MICRO-ORDINATEURS

Philippe DAX

CP/M

ET SA FAMILLE

GUIDE D'UTILISATION

MP/M | CP/M 86 | WORDSTAR | BASIC

sylvie bouvry jean pierre tissier

EYROLLES

DANS LA MÊME COLLECTION

SCHOMBERG	- Le Basic Universel.
SCHOMBERG	- Micro-ordinateurs : Comment ça marche ?
HERNANDEZ	- Pascal par l'exemple.
ASTIER	- La conduite de l'APPLE II.
NOLLET	- La conduite du ZX 81.
PELLIER	- La conduite du TRS 80.
KOSTAKIS & BAÏKOUR	- Tout sur les disques du TRS 80 (à paraître).
LADEVIE	- Votre gestion avec BASIC sur micro-ordinateur.
QUEINNEC	- Langage d'un autre type : LISP.
MONTEIL	- L'assembleur facile du 6502.
LEPAPE	- L'assembleur facile du Z 80.
PELLIER	- Programmez vos jeux d'action rapide sur TRS 80.
OROS & PERBOST	- ZX 81 à la conquête des jeux.
DAX	- CP/M et sa famille.
NOLLET	- Langage machine, trucs et astuces sur ZX 81

CP/M
ET SA FAMILLE
Guide d'utilisation

par

Philippe DAX

collection animée
par Richard SCHOMBERG

EYROLLES

61, Boulevard Saint-Germain - 75005 Paris

1982

Préface

Le système CP/M très populaire aux États-Unis est devenu, au début des années 80, un véritable "standard" pour les micro-ordinateurs à base de microprocesseurs huit bits. Son influence s'exerce tout particulièrement sur la micro-informatique individuelle et domestique, mais aussi sur l'informatique professionnelle et la gestion des PME.

Ce succès, CP/M le doit à lui-même, mais encore à la grande variété de produits logiciels qui ont été développés autour de lui au bon moment. Cette vaste gamme de produits, dont les domaines d'application sont étendus et diversifiés, couvre: les langages de programmation, le traitement de texte, le développement de logiciel, les progiciels de gestion, les communications inter-ordinateurs et les réseaux.

Le "raz de marée CP/M" qui a déferlé Outre-Atlantique a déjà bien entamé sa percée en Europe et tout particulièrement en France.

C'est pour cette raison qu'il m'a semblé souhaitable de mieux faire connaître ce produit, ses extensions et son environnement logiciel à tous ceux qui désirent l'étudier, l'expérimenter, et le pratiquer.

Cet ouvrage est découpé en quatre parties: le système mono-utilisateur CP/M, le système multi-utilisateurs MP/M, les extensions actuelles nées de CP/M, de MP/M et de l'avènement des microprocesseurs 16 bits, et enfin un aperçu sur les principaux produits développés autour de ce que l'on pourrait appeler "la famille CP/M".

Pour tenter de répondre aux besoins et exigences de chacun, une approche plus générale, relative à la structure fonctionnelle des systèmes, est faite au début de chacune des trois premières parties, pour étayer davantage l'aspect descriptif et "guide d'utilisation".

Bien que ce livre n'ait pas la prétention d'être exhaustif sur l'ensemble des commandes standards CP/M et des produits qui gravitent autour de lui, je pense que le lecteur, qu'il soit amateur, étudiant ou professionnel y trouvera les éléments qu'il recherche.

Table des matières

1
Le système CP/M

1.1. INTRODUCTION

CP/M (Control Program for Microcomputer) est un système d'exploitation mono-utilisateur universellement répandu dans le monde de la micro-informatique.

Le système d'exploitation CP/M est destiné aux micro-ordinateurs à base de microprocesseurs à mots de huit bits de type 8080 et 8085 d'Intel et Z 80 de Zilog. CP/M équipe d'autres microprocesseurs et même des " 16 bits ". On reviendra plus loin à l'étude de ces cas particuliers ainsi qu'aux évolutions actuelles et futures des produits CP/M et de sa famille. Pour le moment, on se limitera à l'étude du système CP/M classique, c'est-à-dire des versions commercialisées (releases) sous les libellés : CP/M 1.4, CP/M 2.0 et CP/M 2.2.

