 ou 64 *bits* codés en complément à deux,

– BCD condensé (*packed BCD*) de 18 chiffres. Le *bit* 79 est celui du signe, les *bits* 72 à 78 ne sont pas utilisés.

– flottant simple précision sur 32 *bits* : 23 de mantisse, 8 d'exposant et un de signe.

– flottant double précision sur 64 *bits* : 52 de mantisse, 11 d'exposant et un de signe.

Les drapeaux de deux *bits* associés à chaque registre interne, et organisés sous forme d'un mot de 16 bits, donnent l'état du contenu de chaque registre :

– 00 : contenu valide

– 01 : contenu égal à 0

– 10 : QNaN, SNaN, infini, anormal

– 11 : vide

La présence de ces drapeaux permet d'optimiser le fonctionnement du coprocesseur en évitant les calculs inutiles notamment. Le mot d'état possède la structure suivante :

B	C3	T2	T1	T0	C2	C1	C0	ES	SF	PE	UE	OE	ZE	DE	IE

Les *bits* T0, T1 et T2 donnent le numéro du registre de sommet de pile et les bits C0 à C3 sont les codes condition du coprocesseur. Les drapeaux d'exception sont :

– PE : précision

– UE : *underflow*

– OE : *overflow*

– ZE : division par zéro

– DE : opérande anormal

– IE : opération non valide

Les interruptions 7, 9, 13 et 16 correspondent aux exceptions coprocesseur sur le 80386.

IX.1. Transferts

FLD FILD/FBLD	Chargement dans la pile
	FLD st(*i*) met en sommet de pile le contenu de st(*i*) FLD *mem* charge un nombre codé flottant en sommet de pile FILD *int* charge un entier codé complément vrai en sommet de pile FBLD *bcd* charge un nombre codé BCD en sommet de pile

FXCH	Echange avec le sommet de pile
	FXCH st(*i*) : ST ↔ ST(*i*) Si st(*i*) n'est pas spécifié : ST ↔ ST(1)

FST/FIST	Ecrit

FSTP/FISTP/FBSTP	Ecrit avec pop du sommet de pile

FLD1 / FLDZ / FLDL2E FLDL2T / FLDLG2 FLDLN2 / FLDPI	*Pushes a constant onto the stack*
	ST:=+1.0, ST:=+0.0, ST:=$\log_2(e)$ ST:=$\log_2(10)$, ST:=$\log_{10}(2)$ ST:=$\log_e(2)$, ST:=π

IX.2. Instructions arithmétiques

FABS	Valeur absolue
	$\lvert ST \rvert \rightarrow ST$

FADD/FIADD	Addition
	FADD ST,ST(*i*) FADD ST(*i*),ST FADD *real* (32 ou 64 bits) FADD *int* (16 ou 32 bits)

FADDP	Addition et pop ST
	FADDP ST(*i*),ST : ST(*i*)=ST(*i*)+ST et pop ST

FSUB/FISUB	Soustraction
	source – destination → destination

FSUBP	Soustraction et pop ST
	source – destination → destination

FSUBR/FISUBR	Soustraction inversée
	destination – source → destination

FSUBRP	Soustraction inversée et pop ST
	destination – source → destination

FMUL/FIMUL	Multiplication

FMULP	Multiplication et pop ST

FSCALE	*Scale*
	$ST \times 2^{ST(1)} \rightarrow ST$

FDIV/FIDIV	Division
	destination / source → destination

FDIVP	Division et pop ST
	destination / source → destination

FFDIVR/FIDIVR	Division inversée
	source / destination → destination

FDIVRP	Division inversée et pop ST
	source / destination → destination

FCHS	Changement de signe
	–ST → ST

FRNDINT	Arrondi
	Méthode d'arrondi donnée par le champ (bits 11-10) du mot de contrôle : – 00 arrondi au plus proche (action par défaut) – 01 arrondi au plus près de $-\infty$ – 10 arrondi au plus près de $+\infty$ – 11 troncature au plus près de 0

FSQRT	Racine carrée
	$\sqrt{ST} \rightarrow ST$ (–0 → –0)

FPREM	Calcul du reste
	Reste de ST / ST(1) → ST (garde son signe) ST / ST(1) est tronquée au plus près de 0

FPREM1	Calcul du reste (80387)
	Reste = ST – ST(1)×*quotient* → ST (garde son signe) *quotient* est l'entier le plus proche de ST/ST(1)

FXTRACT	Mantisse et exposant
	exposant→ST et *mantisse* mise en pile

IX.3. Fonctions transcendentales

FPTAN	Tangente partielle
	Y/X=TAN(ST). Y→ST et X mis en pile, puis Y→ST(1) et X→ST – 8087 et 80287 $0 \leq ST \leq \pi/4$ – 80387 $-2^{63} \leq ST \leq 2^{63}$

FPATAN	ArcTangente partiel
	ARCTAN(ST(1)/ST) → ST Il est fait un pop dans la pile et le résultat est mis dans ST. – 8087 et 80287 0≤ST et ST(1)<∞ – pas de limites pour le 80387

FSIN	Fonction sinus (80387)
	$-2^{63} \le ST \le 2^{63}$ SIN(ST) → ST

FCOS	Fonction cosinus (80387)
	$-2^{63} \le ST \le 2^{63}$ COS(ST)→ ST

FSINCOS	Fonction sinus et cosinus (80387)
	SIN mis en pile puis COS dans ST

F2XM1	Calcul de 2^x-1
	x est pris dans ST et le résultat retourné dans ST. • $-.5 \le x \le +.5$ sur les 8087 et 80287 • $-1. \le x \le +1.$ sur le 80387

FYL2X	
	Calcul de ST(1)×$\log_2$(ST) (0<ST<+∞, −∞<ST(1)<+∞) pop, puis résultat dans ST

FYL2PI	
	$ST(1) \times \log_2(ST+1)$ avec $\left\{0 < \lvert ST \rvert < 1 - \sqrt{2}\ 2,\ -\infty < ST(1) < +\infty\right\}$

IX.4. Comparaisons

FCOM/FICOM	Comparaison

FCOMP/FICOMP	Comparaison avec pop

FCOMPP	Comparaison avec double pop

FUCOM FUCOMP / FUCOMPP	Comparaison (80387)
	La différence avec FCOM provient du fait que si un opérande est de type *NaN* (*Not-a-Number*) il n'y a pas exception, mais les codes conditions (bits 8, 9, 10, 14) prennent la valeur 11∅1.

FTST	Test de nullité
	Test par rapport à +0.0.
FXAM	*Examine*
	ST → *condition codes*

IX.5. Instructions de contrôle

FINIT/FNINIT	Initialisation coprocesseur
	Sur le 80387 les codes conditions sont mis à 0, alors qu'ils ne sont pas modifiés dans le cas des 8087 et 80287.
FFREE	
	FFREE ST(*i*)
FNOP	Opération vide
FWAIT	Opération de synchronisation
FDECSTP	Décrémente le pointeur de sommet de pile
	De 0 il passe à 7
FINCSTP	Incrémente le pointeur de sommet de pile
	De 7 il passe à 0
FCLEX/FNCLEX	*Clear exceptions*
FSETPM	*Set Protected Mode* (80287)
FDISI/FNDISI	*Disable Interrupts* (8087)
FENI/FNENI	*Enable Interrupts* (8087)
FSAVE/FNSAVE	Sauvegarde de l'état du coprocesseur
	L'état du processeur occupe 98 octets sauf en mode 32 *bits* sur le 80387 où il occupe 108 octets. Après sauvegarde le coprocesseur est réinitialisé.
FLDCW	*Load Control Word*

FRSTOR	Relecture de l'état du coprocesseur
FSTCW/FNSTCW	*Store Control Word*
FSTSW/FNSTSW	*Store Status Word*
	Le mot d'état est rangé en mémoire. Sur les 80287 et 80387 il peut être écrit dans AX.
FLDENV	*Load Environment State*
	Charge 14 octets (*control word, status word, tag word, instr. pointer* et *operand pointer*) décrivant l'état du coprocesseur.
FSTENV/FNSTENV	*Store Environment State*