Avant d'aborder en détail la description, le fonctionnement et les facilités offertes du système CP/M, revenons légèrement en arrière pour procéder à un bref historique afin de mieux situer CP/M.

1.1.1. Historique du CP/M

CP/M a été conçu en 1973 par Gary Kildall, à une époque où la micro-informatique, qui existait encore à l'état embryonnaire, était réservée à quelques initiés. L'idée de G. Kildall, fut de concevoir un système d'exploitation répondant aux critères suivants :

— adaptabilité à n'importe quel micro-ordinateur bâti autour de microprocesseurs de type 8080, 8085 ou Z 80;

— possibilité de stockage et d'archivage des données et des programmes utilisateurs sur des mémoires auxiliaires à faible coût (disquettes ou floppy-disques);

— fourniture d'une interface logicielle, complète et standard, avec toutes les fonctions nécessaires à un programmeur.

Une phrase de G. Kildall exprime assez bien ce qu'il a voulu faire de CP/M: "CP/M est le pendant logiciel du bus S-100".

Kildall fonde Digital Research en 1976, date à partir de laquelle est entrepris le développement et la commercialisation de CP/M.

1.1.2. Évolution actuelle de CP/M

Actuellement le système CP/M a pris une telle ampleur, qu'il est devenu de facto un "standard" incontestable (comme le bus S-100 sur le plan matériel).

On évalue à plus de trois cent mille le nombre d'utilisateurs du système CP/M, et on compte pas moins de quatre à cinq cents sociétés OEM (Original Equipment Manufacturer), dont plusieurs dizaines en France, qui ont développé des micro-ordinateurs fonctionnant sous CP/M. Parmi les principaux constructeurs ayant adapté CP/M, on trouve: Altos, Digital Equipment, Hewlett Packard, IBM, ICL, Intel, ITT, Mostek, National Semiconductor, REE, Sharp, Xerox, Zenith, Zilog,...

Pourtant, ce système, bien qu'universellement connu, a lui aussi ses détracteurs. Sur certains points, des critiques peuvent lui être adressées, tant sur la philosophie du système lui-même, que sur la réalisation concrête de certains éléments de celui-ci. Mais le succès l'emporte globalement sur les insuffisances. G. Kildall le dit lui-même: "Je connais de vrais spécialistes en système qui, après avoir analysé à la loupe CP/M, n'ont pas été impressionnés". A proprement parler CP/M n'est pas un modèle du genre, mais c'est un système qui "accroche", plaît et bénéficie d'une vaste bibliothèque de programmes, allant des langages de programmation aux progiciels, en passant par des outils logiciels très

diversifiés tels que traitement de texte, communications entre ordinateurs, bases de données, etc...

L'avenir de CP/M, vu sa facilité d'adaptation, a toutes les chances d'être florissant, car il entre dans le "créneau" bien défini de la micro-informatique individuelle, quelle soit domestique ou professionnelle.

Par contre, CP/M risque d'être concurrencé avec l'apparition et la montée des systèmes multi-utilisateurs à usage général (general purpose), de type UNIX. Ces systèmes, destinés à gérer plusieurs utilisateurs simultanément, équipent des micro-ordinateurs plus puissants à base de microprocesseurs 16 et 32 bits.

Face à cette concurrence inévitable, Digital Research a trouvé la parade avec des produits issus de CP/M et compatibles avec lui tels que MP/M, MP/M-II, CP/M-86, MP/M-86, CP/NET, MP/NET, etc..., produits que nous analyserons dans la suite de ce livre.

	mono-utilisateur mono-tâche	mono-utilisateur multi-tâches	multi-utilisateurs multi-tâches	réseau
8 bits	CP/M	MP/M	MP/M MP/M-II	CP/NET MP/NET
16 bits	CP/M-86	MP/M-86 CCP/M-86	MP/M-86	CP/NET-86 MP/NET-86

Fig. 1 — *Répartition de la famille CP/M*

1.2. LE MATÉRIEL REQUIS PAR CP/M

1.2.1. Microprocesseurs

La conception modulaire de CP/M, lui permet de s'adapter à des machines réalisées à base de microprocesseurs du type 8080, 8085 et Z 80, possédant chacun un code opératoire commun (jeu d'instructions machine).

L'ensemble des modèles de base du CP/M (CCP, BDOS, BIOS), dont l'analyse est détaillée plus loin, sont écrits soit en assembleur 8080, soit en langage structuré PL/M ; le résultat produisant du langage machine 8080 (sous-ensemble des langages machine du 8085 et Z 80). Le Z 80 plus puissant au niveau du code opératoire que celui du 8080 sera donc sous utilisé. Mais rien n'empêche l'utilisateur d'utiliser les instructions du Z 80 si le micro-ordinateur est équipé de ce microprocesseur. Le système CP/M est dit "machine dependent", c'est-à-dire dépendant du CPU à cause du langage machine. CP/M n'est donc pas transportable sur n'importe quelle machine, comme le sont théoriquement les systèmes UCSD et UNIX, qui eux sont écrits en langage évolué: Pascal pour UCSD et C pour UNIX.

Pour CP/M, on ne peut parler de transportabilité des produits qu'entre machines évoluant dans un contexte CP/M. Dans ce seul cas les objets ou les binaires peuvent être transportés d'une machine vers une autre.

La profusion des micro-ordinateurs fonctionnant sous CP/M, tend à prouver que CP/M est un système facilement implantable, malgré le handicap des CPUs imposés. En fait CP/M a l'atout d'être indépendant de la configuration matérielle qui entoure le microprocesseur. C'est cet environnement que nous allons décrire.

1.2.2. La mémoire centrale

Le système lui-même n'occupe que 6,5 Kilo-octets en mémoire vive RAM (Random Access Memory). Mais un certain nombre de produits standards (éditeur de texte ED, debugger DDT, assembleur ASM, etc), fonctionnant dans le contexte CP/M et livrés avec, nécessitent un minimum de 16 Kilo-octets. La taille mémoire peut s'étendre jusqu'à 64 Kilo-octets en configuration maximum.

1.2.3. La mémoire secondaire

CP/M nécessite en outre une mémoire dite secondaire qui n'est autre qu'un support magnétique adressable directement ; c'est-à-dire un ou plusieurs disques. Habituellement les systèmes de micro-informatique individuels sont équipés de disquettes ou disques souples.

1.2.3.1. Les disquettes ou disques souples

Il existe deux types de disquettes reconnaissables par leur encombrement: les disquettes au diamètre de 5 pouces 1/4 et les disquettes 8 pouces à caractère plus professionnel. La capacité des disquettes peut donc varier selon leur diamètre.

Une autre caractéristique importante des disquettes est la densité. On en distingue généralement deux: la simple densité (26 secteurs par piste) et la double densité (48 ou 52 secteus par piste). De plus, pour certaines disquettes, il est possible d'utiliser les deux faces magnétiques, ce qui accroît d'autant plus la capacité globale de stockage.

Les disquettes 8 pouces sont compatibles au format IBM 3740 pour améliorer la portabilité des produits. Le formattage des disquettes CP/M est en effet le même que celles d'IBM, mais les informations utiles ne sont pas compatibles (organisation des catalogues et des fichiers, données en EBCDIC, ...). Des utilitaires permettent cependant de convertir leur contenu d'un format à l'autre.