IX.6. Exemple

On désire calculer les 16 premiers coefficients *corr*(*j*) de corrélation sur des blocs de signal *sig*(*j*) de 3000 points. On utilise l'expression suivante :

$$corr_j = \sum_{k=0}^{2999-j} sig(k)sig(k+j)$$

Le programme écrit en Basic Microsoft est le suivant :

```
DEFINT I-N
DIM correl(15), Sig(2999)
'* initialisation du bloc de données
FOR i = 0 TO 5: Sig(i) = 1 / (2 ^ i): NEXT i
'* calcul des 16 premiers coefficients de corrélation
FOR j = 0 TO 15
    icorfin = 2999 - j
    '* on transmet par valeur l'adresse du tableau des données Sig()
    '* les autres paramètres sont transmis par adresse
    iSeg = VARSEG(Sig(0)): iOff = VARPTR(Sig(0))
    CALL FCorrel(j, BYVAL iSeg, BYVAL iOff, icorfin, correlJ)
    correl(j) = correlJ
NEXT j
END
```

Le sous-programme de calcul de chaque coefficient est le suivant :

```
    .MODEL medium
    .CODE
    .8087
    .286

     PUBLIC  FCorrel
FCorrel     PROC
    push    bp                  ; Entry sequence - save old BP
    mov     bp,sp               ; Set stack framepointer
    pusha
    fsave   [bp-110]            ; sauvegarde état et initialisation
                                ; sans synchronisation
                                ; (nécessite 94 octets)
    mov     bx,[bp+12]          ; Sig Segment
    mov     es,bx
    mov     bx,[bp+10]          ; Sig Offset
    mov     si,bx
    mov     bx,[bp+8]           ; icorfin
    mov     cx,[bx]
    inc     cx                  ; on va de 0 à icorfin
    mov     bx,[bp+14]          ; indice j
    mov     bx,[bx]             ; sert pour accéder à Sig(k+j)
    shl     bx,2
    fldz                            ; push 0 onto the stack
bclFcorrel:
    fld     dword ptr es:[si]       ; Sig(k) --> ST
    fmul    dword ptr es:[si][bx]   ; ST * Sig(k+j) --> ST
    fadd                            ; st(1):=st(1)+st and pop the stack
    add     si,4
    loop    bclFcorrel

    mov     bx,[bp+6]               ; résultat
    fst     dword ptr[bx]
    frstor  [bp-110]            ; on restaure l'état du coprocesseur
    popa
    pop     bp                  ;
    ret     10                  ; on restitue le pointeur de pile
FCorrel  ENDP
    END
```

Généralement les assembleurs se chargent de rajouter si nécessaire les instructions de synchronisation `WAIT`. Ici cela sera fait sur les instructions `fsave` et `frstor`.

ANNEXE C

Notes diverses

1. Générateur d'horloges

Le générateur d'horloge constitué par un *timer* I8253 est utilisé pour émettre le son du haut-parleur, pour les signaux de rafraîchissement des mémoires et pour l'horloge temps réeel.

– Le haut parleur est contrôlé à partir du *timer* 3 et du registre d'adresse 61H du contrôleur de clavier I8742 selon le schéma :

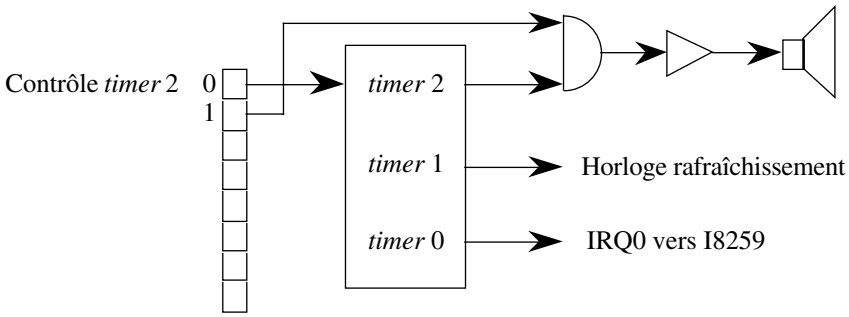


Fig.1 – *Connexion du haut-parleur*

Le contrôleur I8253 est initialisé grâce à la séquence :

```
mov     al,0B6H
out     43H,al
```

qui correspond à la configuration :

1 0	1 1	0 1 1	0
timer 2	transfert de deux octets	Mode division de fréquence	Comptage binaire

Le contrôle de la fréquence de la sonnerie est faite de la façon suivante :

```
  mov  al,0          ; initialisation du diviseur de fréquence :
  out  42H,al        ; on écrit en 42H le poids faible
  mov  al,0AH        ; puis le poids fort (ici on a choisi 0A00H).
  out  42H,al        ; fréquence = (1 193 180 / 0A00H)
  ;
  in   al,61H        ;
  push     ax        ;
  or   al,3          ; contrôle du timer et du haut-parleur
  out  61H,al        ;
  ;
  mov  ax,15         ; durée du BEEP --------------
  xor cx,cx          ;
attente:
  loop attente       ;
  dec  ax            ;
  jnz  attente       ; ----------------------------
  ;
  pop  ax            ; on restitue le registre
  out  61H,al        ; de contrôle du timer
```

– Programmation de l'horloge : l'horloge utilise un décompteur (compteur 0) d'un contrôleur I8253 (*Programmable Interval Timer*) qui envoie une interruption (IRQ0) chaque fois que son contenu passe à zéro. Normalement sa valeur initiale est fixée à zéro, ce qui lui donne une période de 65536 *tops* d'horloge (horloge à 1,19318 MHz). On peut modifier cette valeur initiale pour obtenir d'autres périodes :

```
mov   al,36H      ; comptage binaire sur 16 bits
out   43H,al      ; registre de mode
mov   ax,new_val  ; nouvelle valeur initiale du décompteur
out   40H,al      ; poids faible
mov   al,ah
out   40H,al      ; poids fort
```

Pour le remettre en son état d'origine, il suffit de lui redonner une valeur initiale égale à 0.

– Lecture du contenu du *timer* 0 :

```
xor    al,al
out    43h,al
jmp    $+2
in     al,40h
mov    ah,al
jmp    $+2
in     al,40h
xchg   ah,al
```

2. Affichage du contenu de la RAM CMOS

```
DEFINT I-N
CLS
RESTORE donnees
FOR i = 0 TO &H3F
    IF ((i + 1) MOD 20 = 0) THEN
        WHILE INKEY$ <> " ": WEND
    END IF
    ' on écrit l'adresse en 70H
    OUT &H70, i
    j = INP(&H71):              ' et on lit le contenu en 71H
    READ x$
    PRINT TAB(6); HEX$(i); TAB(12); j; TAB(17);
    PRINT HEX$(j); TAB(30); x$
NEXT i
END

donnees:
DATA "Secondes","Secondes alarme","Minutes","Minutes alarme"
DATA "Heure","Heure alarme"
DATA "Jour semaine","Jour mois","Mois","Année"
DATA "Reg.1 horloge","Reg.2 horloge","Reg.3 horloge","Reg.4 horloge"
DATA "Etat contrôleur","Etat retour en mode non protégé"
DATA "Type lecteur", "Type lecteur AT", "Type DD", "?"
DATA "Octet équipement", "Taille L < 1024K pour SETUP" ,"Taille H < 1024K
pour SETUP"
DATA "Taille L > 1024K pour SETUP" ,"Taille H > 1024K pour SETUP"
DATA "Autre type de DD 1", "Autre type de DD 2", "?","?","?","?","?","?"
DATA "?","?","?","?","?","?","?","?","?","?","?","?","?","?"
DATA "CRC L","CRC H"
DATA "Taille L > 1024K pour RESET" ,"Taille H > 1024K pour RESET"
DATA "Siècle" ,"Drapeau à la Mise en Route"
DATA "?","?","?","?","?","?","?","?","?","?","?","?","?","?"
```