1.2.3.2. Les disques durs

Des disques "durs" ou rigides peuvent aussi remplacer les disquettes, donnant ainsi des capacités très confortables; capacités allant de 5 à 40 Méga-octets (type Winchester ou Cynthia). Comme pour les disquettes il existe plusieurs formats de disques durs: 5 pouces 1/4, 8 pouces et 13 pouces. Le disque dur se trouve dans une enceinte hermétique pour éviter tout contact avec la poussière. Bien que les bras d'accès aux pistes soient mobiles, le disque est fixe et intégré à la machine. Il existe pourtant des disques durs avec une partie amovible et une partie fixe, permettant ainsi de changer de "pack" de disque si le besoin s'en fait sentir.

1.2.4. La console, poste de travail intéractif

Le système CP/M étant par définition un système mono-poste, il impose la présence d'un terminal de type conversationnel. Ce terminal, appelé aussi "console" dans la terminologie informatique, est le plus souvent un terminal du type "CRT" (Cathode Ray Tube), c'est-à-dire un ensemble écran-clavier alphanumérique.

1.2.5. Les périphériques annexes

Selon la configuration matérielle adoptée sur le micro-ordinateur, celui-ci peut disposer de "ports" d'entrées-sorties spécifiques ou auxiliaires. En particulier on peut trouver une sortie imprimante sur un "port" série ou sur un "port" parallèle (8 bits). D'autres "ports" séries ou interfaces RS232 C (avis V24 du CCITT) peuvent faire partie de l'ensemble, soit en standard soit sur option.

1.3. DESCRIPTION FONCTIONNELLE DU CP/M

Sur toute machine où est implanté le CP/M existe un "cold start loader", système minimal en "PROM" qui réalise le "bootstrap" (amorçage ou démarrage du système). Cette opération initialise le coupleur de disquettes et charge en mémoire "RAM" les modules du système d'exploitation (CCP, BDOS, BIOS).

Le système CP/M est donc constitué de trois modules fonctionnels :

— CCP : Console Command Processor : processeur de commande console ;

— BDOS : Basic Disk Operating System : système de gestion de base des fichiers du disque ;

— BIOS : Basic Input Output System : système d'entrées-sorties de base.

On peut considérer que le CCP est l'interface homme/machine, le BDOS l'ensemble des modules de gestion logique de la mémoire secondaire, et le BIOS le noyau physique du CP/M.

L'ensemble conjugué du CCP et du BDOS constitue le noyau logique du système CP/M, fourni par Digital Research, indépendant de l'environnement externe donc, transportable sur une machine à base de 8080 ou Z 80.

Par contre le BIOS, qui contient les programmes communiquant directement avec les unités physiques, est écrit par le constructeur du micro-ordinateur, pour l'adapter à l'environnement matériel.

1.3.1. Le CCP

Le CCP est découpé en deux parties fonctionnelles :

— l'interpréteur de commandes,
— les commandes intégrées.

1.3.1.1. L'interpréteur de commandes

Le CCP est essentiellement un interpréteur de commandes. En effet il lit sur la console les ordres tapés par l'utilisateur, et en fait l'analyse syntaxique avant de lancer l'exécution du programme. Pour ce faire, le CCP utilise les fonctionnalités des autres modules du système : le BIOS et le BDOS. C'est en liaison avec le BIOS que le CCP réalise le dialogue avec la console utilisateur. Certains caractères spéciaux sont filtrés pour offrir à l'utilisateur des fonctions d'édition : effacement d'un caractère ou d'une ligne, fin de message, fin de ligne, tabulation, etc...

Une fois la commande acceptée sur le plan syntaxique, le CCP charge le fichier du type COM dans la zone TPA réservée au programme utilisateur. Dès que le contrôle est donné au programme utilisateur, le CCP devient inutilisable pendant la phase d'exécution de celui-ci. Certains programmes très volumineux peuvent alors utiliser le fait que le CCP soit inactif pour déborder de la zone TPA et venir recouvrir une partie du CCP contiguë au TPA. Dans ce cas de figure le CCP est détruit par "écrasement", mais de manière temporaire. En effet toute terminaison d'un programme en cours d'exécution, quelle soit normale ou autoritaire (<ctrl-C> tapé au clavier), réalise une réinitialisation du système, appelée "démarrage à chaud" ("warm start"). Cette réinitialisation ramène en particulier le CCP du disque vers la mémoire, et une nouvelle invitation à taper une commande est alors transmise à l'utilisateur.

L'invitation à taper une commande est signalée par l'apparition sur le terminal d'un "préfixe" ou "prompt" de la forme suivante :

A> où "A" représente le nom du disque courant.

1.3.1.2. Conventions d'utilisation des commandes

Dans la suite de ce livre, lorsqu'il s'agira d'illustrations par des exemples, pour différencier ce qui est émis par la machine ("prompt",

réponses) de ce qui est tapé au clavier par l'usager (commandes), nous adopterons les conventions suivantes :

— Toute commande tapée au clavier sera soulignée.

— Le retour chariot noté <CR>, exigé pour terminer une commande ne sera pas figuré, sauf dans le cas où une clarification s'avèrera nécessaire.

— Les caractères spéciaux, obtenus en appuyant simultanément sur la touche "CONTROL" et sur le caractère spécifié, seront représentés par les symboles "ctrl" ou "↑" suivis du caractère en question, exemple : <ctrl-C> ou ↑C.

1.3.1.3. Syntaxe d'une commande

Une commande, qu'elle soit système ou utilisateur, se présente sous la forme d'une suite de chaînes de caractères alphanumériques, séparés éventuellement par des espaces (blancs) et terminée obligatoirement par un retour-chariot (touche "Return" du clavier). La chaîne de caractères qui suit immédiatement le " prompt" représente le nom proprement dit de la commande, et les chaînes suivantes les arguments ou les paramètres associés.

Il existe deux catégories de commandes sous CP/M : les commandes résidentes intégrées au système dans le module CCP, et les commandes non résidentes enregistrées sur disque sous la forme de fichiers standards ayant le type COM qui leur octroie le statut de fichiers exécutables.

Les arguments peuvent représenter, soit des noms de fichiers (voir la section "référence à un fichier"), soit des mots réservés ou des valeurs numériques.

Dans le cas des commandes non résidentes, il est possible de spécifier le nom du disque sur lequel se situe le fichier exécutable associé, sans préciser son type COM qui est implicite.

L'exemple ci-dessous :

```
A>B:STAT    A:FICHIER.ASM
```

montre que l'utilisateur est allé chercher le fichier "STAT.COM" sur le disque "B", pour connaître des renseignements concernant le fichier "FICHIER.ASM" sur le disque "A".

Le CCP accepte indifféremment les lettres majuscules et minuscules en entrée, et les convertit systématiquement en majuscules.

Sur le plan du fonctionnement interne, le CCP exécute directement la commande si celle-ci est résidente, sinon recherche sur le disque courant le nom associé du fichier exécutable de type COM et charge ce fichier en mémoire dans la zone TPA, puis éventuellement s'occupe d'ouvrir le premier fichier mis en argument.

1.3.1.4. Sélection d'un disque

Chaque disque sous CP/M contient un catalogue (directory) qui lui est propre. Ce catalogue permet de répertorier l'ensemble des fichiers présents sur le disque.

Si le système dispose de plusieurs supports disques (disquettes ou disques durs), l'utilisateur peut facilement sélectionner le support sur lequel il désire travailler. Il suffit de taper le nom du disque après le "prompt": lettres A, B, ..., P suivi du caractère ":", pour les disques numérotés 0, 1, ..., 15.

Exemple de commutation de disques:

A>B:	l'utilisateur sélectionne le disque "B".
B>	nouveau "prompt" émis par le CCP.
B>A:	l'utilisateur revient sur le disque "A".
A>	

1.3.1.5. Notion d'utilisateur

Il n'existe pas de notion de propriétaire sous la version 1.4 de CP/M et n'importe quel utilisateur, peut accéder à tous les fichiers du ou des disques présentement connectés. La notion de propriété de l'information est attachée à celle de propriété physique du support.