3. Bus XT

	B			A	
	1	GND	I/O CHCK	1	Erreur parité ou ext. I/O
Reset Driver	2	+Reset DRV	D7	2	Données
	3	+5V	D6	3	
Interr. Request	4	+IRQ2	D5	4	
	5	–5V	D4	5	
DMA Request	6	+DRQ2	D3	6	
	7	–12V	D2	7	
Sélection d'une carte d'extension	8	Card Sel.D	D1	8	
	9	+12V	D0	9	
	10	GND	+I/O CH.RDY	10	I/O Channel Ready
Ecriture mémoire	11	MEMW	+AEN	11	Add.Enable pour DMA
Lecture mémoire	12	MEMR	A19	12	Adresses
Ecriture E/S	13	IOW	A18	13	
Lecture E/S	14	IOR	A17	14	
DMA Acknowledge 3	15	DACK3	A16	15	
	16	+DRQ3	A15	16	
	17	DACK1	A14	17	
	18	+DRQ1	A13	18	
	19	DACK0	A12	19	
Horloge de base	20	CLOCK	A11	20	
	21	Reservé	A10	21	
	22	Reservé	A9	22	
	23	+IRQ5	A8	23	
	24	Reservé	A7	24	
	25	+IRQ3	A6	25	
	26	DACK2	A5	26	
Fin de comptage lors de DMA	27	+T/C	A4	27	
Latch des signaux d'adresse	28	+ALE	A3	28	
	29	+5V	A2	29	
Oscillateur 14,31818 MHz	30	+OSC	A1	30	
	31	+GND	A0	31	

4. Connecteur parallèle

Les registres utilisés pour la gestion de l'interface parallèles sont les suivants :

– Registre 3BEH ‘écriture d'état’ :

bit	
0	Etat ligne Strobe
1	Etat ligne Auto Feed
2	Etat ligne Initialisation
3	Etat ligne Sélection entrée
4	0 : interruption inhibée, 1 : interruption validée
5	0 : latch d'écriture validé, 1 : latch d'écriture inhibé

– Registre 3BEH ‘lecture d'état’ :

bit	
0	Etat ligne Strobe
1	Etat ligne Auto Feed
2	Etat ligne Initialisation
3	Etat ligne Sélection entrée
4	Etat ligne validation d'int.

– Registre 3BDH ‘lecture d'état d'initialisation’ :

bit	
3	Etat ligne Error
4	Etat ligne SLCT
5	Etat ligne Page End
6	Etat ligne Acknowledge
7	Etat ligne Busy

– Le registre 3BCH est le registre de données

– Constitution du bus parallèle :

1	Strobe
2	D0
3	D1
4	D2
5	D3
6	D4
7	D5
8	D6
9	D7
10	$\overline{\text{Acknowledge}}$
11	$\overline{\text{Busy}}$
12	fin de papier
13	$\overline{\text{sélection}}$
14	$\overline{\text{Auto Feed}}$
15	$\overline{\text{Erreur}}$
16	$\overline{\text{Initialisation imprimante}}$
17	$\overline{\text{Entrée sélection}}$
18-25	Masse

5. Lecteur de disquettes

Les registres utilisés 3F0H (décodés de 3F0H à 3F3H) à 3F4H pour la gestion de l'interface disquettes sont les suivants :

– Registre 3F4H ‘lecture d'état’ :

bit	
0	Lecteur A en cours d'opération
1	Lecteur B en cours d'opération
2	Lecteur C en cours d'opération
3	Lecteur D en cours d'opération
4	Contrôleur en cours d'opération
5	Indicateur DMA
6	0 : lecture, 1 : écriture
7	Demande de transfert avec µP

– Registre 3F0H (décodé de 3F0H à 3F3H sur le PC) ‘commande’ :

bit	
0-1	Numéro du lecteur sélectionné
2	Reset contrôleur 8272
3	Contrôle canal DMA
4	Validation sélection lecteur 0
5	Validation sélection lecteur 1
6	Validation sélection lecteur 2
7	Validation sélection lecteur 3

Les *bits* D4 à D7 du bus de données à 1 contrôlent la mise en route des moteurs des lecteurs.

6. Canaux DMA

A chaque canal DMA est associé deux registres 16 *bits* :

adresses	
0-1	Canal 0
2-3	Canal 1
4-5	Canal 2
6-7	Canal 3

Registre de commande/état 8 *bits* (adresse 8) :

bit	Commande	Etat
0	Transfert mémoire-mémoire :1 validé / 0 inhibé	T/C canal 0
1	Registre adr. : 0 hold canal 0 inhibé / 1 validé	T/C canal 1
2	Contrôleur : 0 valide / 1 inhibé	T/C canal 2
3	Timing : 0 normal / 1 compressé	T/C canal 3
4	Priorité : 0 fixe / 1 rotative	DREQ canal 0
5	Ecriture : 0 mode retardé / 1 étendu	DREQ canal 1
6	DREQ : 0 actif bas / 1 actif haut	DREQ canal 2
7	DACK : 0 actif bas / 1 actif haut	DREQ canal 3

Registre de demande DMA (adresse 9) :

bit	Signification
0-1	(1-0) = numéro de canal
2	Demande DMA : 0 : non / 1 : oui
3-7	inutilisé

Registre de validation de masquage (adresse 0AH) :

bit	Signification
0-1	(1-0) = numéro de canal
2	Masquage : 0 : non / 1 : oui
3-7	inutilisé

Registre de mode DMA (adresse 0BH) :

bit	Signification
0-1	(1-0) = numéro de canal
2-3	(3-2) : 00 vérification / 01 écriture / 10 lecture
4	Auto-initialisation : 1 validée
5	0 : Incrément / 1 : décrément
6-7	(7-6) mode : 00 demand / 01 single / 10 block / 11 cascade

Autres registres :

adresse	Signification
0CH	8 bits : pointeur d'octet (en écriture)
0DH	8 bits : temporaire (en lecture)
0EH	8 bits : réinitialisation du masquage (en écriture)
0FH	8 bits : positionnement des bits de masquage (en écriture)
80H à 83H	Registres 4 bits d'extension d'adressage

7. Contrôleurs d'interruption 8259

Chaque PIC a 4 registres 8 *bits* a pour l'initialisation (ICW1 à ICW4) et 3 registres de fonctionnement (OCW1 à OCW3). Accéder aux OCW_i ou aux ICW_i dépend de ce que l'on écrit dans ICW1.

7-1- Initialisation

```
; comme D4=1 on écrit dans ICW1
mov    al,11H             ; ICW1 (IC4=1,SNGL=0,ADI=0,LTIM=0)
out    20H,al             ; ADI:intervalle d'adresse=8
                          ; LTIM:détection de fronts
                          ; SNGL:seul 8259 maître du système
                          ; IC4:on va écrire das ICW4
mov    al,08H             ; base des IRQ (IT0->IRQ8,IT1->IRQ9,…)
out    21H,al             ; ICW2
mov    al,04H             ; l'entrée 2 correspond à un esclave
                          ; (ce PIC est maître)
out    21H,al             ; ICW3
mov    al,01H             ; µPM=1 -> processeur 8086
                          ; AEOI=0 (le EOI n'est pas automatique)
out    21H,al             ; ICW4
mov    al,11111111b       ; masquage des IT
out    21H,al             ; OCW1
;
; comme D4=1 on écrit dans ICW1
mov    al,11H
out    A0H,al
mov    al,70H             ; base des IRQ (IT0->IRQ70H,IT1->IRQ71H,…)
out    A1H,al             ; ICW2
mov    al,02H             ; esclave niveau 2
out    A1H,al             ; ICW3
mov    al,01H             ; µPM=1 -> processeur 8086
```

```
                              ; AEOI=0 (le EOI n'est pas automatique)
out     A1H,al                ; ICW4
mov     al,11111111b          ; masquage des IT
out     A1H,al                ; OCW1
```

7-2- Traitement de l'interruption

L'acquittement se fait par la commande EOI (*End Of Interrupt*) envoyée au contrôleur d'interruption. Dans le cas du PC cette commande est codée 20H :

```
cli
mov     al,20H
out     20H,al          ; ou mov A0H,al …
```

8. Interface série

L'interface série utilise un contrôleur de type USART (NS/PC16450N de *National Semiconductor*). Ce dernier permet le contrôle de la vitesse de transmission (de 50 à 9600 bps), du nombre de *bits* échangés (5 à 8), du nombre de *bits* de stop (1, 1,5 ou 2), de la parité (paire, impaire ou sans) et des interruptions. Des contrôleurs plus performants permettent désormais de travailler avec des modems plus rapides.