Une amélioration a été portée sous la version CP/M 2.2 pour permettre à plusieurs utilisateurs d'avoir leur propre catalogue sur le même support, dans la mesure où ceux-ci se partageraient à tour de rôle le micro-ordinateur et l'espace disque.

L'espace disque est en effet cloisonné en plusieurs partitions logiques appelées "zones utilisateurs". Ce découpage en zones est tota-

lement transparent à l'utilisateur car le cloisonnement n'est pas réalisé physiquement sur le support, mais logiquement au niveau du catalogue général. Le premier octet de chaque entrée étant alors associé au numéro de l'utilisateur (0 par défaut). Un utilisateur qui se connecte, peut alors choisir sa partition à l'aide de la commande :

A>USER n

où n représente un numéro de zone utilisateur allant de 0 à 15.

La protection de l'information n'est pas effective car cette commande est accessible à n'importe quel utilisateur, et la notion de "mot de passe" n'existe pas. Cela peut être vu comme une déficience, mais il faut bien noter que CP/M est conçu pour être un système d'exploitation mono-utilisateur, et que plusieurs usagers ne peuvent pas se partager la machine simultanément.

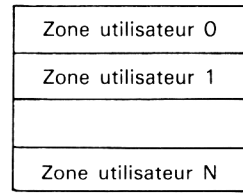


Fig. 2 — *Représentation logique des zones utilisateurs*

1.3.1.6. Référence à un fichier

On accède aux fichiers gérés par le BDOS au moyen d'un identificateur dont la forme générale comprend trois champs de chaînes de caractères, sous la forme suivante :

disque:fichier.type *exemple* : A:EXEMPLE.CPM

Cet identificateur représente une référence de fichier qui peut être soit explicite soit ambiguë.

Une référence explicite à un fichier identifie uniquement un seul et même fichier, tandis qu'une référence ambiguë peut satisfaire plusieurs fichiers différents.

Le premier champ représente le nom du disque sur lequel l'utilisateur désire se positionner. Il est symbolisé par une seule lettre (de 'A' à 'P'); la lettre "A" étant associée au disque 0 supportant le système CP/M, la lettre "B" au disque 1, ..., la lettre "P" au disque 15.

Cet identificateur n'est pas obligatoire dans la mesure où le fichier se trouve sur le disque courant. La référence à un fichier peut alors être du type :

fichier.type *exemple* : FILE.ASC

Les deux autres champs représentent le nom principal du fichier et son type ou extension. Le champ "type" est optionnel, mais il est utilisé généralement pour décrire ce que contient le fichier, et à ce titre il peut être très utile. Par exemple ASM pourra désigner un programme écrit en assembleur, OBJ un module objet, COM une commande exécutable, etc...

Les deux champs du nom sont séparés par un "." comme il est montré ci-dessous :

pppppppp.ttt *exemple* NOMFICH.TEX

où "pppppppp" représente le nom principal (de un à huit caractères), et "ttt" le type du fichier (un à trois caractères). Comme on l'a déjà mentionné, le nom :

pppppppp *exemple* : PROGRAM

est aussi accepté et son type par défaut est équivalent à trois blancs consécutifs.

Un nom de fichier explicite ne doit pas contenir les caractères spéciaux suivants :

. , ; = ? * < > [ ]

alors que tous les autres caractères alphanumériques et les autres caractères spéciaux sont autorisés.

+++---	est correct	AZE;QWE	est incorrect
!A"Z#E$R	est correct	A.Z,E;R=	est incorrect
(azerty)	est correct	<qwerty>	est incorrect

Une référence ambiguë peut identifier un ou plusieurs fichiers, elle est généralement utilisée pour rechercher un fichier dans un catalogue, jusqu'à ce qu'une coïncidence se présente avec une entrée du catalogue. La forme d'une référence ambiguë est semblable à celle d'une référence explicite, à la seule différence que les caractères "?" et "*" ont des fonctions particulières que nous allons expliciter ci-après.