– Constitution du connecteur :

1		terre
2	PC->	TxD (données du PC vers terminal)
3	PC<-	RxD (données du terminal vers PC)
4	PC->	RTS (demande d'émission-le PC indique au terminal qu'il peut envoyer un caractère)
5	PC<-	CTS (prêt à émettre-le terminal indique au PC qu'il peut envoyer un caractère)
6	PC<-	DSR
7		masse
8	PC<-	DCD (le terminal indique qu'il est en ligne-détection de porteuse)
20	PC->	DTR (le PC indique au terminal qu'il est prêt)
22	PC<-	RI (indicateur de sonnerie-tentative d'entrer en communication)

– Adresses des contrôleurs :

port1	port 2	bit 7 (reg. comm. ligne)	
3F8H	2F8H	0	donnée à émettre
3F8H	2F8H	0	donnée reçue
3F8H	2F8H	1	diviseur de fréquence poids
3F9H	2F9H	1	faible
			diviseur de fréquence poids fort
3F9H	2F9H	0	
3FAH	2FAH	ϕ	définition des interruptions
			identification des interruptions
3FBH	2FBH	ϕ	
3FCH	2FCH	ϕ	commande de ligne
3FDH	2FDH	ϕ	commande de modem
3FEH	2FEH	ϕ	état ligne
			état modem

Remarque : lorsque le *bit* DLAB (*bit* 7 du registre de commande) est à 1, on accède au registre du diviseur de fréquence (2F8/2F9 ou 3F8/3F9). Sinon on accède aux registres de donnée (2F8/3F8) ou au registre de définitin des interruptions (2F9/3F9).

La valeur du diviseur est obtenue par la formule : $diviseur = \frac{115200}{bps}$ où bps est la vitesse de transfert en *bits*/seconde. La vitesse de transfert ne peut excéder 9600. Le registre de définition des interruptions est initialisé à 0.

– Registre de définition (écriture) ou d'état (lecture) de transmission :

bit	
1 et 0	00 : 5 bits
	01 : 6 bits
	10 : 7 bits
	11 : 8 bits
2	0 : un bit de STOP
	1 : plus d'un bit de STOP
3	0 : pas de parité
	1 : parité validée
4	0 : parité impaire
	1 : parité paire
5	
6	0 : SOUT inhibé
	1 : SOUT validé
7	bit DLAB

– Registre de définition des interruptions :

bit	Signification
0	1 : valide les interruptions à la réception lorsque le tampon de ligne est rempli
1	1 : valide les interruptions à l'émission lorsque le tampon de ligne est vide
2	1 : valide les interruptions sur changement d'état ligne
3	1 : valide les interruptions sur changementd'état modem
4-7	inutilisé

A l'initialisation, tous ces *bits* sont mis à 0. La priorité de ces différentes interruptions est, dans l'ordre des priorités décroissantes : ‘état ligne’, ‘donnée reçue’, ‘donnée émise’ et ‘état modem’.

– Registre d'identification d'interruption :

bit	
0	0 : indique d'une interruption est en attente de traitement
1-2	00 : état registre ligne (écrasement caractère, erreur de parité, de trame ou break)
	01 : donnée reçue
	10 : émetteur prêt
	11 : état registre modem (CTS, RI, DCD, DSR)
3-7	inutilisés (à 0)

– Registre de contrôle modem :

bit	Signification
0	sortie auxiliaire $\overline{\text{DTR}}$
1	sortie auxiliaire $\overline{\text{RTS}}$
2	sortie auxiliaire $\overline{\text{OUT1}}$
3	sortie auxiliaire $\overline{\text{OUT2}}$
4	1 : valide le test d'écho : les lignes DTR, RTS, OUT1 et OUT2 sont redirigées de manière interne vers les entrées CTS, DSR, DCD et RI.
5-7	0

– Registre d'état ligne :

bit	Signification
0	1 : donnée prête à être lue
1	1 : erreur d'écrasement de caractères (des caractères ont été reçus mais pas lus)
2	1 : erreur de parité
3	1 : erreur de trame (premier bit de stop à 0)
4	1 : indique une coupure ligne (break indicator)
5	1 : indique que le tampon d'émission est vide
6	registres d'émission **et** de réception vides
7	0

– Registre d'état modem :

bit	Signification
0	1 : indique un changement d'état de la ligne CTS
1	1 : indique un changement d'état de la ligne DSR
2	1 : indique une fin de sonnerie
3	1 : indique un changement d'état de la ligne DCD
4	$\overline{\text{CTS}}$ ou RTS du registre de commande modem si test d'écho
5	$\overline{\text{DSR}}$ ou DTR du registre de commande modem si test d'écho
6	$\overline{\text{RI}}$ ou OUT1 du registre de commande modem si test d'écho
7	$\overline{\text{DCD}}$ ou OUT2 du registre de commande modem si test d'écho

Exemple de traitement de la ligne série sous interruptions en C Microsoft

Le traitement sous interruptions de données transmises sur le ligne série peut être réalisé directement en langage évolué. L'exemple ci-après illustre le déroutement de l'interruption provenant du contrôleur série vers la fonction dénommée `NewIntr0Bh`. Le langage utilisé est le C Microsoft.

La transmission utilise les interruptions en émission et en réception. En émission, il est nécessaire, lorsqu'on désire transférer un bloc de données, d'envoyer une première donnée après test de l'état de la ligne, puis d'envoyer les suivantes en réponse à l'interruption ‘tampon de transmission vide’.

Lors de l'initialisation il faut effectuer une première lecture du registre de donnée pour ‘débloquer’ les interruptions qui, sans cela, ne sont pas transmises.

```
#include <dos.h>
…
void (_interrupt _far *OldIntr0Bh)(void);
void _interrupt _far NewIntr0Bh)(void);
void init_ser(void);
void SerIntrInstall(void);
void SerIntrRestore(void);
…
/* installation du nouveau traitement */
void SerIntrInstall(void)
{
OldIntr0Bh = _dos_getvect(0xb);
_dos_setvect(Oxb,NewIntr0Bh);
}
/* restoration de l'ancienne adresse de traitement */
void SerIntrRestore(void)
{
_dos_setvect(Oxb,OldIntr0Bh);
}
/* Nouvelle procédure de traitement */
void _interrupt _far NewIntr0Bh()
{
char rxchar, rxstatus;
rxstatus = (char)(inp(0x2fa) & 6) ;

if (rxstatus == 4)      /* tampon de réception plein */
    {
    rxchar = (char)(inp(0x2f8) & 0xff);  /* lecture caractère */
    …
    }
else if (rxstatus == 2)      /* tampon d'émission vide (il faut */
                             /* déjà avoir envoyé un caractère) */
    {
    …
    outp(0x2f8,(int)rxchar);
    …
    }
    outp(0x20,0x20);                    /* handshake contrôleur */
}

/* initialisation du port série */
void init_ser(void)
{
unsigned rchar;

/* configuration 7 bits, 1 bit de stop, parité paire avec DLAB=1 */
```

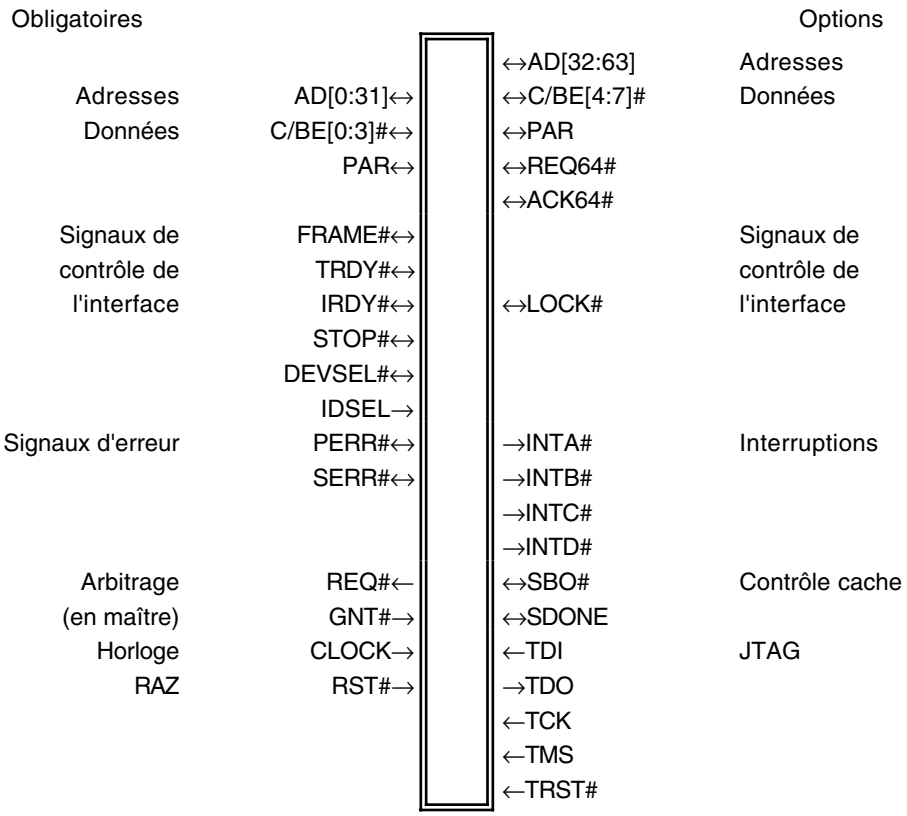
```
outp(0x2FB, 0x8a);

/* configuration 9600 bps avec DLAB=1 */
outp(0x2F9, 0);            /* poids faible */
outp(0x2F8, 0xc);          /* poids fort */

/* reconfiguration 7 bits, 1 bit de stop, parité paire avec DLAB=0 */
outp(0x2fb, 0xa);

/* configuration OUT2=high, RTS=high, DTR=high avec DLAB=0 */
outp(0x2fc, 0xb);

/* set interrupt en émission et réception avec DLAB=0 */
outp(0x2F9, 3);

/* lecture de tous les états et du registre de donnée. Cela permet en outre
de mettre le contrôleur dans le bon état */
outp(0x2F8, 0);         /* on envoie une première donnée */
rchar=(char)inp(0x2f8); /* lecture */
rchar=(char)inp(0x2f9); …
rchar=(char)inp(0x2fa); …
rchar=(char)inp(0x2fb); …
rchar=(char)inp(0x2fc); …
rchar=(char)inp(0x2fd); …
rchar=(char)inp(0x2fe); …
}
```