Le caractère "?" permet de remplacer n'importe quel autre caractère permis dans un nom de fichier, à la position indiquée par ce caractère. Ainsi la référence ambiguë suivante:

N?M.T??

satisfait aux noms de fichiers suivants (s'ils existent):

NOM.TEX	? ⟶ O	et ?? ⟶ EX
NUM.TYP	? ⟶ U	et ?? ⟶ YP
NAM.TTY	? ⟶ A	et ?? ⟶ TY

mais ne satisfait pas les noms de fichiers suivants:

RAM.TPA	R faux
NAME.TTY	E faux
NAM.HEX	H faux

Le caractère "*" quant à lui, a la même signification que "?", mais au lieu de remplacer un seul caractère à une position donnée, il peut remplacer tout un champ, ainsi:

.

est équivalent à la référence ambiguë au fichier:

????????.???

c'est-à-dire à tous les fichiers présents sur le disque sélectionné. De même:

pppppppp.*	est équivalent à:	pppppppp.???
*.ttt	est équivalent à:	????????.ttt

L'utilisation simultanée de "?" et "*" est également possible, par exemple: NOM??.*

1.3.1.7. Types prédéfinis

Il existe un certain nombre de types prédéfinis sous CP/M qui permettent de bien spécifier ce que représente le contenu du fichier: source

symbolique, binaire absolu ou translatable, données numériques, information temporaire, etc...

La liste ci-dessous montre les principaux standards de fichiers exploitables sous CP/M, bien qu'elle ne soit pas exhaustive.

COM	: commande exécutable	HEX	: code machine format Intel
ASM	: source assembleur 8080	PRN	: fichier image imprimante
BAK	: source "backup" de ED	$$$	: fichier temporaire sous ED
BAS	: source BASIC	PAS	: source PASCAL
COB	: source COBOL	FOR	: source FORTRAN
REL	: module objet relogeable	INT	: module objet intermédiaire

1.3.1.8. Les caractères de contrôle d'édition

Le module CCP permet en outre de gérer la procédure "appareil" avec le terminal en traitant certains caractères de contrôle du clavier, correspondant à des fonctions de saisie ou d'édition.

<rubout>	: effacement du caractère précédent ;
<ctrl-H>	: effacement du caractère précédent (en CP/M 2.2.) ;
<ctrl-U>	: effacement d'une ligne entière ;
<ctrl-X>	: idem à <ctrl-U> ;
<ctrl-R>	: re-imprime la ligne courante d'une manière propre ;
<ctrl-M>	: fin de ligne (retour chariot : <CR>) ;
<ctrl-J>	: line-feed : <LF> ;
<ctrl-E>	: passage à la ligne suivante sans transmission de la ligne au CCP ;
<ctrl-I>	: mise en œuvre de la tabulation, par tacquets de 8 colonnes ;
<ctrl-Z>	: fin de fichier (utilisé pour ED et PIP) ;
<ctrl-C>	: relance du système par un appel au "warm start" ;
<ctrl-P>	: écho des entrées-sorties console sur l'imprimante, fonctionne en bascule (validation et invalidation) ;
<ctrl-S>	: suspension de l'impression sur la console.

1.3.1.9. Commandes intégrées au CCP

En plus des fonctions de dialogue, le CCP contient cinq ou six commandes résidentes d'utilisation courante, dont la mise en œuvre et l'utilisation seront explicitées plus loin :

- **DIR** : impression des noms des fichiers du catalogue
- **ERA** : destruction d'un fichier
- **REN** : changement de nom d'un fichier

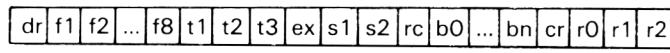
1.3.2.3. Le bloc de contrôle de fichier FCB

Sur le plan externe, c'est-à-dire celui de l'utilisateur, un fichier est repéré par son nom, son type et le nom du disque sur lequel il se trouve.

Sur le plan interne, c'est-à-dire celui du système, un fichier est associé à un certain nombre de paramètres : adresse physique des blocs d'allocation, compteurs d'enregistrement, ...