9. Contrôleur clavier

Un micro-contrôleur I8742 est utilisé pour gérer les échanges avec le clavier (communication série RS-422). La communication entre processeur et micro-contrôleur passe par des registres aux adresses 60H, 61H, 64H. En 60H on peut venir lire la donnée envoyée par le clavier, ou y écrire une commande pour le clavier. Le registre 64H est utilisé en tant que registre d'état ou de registre de commande. Il faut remarquer qu'un écriture de 0FxH en 64H force les *bits* 0 à 3, en fonction de la valeur de x, du port de sortie du micro-contrôleur. Or le *bit* 0 est relié au *reset* ! Un octet de la RAM CMOS est utilisé pour coder le type de *reset*.

10. Bus PCI

Le bus *PCI* (*Peripheral Component Interface*) a été défini à l'initiative d'Intel et avec la participation de Compaq, IBM et NCR, pour pallier le manque de performances du bus *ISA*. Le composant qui joue le rôle d'interface entre le bus du processeur et le bus *PCI* permet de travailler à une vitesse maxima de 33 MHz, vitesse indépendante de celle du processeur. Le bus *PCI* possède un certain nombre de lignes obligatoires et prévoit les extensions futures par ses lignes optionnelles :



11. Quelques processeurs

		Vitesse (MHz)		Copr.		
Intel	486sxj			–		machine 16 *bits*
	486sl	25		–		
	486sx	25,33		–		
	486dx	33,50		×		
	486dx2	50,66		×		25/50 ou 33/66
	486dx4	75		×		25/75
		100		×		33/100
		100		×		50/100
	Pentium	60,66	0,8µ	×		Caches 8Ko/8Ko, associatif par bloc
		75		×		à 2 blocs
	P54C	90/100		×		Caches 8Ko/8Ko, associatif par bloc à 2 blocs
Cyrix	486slc					
	486dx2V		cmos 0,6µ		3,3V	Cache *Write Back* de 8 Ko
	M1	90/100				32 registres avec fenêtrage, Prédiction de branchement

AMD	486dx 486sx2 486dx2 K5	40 66 40/80 40/120				Cache 16 Ko
IBM	486slc2	25/50		–		Copr.387sx/CX83s87 25MHz
NexGen	Nx586	60/66		–		22 registres avec fenêtrage Caches 16Ko/16Ko associatif par bloc à 4 blocs. FPU optionnel

ANNEXE D

Interruption Multiplex

Pour chacun des utilitaires la fonction 0 est une fonction d'identification qui retourne 0FFH en cas d'installation. Il est possible de retourner plus d'informations. Normalement les numéros d'identification laissés aux utilisateurs vont de 192 à 255.

1. Print.exe

INT 2FH fonction 0100H	5 Lire la présence de PRINT.EXE
	AX 0100H
	AL 0FFH si PRINT.EXE a été chargé
	Get Print.exe Install State

INT 2FH fonction 0101H	5 Ajoute un fichier dans la queue
	AX 0101H DS:DX adresse d'une structure dont le premier octet est 0 et le reste un nom de fichier terminé par 0.
	CY=1 AX contient un code erreur
	Add file to queue

Codes d'erreurs :

Code	Signification
0001	fonction non valide
0002	fichier introuvable
0003	chemin introuvable
0004	trop de fichiers ouverts
0005	accès refusé
0008	queue d'impression pleine
000C	accès invalide
000F	lecteur invalide

INT 2FH fonction 0102H	
	5 Supprime un fichier de la queue d'impression
	AX 0102H DS:DX adresse du nom d'un fichier (terminé par le caractère 0)
	CY=1 AX contient un code d'erreur
	Remove file from queue

INT 2FH fonction 0103H	
	5 Vide la queue d'impression
	AH 0103H
	Remove all files from queue

INT 2FH fonction 0104H	
	5 Suspension de l'impression
	AX 0104H
	DX nombre d'erreurs DS:SI adresse de la queue d'impression (table d'entrées de 64 octets contenant les noms des fichiers)
	Hold print job and get status

INT 2FH fonction 0105H	
	5 Reprise des impressions
	AX 0105H
	Release print jobs

INT 2FH fonction 0106H	
	5 Lire le périphérique d'impression
	AX 0106H
	CY=0 et AX=0 queue d'impression vide CY=1 AX=2 DS:SI adresse de l'en-tête de périphérique d'impression
	Get printer device status

2. Assign.com

INT 2FH fonction 0600H	
	5 Test d'installation de ASSIGN.COM
	AX 0600H
	AL 0FFH si ASSIGN.COM est installé
	Get assign.com installed state

3. Share.exe

INT 2FH fonction 1000H	
	5 Test d'installation de SHARE.EXE
	AX 1000H
	AL 0FFH si SHARE.EXE est installé
	Get share.exe installed state

4. Installation réseau

INT 2FH fonction 1100H	
	5 Test d'installation de réseau
	AX 1100H
	AL 0FFH si driver de réseau installé
	Get Network installed state Utilisé par les systèmes de gestion de fichier utilisant le réseau ou systèmes particuliers (gestion du CD-ROM MicroSoft par exemple).

5. Nlsfunc.exe

INT 2FH fonction 1400H	
	5 Test d'installation de NLSFUNC.EXE
	AX 1400H
	AL 0FFH si NLSFUNC.EXE est installé
	Get NLSFUNC.EXE installed state

6. Inoccupation du programme appelant

Cette fonction peut être utilisée pour informer le système d'exploitation qu'il est en attente d'une opération d'entrée-sortie. Il est donc possible au système de lancer l'exécution d'une autre tâche.

INT 2FH fonction 1680H	
	5 Appel inoccupé de MSDOS
	AX 1680H
	AL 0 indique que le système supporte cette interruption du programme.
	Cet appel informe MSDOS que le programme appelant accepte d'être interrompu (attente d'entrée/sortie appelant cette fonction de façon répétitive). Il faut s'assurer au préalable que INT 2FH est bien installé.

7. Ansi.sys

INT 2FH fonction 1A00H	
	5 Test d'installation de ANSI.SYS
	AX 1A00H
	AL 0FFH si ANSI.SYS est installé

Get ANSI.SYS installed state

8. Himem.sys

INT 2FH fonction 4300H	
	5 Test d'installation de HIMEM.SYS
	AX 4300H
	AL 80H si HIMEM.SYS est installé
	Get himem.sys installed state

INT 2FH fonction 4310H	
	5 Lit point d'entrée de HIMEM.SYS
	AX 4310H
	ES:BX adresse du point d'entrée de HIMEM.SYS
	Get himem.sys Address

HIMEM.SYS gère le bloc de mémoire haute (*HMA*), les *UMB* et l'état de la ligne A20. En principe il faut être prudent lorsqu'on utilise les fonctions de HIMEM.SYS si MSDOS l'utilise déjà. En pratique les programmes appelant donnent dans AH le numéro de la fonction et appellent ensuite le point d'entrée de HIMEM.SYS.

Sous-fonction (hexa)	
0	Lecture numéro de version du driver XMS
1	Réservation HMA
2	Libération HMA
3	Activation globale de la ligne A20
4	Désactivation globale de la ligne A20
5	Activation locale de la ligne A20
6	Désactivation locale de la ligne A20
7	Lecture état de la ligne A20
8	Requête de mémoie étendue libre
9	Allocation de bloc de mémoire étendue
A	Libération de bloc de mémoire étendue
B	Déplacement de bloc de mémoire étendue
C	Verrouillage de bloc de mémoire étendue
D	Déverrouillage de bloc de mémoire étendue
E	Lecture informations sur *handle*
F	Changement de taille de bloc
10	Allocation de bloc en mémoire haute (UMB)
11	Libération de bloc UMB.

(Se reporter à l'annexe E pour le descriptif des fonctions XMS).

9. Doskey.com

INT 2FH fonction 4800H	5 Test d'installation de DOSKEY.COM
	AX 4800H
	AL≠0 si DOSKEY.COM est installé
	Get doskey.com installed state

INT 2FH fonction 4810H	5 Lecture de la ligne de commande
	AX 4810H DS:DX adresse d'un tampon recevant la ligne de commande
	Lecture de 126 caractères de la ligne de commande. Le tampon a la structure suivante : – octet 0 : 80H – octet 1 : nombre de caractères lus–1 (on ne compte pas le retour charriot) – octet 2 à n : caractères lus

10. Construction de chaîne de notification

INT 2FH fonction 4B01H	5 Construit chaîne de notification
	AX 4B01H

INT 2FH fonction 4B02H	5 Teste présence du commutateur
	AX 4B02H

INT 2FH fonction 4B03H	5 Affecte un identificateur au commutateur
	AX 4B03H

INT 2FH fonction 4B04H	5 Libère un identificateur du commutateur
	AX 4B04H

INT 2FH fonction 4B05H	5 Identifie les données en instance
	AX 4B05H

11. Keyb.com

INT 2FH fonction AD80H	5 Lire version de KEYB.COM
	AX AD80H
	BH code majeur BL code mineur
	Get keyb.com version number

INT 2FH fonction AD81H	5 Ecrit le numéro de code de page actif de *keyb.com*
	AX AD81H BX nouveau code de page
	CY=1 et AX=1 : code page invalide

INT 2FH fonction AD82H	5 Initialisation drapeau de pays
	AX AD82H BL 0 si national, 0FFH si étranger
	CY=1 BL est incorrect
	Set keyb.com country flag

INT 2FH fonction AD83H	5 Lecture drapeau de pays
	AX AD83H
	BL 0 ou 0FFH
	Get keyb.com country flag

12. Installation de commandes

INT 2FH fonction AEH	
	AH AEH

Installable Command Permet de défnir de nouvelles commandes dans COMMAND.COM. Ces nouvelles commandes sont aussi accessibles via INT 2EH.

13. Graftabl.com

INT 2FH fonction B000H	5 Lit état d'installation de GRAFTABL.COM
	AX B000H

14. Append.exe

INT 2FH fonction B700H	5 Lit état d'installation de APPEND.EXE
	AX B700H

INT 2FH fonction B702H	5 Lit version de APPEND.EXE
	AX B702H

INT 2FH fonction B704H	5 Lit adresse de la liste des fichiers de APPEND.EXE
	AX B704H
	ES:DI adresse de la liste (chaîne ASCIIZ) des répertoires traités (les noms sont séparés par des points-virgules)

INT 2FH fonction B706H	5 Lit drapeau de mode de APPEND.EXE
	AX B706H
	BX mode : *bit* 0 indicateur d'activité 12 APPEND s'applique aux fichiers indiqués 13 option /path active 14 option /e active 15 option /x active

INT 2FH fonction B707H	5	Ecrit drapeau de mode de APPEND.EXE
	AX B707H	

INT 2FH fonction B711H	5	Initialise le drapeau de nom réel
	AX B711H	

ANNEXE E

Gestion de la mémoire

Lors de la mise en circulation des premiers PC, l'espace mémoire disponible de 640 Koctets semblait plus que suffisant pour toutes les applications à venir. Il fallut rapidement déchanter. Les nouveaux microprocesseurs de la famille I80X86, dont le I80286 sur les PC/AT, offrirent très vite un espace adressable bien supérieur (16Mo puis 4Go pour les I80386). Malheureusement, le système d'exploitation MSDOS ne sait toujours pas gérer cette mémoire supplémentaire. Les premiers logiciels à tirer parti de cette mémoire dite *étendue* furent les gestionnaires de disques virtuels, puis les tableurs qui y trouvèrent un moyen de simuler la mémoire *expansée* pour y mettre leurs données.

D'une manière générale, l'utilisation de la mémoire étendue pose deux problèmes :

– son accès à partir d'un programme s'exécutant en mode réel sous MSDOS (le mode de travail du 80X86). Cela concerne les programmes cherchant un espace mémoire supplémentaire pour y mettre des données,

– l'accès aux fonctions du DOS et du BIOS à partir de programmes travaillant en mode protégé.

Les différents utilitaires qui offrent ces facilités sont répertoriées sur le schéma suivant :

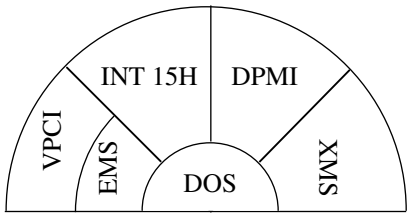


Fig.1 – *Les gestionnaires de mémoire*

1. Gestion de la mémoire EMS

INT 67H/1	Lecture d'état	LIM/EMS
	AH 40H	

AH=00	BX = segment
AH≠00	80H problème de *driver* EMM
	81H problème sur la carte EMS

INT 67H/2	Lecture de l'adresse de segment des pages accessibles (*frame*)	LIM/EMS
AH 41H		
AH=00	BX = segment	
AH≠00	80H problème de *driver* EMM	
	81H problème sur la carte EMS	

INT 67H/3	Lecture nombre de pages total et disponibles	LIM/EMS
AH 42H		
AH=00	BX = nombre de pages libres	
	DX = nombre total de pages EMS	
AH≠00	80H problème de *driver* EMM	
	81H problème sur la carte EMS	

INT 67H/4	Demande d'allocation de pages	LIM/EMS
AH 43H		
BX = nombre de pages de 16 Ko demandé		
AH=00	DX = numéro logique (*Handle*)	
AH≠00	80H problème de *driver* EMM	
	81H problème sur la carte EMS	
	85H plus de numéro logique disponible	
	87H il n'y a plus assez de pages libres	
	88H le nombre de pages demandé est nul	
Les numéros logiques sont de la forme FF00H, FE01H, FD01H, …		

INT 67H/5	Appel des pages logiques dans l'espace physique	LIM/EMS
AH 44H		
BX numéro de page logique		
DX numéro logique (*Handle*)		
AL numéro de page physique (0 à 3)		
AH≠0	80H problème de *driver* EMM	
	81H problème sur la carte EMS	
	83H numéro logique incorrect	
	8AH numéro de page logique incorrect	
	8BH numéro de page physique incorrect	

INT 67H/6	Libération de pages	LIM/EMS
AH 45H		
DX numéro logique (*Handle*)		

AH≠0	80H problème de *driver* EMM 81H problème sur la carte EMS 83H numéro logique incorrect 85H sauvegarde erronée de l'image de l'allocation.

INT 67H/7	Lecture de la version EMM	LIM/EMS
AH 46H		
AH=00	AL = numéro de version (codé BCD)	
AH≠00	80H problème de *driver* EMM 81H problème sur la carte EMS	

INT 67H/8	Sauvegarde d'une image d'allocation	LIM/EMS
AH 47H DX numéro logique (*Handle*)		
AH 00 correct 80H problème de *driver* EMM 81H problème sur la carte EMS 83H numéro logique incorrect 8CH zone de mémoire pour la sauvegarde est pleine 8DH sauvegarde déjà faite précédemment		

INT 67H/9	Restitution d'une image d'allocation	LIM/EMS
AH 48H DX numéro logique (*Handle*)		
AH≠0	80H problème de *driver* EMM 81H problème sur la carte EMS 83H numéro logique incorrect 8EH pas de sauvegarde correspondant à ce numéro logique	

INT 67H/0CH	Lecture du nombre de numéros logiques	LIM/EMS
AH 4BH		
AH=00	BX = nombre de numéros logiques affectés	
AH≠00	80H problème de *driver* EMM 81H problème sur la carte EMS	

INT 67H/0DH	Lecture du nombre de pages allouées	LIM/EMS
AH 4CH DX numéro logique (*Handle*)		
AH=00	BX = nombre de numéros logiques affectés	
AH≠00	80H problème de *driver* EMM 81H problème sur la carte EMS 83H numéro logique incorrect	

INT 67H/0EH	Sauvegarde générale	LIM/EMS

AH 4DH ES:DI adresse de tableau	
AH=00 AH≠00	BX = nombre de numéros logiques affectés 80H problème de *driver* EMM 81H problème sur la carte EMS
Chaque entrée du tableau contient deux octets pour le numéro logique et deux octets pour le nombre de pages allouées correspondant.	

2. Contrôle du fonctionnement en mode virtuel (VPCI)

D'une manière générale, l'utilisation de la mémoire étendue peut passer par trois techniques :

– l'allocation de type EMS : cette allocation se fait par demande de pages par l'intermédiaire des appels standards aux fonctions EMS vues précédemment,

– la seconde technique (*top-down allocation*) consiste à installer un *driver* qui déroute l'interruption 15H. Lors d'une demande d'allocation, la taille de mémoire étendue disponible est diminuée, de telle sorte que l'appel à la fonction 88H de l'interruption 15H retourne une taille inférieure à la précédente lors des nouvelles demandes.

– la troisième technique est utilisée en particulier par le gestionnaire de disque virtuel VDISK.SYS d'IBM (on modifie notamment l'interruption INT 19H de *boot*). Cette méthode est particulièrement délicate à mettre en œuvre car mettant en jeu une technique de marquage de l'espace mémoire utilisé.

La mémoire étendue est maintenant largement utilisée, soit pour y installer des disques virtuels, soit, plus récemment, par des environnements tels que :

– les systèmes "presque multitâches" (DOS *multitaskers*) qui gèrent plusieurs programmes DOS en mode virtuel 8086, eux-mêmes restant en mode protégé,

– les DOS-*extenders* : un DOS-*extender* constitue un environnement complet qui permet à des programmes écrit pour DOS d'accéder au mode protégé lorsque le besoin s'en fait sentir,

– les émulateurs de mémoire expansée.

La cohabitation de telles applications est assez critique. En effet le traitement de toutes les exceptions engendrées en mode virtuel 86 (par exemple les entrées-sorties) doit être entrepris en mode protégé. Or un seul programme de contrôle peut utiliser le mode protégé. C'est la raison pour laquelle, en 1987, un certain nombre de constructeurs de cartes d'extensions mémoire et autres concepteurs de logiciels ont tenté de mettre un peu d'ordre dans le gestion de l'espace mémoire au-delà des 640 Ko. Ce travail a débouché sur une norme appelée *VPCI* (*Virtual Control Program Interface*) qui définit un ensemble de fonctions d'appel à un gestionnaire d'espace mémoire étendue lorsqu'on utilise un 80386.

Lorsque le *driver* VPCI (VPCI *server*) est installé, il déroute les appels EMS. Les applications doivent, pour l'utiliser, vérifier que le processeur est bien un 80386, qu'il y a un émulateur EMS (dans ce cas, la première demande d'allocation fait passer en mode virtuel 86) puis qu'il y a bien le *driver* VPCI.

Un astérisque indique les appels qui sont disponibles seulement en mode protégé.

INT 67H/DEH	Test de présence du driver VPCI VPCI
	AH DEH AL 0

INT 67H/DEH	Définition de la table de pages VPCI
	AH DEH AL 1

INT 67H/DEH	Lecture de l'adresse physique maxima VPCI
	AH DEH AL 2

INT 67H/DEH	(*) Lecture du nombre total de pages de 4Ko disponibles VPCI
	AH DEH AL 3

INT 67H/DEH	(*) Allocation d'une page de 4Ko VPCI
	AH DEH AL 4

INT 67H/DEH	(*) Libération d'une page de 4Ko VPCI
	AH DEH AL 5

INT 67H/DEH	Lecture de l'adresse d'une page dans dans le premier Mo VPCI
	AH DEH AL 6

INT 67H/DEH	Lecture du contenu du registre CR0 VPCI
	AH DEH AL 7
	CR0 est le registre de contrôle du 386 (*Machine Status Word*)

INT 67H/DEH	Lecture du contenu des registres de mise au point du 80386 VPCI
	AH DEH AL 8

INT 67H/DEH	Chargement des registres de mise au point du 80386 VPCI
	AH DEH AL 9

INT 67H/DEH		Lecture des vecteurs d'interruptions correspondant aux interruptions physiques sur le contrôleur I8259	VPCI
	AH DEH AL 0AH		

INT 67H/DEH		Chargement des vecteurs d'interruptions sur le contrôleur I8259	VPCI
	AH DEH AL 0BH		

INT 67H/DEH	(*)	Passage du mode virtuel 86 au mode protégé ou inversement (selon le mode courant)	VPCI
	AH DEH AL 0CH		

3. Le driver XMS

Le *driver* XMS[1], fourni par Microsoft sous le nom *himem.sys*, permet aux programmes s'exécutant sous DOS d'accéder à un supplément de mémoire pris en HMA (les 64 premiers Koctets de la mémoire étendue), UMB (entre 640K et 1 Mo) ou EMB (mémoire étendue au-delà de la HMA). Ses spécifications résultent des travaux communs de Lotus®, Intel®, Microsoft® et AST®Research.

Pour savoir où est installé le *driver* XMS, il faut utiliser la fonction 4310H de l'interruption multiplex après avoir vérifié qu'il est bien installé (fonction 4300H). Ayant l'adresse dans ES:BX, l'accès s'y fait en donnant dans AH le numéro de fonction puis en faisant un CALL vers cette adresse :

```
XMSoffset   dw   (?)
XMSsegment  dw   (?)
…
mov    ax,4310H
int    2FH
mov    word ptr[XMSoffset],bx
mov    word ptr[XMSsegment],es
…
mov    ah,numfonc
call   [XMSOffset]
cmp    ax,1          ; erreur
jge    erreur
…
```

Fonction 0		Lecture numéro de version	XMS
	AX 0		

[1]Microsoft Corp, Lotus dev. Corp., Intel Corp., AST Research “eXtended Memory Specification (XMS) 2.0

AX	numéro de version codée BCD
BX	numéro (interne) de révision
DX	1 présence de HMA
	0 pas de HMA

Fonction 1 — Réservation de mémoire HMA — XMS

AX	1	
DX	taille requise en octets si l'appelant est un TSR ou un DD	
	0FFFFH si l'appelant est un programme d'application	
AX	1	si requête satisfaite
	0	sinon
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	90H	pas de HMA
	91H	HMA déjà utilisé
	92H	taille demandée < HMAMIN

Si la taille requise est ≥ paramètre HMAMIN de la ligne de commande, la requête est satisfaite.

Fonction 2 — Libération de mémoire HMA — XMS

AX	2	
AX	1	si requête satisfaite
	0	sinon
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	90H	pas de HMA
	93H	HMA non allouée

Fonction 3 — Validation globale de la ligne A20 — XMS

AX	3	
AX	1 si requête satisfaite, 0 sinon	
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	82H	erreur

Fonction 4 — Invalidation globale de la ligne A20 — XMS

AX	4	
AX	1 si requête satisfaite, 0 sinon	
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	82H	erreur

Fonction 5 — Validation locale de la ligne A20 — XMS

AX	5

AX	1 si requête satisfaite, 0 sinon
BL	80H fonction inexistante
	81H un driver VDISK est détecté
	82H erreur

Fonction 6	Invalidation locale de la ligne A20 XMS
AX	6
AX	1 si requête satisfaite, 0 sinon
BL	80H fonction inexistante
	81H un driver VDISK est détecté
	82H erreur

Fonction 7	Vérification de la ligne A20 XMS
AX	7
AX	1 si la ligne A20 est physiquement active, 0 sinon
BL	00H fonction OK
	80H fonction inexistante
	81H un driver VDISK est détecté

Fonction 8	Lecture de la taille mémoire étendue disponible XMS
AX	8
AX	taille du plus grand bloc disponible en ko
DX	taille totale disponible en ko
La HMA n'est jamais comprise dans la valeur retournée	

Fonction 9	Allocation de mémoire étendue XMS
AX	9
DX	taille requise en ko
AX	1 requête OK (DX *handle* affecté. 0 erreur :
BL	80H fonction inexistante
	81H un driver VDISK est détecté
	A0H toute la mémoire disponible est déjà allouée
	A1H tous les *handles* sont utilisés

Fonction 0AH	Libération de mémoire étendue XMS
AX	0AH
DX	numéro logique du bloc à libérer
AX	1 requête satisfaite, 0 erreur
BL	80H fonction inexistante
	81H un driver VDISK est détecté
	A2H *handle* non correct
	ABH *handle* verrouillé

Fonction 0BH	Déplacement de bloc de mémoire étendue XMS
AX	0BH
DS:SI	pointeur vers structure

AX	1 requête satisfaite, 0 erreur BL 80H fonction inexistante 81H un driver VDISK est détecté 82H erreur dûe à la ligne A20 A3H *handle* source non correct A4H *offset* source non correct A5H *handle* destination non correct A6H *offset* destination non correct A7H longueur non valide A8H si l'opération provoque un recouvrement incorrect A9H erreur de parité
Les transferts peuvent être faits de mémoire étendue à mémoire étendue ou entre mémoire conventionnelle et mémoire étendue. Il ne faut pas valider la ligne A20 avant l'appel à cette fonction.	

La structure utilisée est la suivante :

```
EMM    struc
  longueur dd   (?)  ; nombre d'octets à transférer (pair)
  NologS   dw   (?)  ; handle source (si NologS=0, l'offset source est
                     ; interprété comme une paire Segment:Offset en
                     ; mémoire conventionnelle
  OffsetS  dd   (?)
  NologD   dw   (?)  ; handle destination
  OffsetD  dd   (?)
EMM    ends
```

Fonction 0CH	Verrouillage de bloc de mémoire étendue XMS
	AX 0CH DX numéro logique du bloc
	AX 1 requête satisfaite : DX:BX adresse linéaire 32 *bits* du bloc verrouillé. 0 erreur BL 80H fonction inexistante 81H un driver VDISK est détecté A2H *handle* non correct ACH trop de blocs verrouillés ADH échec

Fonction 0DH	Déverrouillage de bloc de mémoire étendue XMS
	AX 0DH DX numéro logique du bloc

AX	1	requête satisfaite
	0	erreur
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	A2H	*handle* non correct
	A3H	bloc non verrouillé

Fonction 0EH — Lecture information sur bloc de mémoire étendue — XMS

AX	0EH
DX	numéro logique du bloc

AX	1	requête satisfaite
	BH	nombre de blocs verrouillés
	BL	nombre de *handles* disponibles
	DX	longueur du bloc en ko
	0	erreur
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	A2H	*handle* non correct

Fonction 0FH — Réallocation de bloc de mémoire étendue — XMS

AX	0FH
DX	numéro logique du bloc
BX	nouvelle taille requise en ko

AX	1 requête satisfaite, 0 erreur :	
	BL 80H	fonction inexistante
	81H	un driver VDISK est détecté
	A0H	toute la mémoire est déjà allouée
	A1H	*handles* tous utilisés
	A2H	*handle* non correct
	ABH	le bloc est verrouillé

Fonction 10H — Demande de bloc de mémoire expansée — XMS

AX	10H
DX	taille requise en paragraphes

AX	1	requête satisfaite
	BX	segment de l'UMB
	DX	taille du bloc alloué en paragraphes
	0	erreur
	DX	taille de l'UMB le plus grand en paragraphes
	BL 80H	fonction inexistante
	B0H	un UMB plus petit est disponible
	B1H	aucun UMB n'est disponible

Fonction 11H	Libération de bloc de mémoire expansée			XMS
	AX	11H		
	DX	segment de l'UMB		
	AX	1	requête satisfaite	
		0	erreur	
		BL	80H	fonction inexistante
			B2H	segment UMB non valide

4. Fonctions de gestion mémoire de INT 15H

Ces fonctions du BIOS sont les premières à avoir été implémentées pour faciliter les transferts entre mémoire conventionnelle et mémoire étendue. Normalement les *drivers* XMS sont accrochés à l'interruption 15H. Ils doivent donc tester AH pour voir s'il s'agit des fonctions 87H ou 88H.

INT 15H/87H	Transfert de mémoire (passage au mode protégé du 80286)	
	AH	87H
	CX	nombre de mots à déplacer
	ES:SI	adresse de la GDT
	AH	code état – 0 pas d'erreur – 1 erreur de parité sur la mémoire – 2 GDT incorrecte – 3 le mode protégé n'a pu être initialisé

INT 15H/88H	Lecture de la taille mémoire étendue
	AH 88H
	Taille mémoire étendue en Ko. S'il y a un *driver* XMS accroché à l'interruption 15H, celui-ci doit retourner une taille égale à 0 pour protéger la HMA.

5. Fonctions de gestion mémoire DPMI

Les spécifications DPMI (*DOS Protected Mode Interface*) font l'objet d'une norme décrite par Intel[(2)]. DPMI définit :

– un ensemble de fonctions BIOS et DOS appelables à partir de programmes s'exécutant en mode protégé,

– un ensemble de fonctions accessibles par INT 31H permettant aux programmes qui s'exécutent en mode protégé d'allouer de la mémoire, modifier les descripteurs, etc.

DPMI offre des services à des programmes dits *clients*. Ceux-ci peuvent être des programmes utilisateurs ou des *extenders*. Ces deniers constituent une couche logicielle qui est cliente, entre autres, de DPMI et qui offre es services aux programmes utilisateurs. Les autres services offerts par les *extenders*, sont généralement, dans l'ordre, VCPI/EMS, XMS et les fonctions de INT 15H.

[(2)]Intel & Microsoft "Protected Mode API For DOS Extended Applications, Version 0.9" 08/91

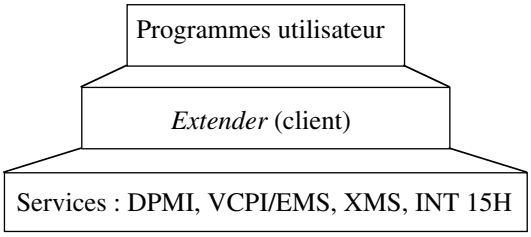


Fig.2 – *Les extenders*

BIBLIOGRAPHIE

RALF BROWN & JIM KYLE — PC INTERRUPTS - A Programmer's Reference to BIOS, DOS and Third-Party Calls, Addison Wesley.

MICROSOFT — MS-DOS Operating System Programmer's Reference Manual, Microsoft Corp., Bellevue, WA, 1984.

MICROSOFT — MS-DOS Guide de référence du Programmeur, Microsoft Press, DUNOD 1991.

KING R. A. — Guide de PC-DOS, Editions Sybex.

La bible du PC — Editions Micro Applications.

MICROSOFT — Documentation sur le macro-assembleur MASM.

IBM — The IBM PC Technical Reference Manual - IBM Corp., Boca Raton, FL, 1981.

IBM — The IBM PCAT Technical Reference Manual - IBM Corp., Boca Raton, FL, 1984.

IBM — Disk Operating System Technical Reference Manual - IBM Corp., Boca Raton, FL, 1983.

INTEL — Intel Microsystem Components Handbook, Intel Corp., Sante Clara, CA.

CODES ASCII

	0	1	2	3	4	5	6	7
0				0	@	P	`	p
1			!	1	A	Q	a	q
2			"	2	B	R	b	r
3			#	3	C	S	c	s
4			$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7			'	7	G	W	g	w
8			(	8	H	X	h	x
9			)	9	I	Y	i	y
A			*	:	J	Z	j	z
B			+	;	K	[	k	{
C			,	<	L	\	l	|
D			-	=	M	]	m	}
E			.	>	N	^	n	~
F			/	?	O	_	o	

A

B

C

D

E

F

G

H

I

J

K

L

M

N