L'ensemble de ces informations, externes et internes qui caractérisent totalement le fichier, sont regroupées dans une zone mémoire appelée FCB (File Control Block). La taille d'un FCB est de 33 octets pour les fichiers à accès séquentiel et de 36 octets pour les fichiers à accès direct. Il n'y a pas de limitation sur le nombre de FCB utilisables dans un programme. Il existe un FCB par défaut, généralement traité par le CCP, à l'adresse 005CH.

La structure d'un FCB est la suivante :



00 01 02 ... 08 09 10 11 12 13 14 15 16 ... 31 32 33 34 35

dr	numéro du disque (0 - 16) 0 ⟹ disque courant pris par défaut 1 ⟹ disque A, 2 ⟹ disque B, ... 16 ⟹ disque P.
f1...f8	nom du fichier en ASCII majuscule.
t1, t2, t3	type du fichier complété à blanc éventuellement.
ex	numéro courant de l'extension du fichier, initialement à 0.
s1, s2	réservés au BDOS.
rc	compteur d'enregistrement par extension (0 - 128).
b0...bn	carte d'allocation disque des blocs du fichier.
cr	numéro d'enregistrement courant en mode séquentiel.
r0, r1, r2	numéro d'enregistrement optionnel en mode direct (0 - 65535).

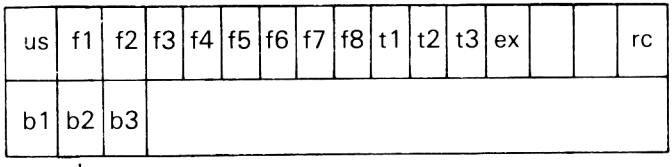
1.3.2.4. Notion de catalogue de fichiers

Pour chaque fichier en ligne, le BDOS gère un catalogue appelé "directory" qui contient le nom du fichier et les informations nécessaires à son accès (taille en blocs, adresse de chaque bloc, ...).

Chaque entrée de ce catalogue (128 entrées maximum) a une taille de 32 octets. Les 16 premiers octets représentent la copie conforme des 16 premiers octets du FCB au moment de la fermeture du fichier, nom du fichier, type du fichier et compteur d'enregistrement, à l'exception près que le premier octet n'exprime plus le numéro du disque associé au fichier, car le catalogue est unique pour ce disque, mais le numéro de l'utilisateur (voir commande USER sous CP/M 2.2).

Les 16 octets suivants représentent la "MAP" d'occupation du disque par le fichier. A chaque octet de la "MAP" est associé un numéro de bloc de nK-octets suivant la génération système. Le catalogue occupe lui-même un ou deux blocs selon les configurations disques. L'allocation du premier fichier sur le disque sera alors contiguë au catalogue, et le numérotage commencera à partir de 1 ou 2 selon la configuration.

Structure d'une entrée de catalogue:



numéros relatifs par rapport à la fin du catalogue des blocs d'allocation sur disque du fichier concerné.

Les recherches dans le catalogue sont facilitées, du fait qu'il s'agira d'opérer par comparaisons successives de chacune des entrées avec le FCB utilisateur, sur le champ symbolique: nom du fichier.

1.3.2.5. Fonctions du BDOS

Le BDOS fournit à l'utilisateur et aux autres parties du système de nombreuses routines de service, dont:

— la réinitialisation du CP/M ;
— la sélection d'un disque ;
— la création, l'ouverture et la fermeture d'un fichier ;
— la lecture ou l'écriture d'un enregistrement ;
— la recherche d'une entrée dans le catalogue d'un disque ;
— la création et la destruction d'entrées dans le catalogue ;
— le changement de nom d'un fichier ;
— l'initialisation de l'adresse DMA (accès direct mémoire) ;
— la lecture de vecteurs (disques en ligne, disque courant) ;
— la protection temporaire d'un disque en écriture ;
— le positionnement d'un attribut de fichier (protection) ;
— lecture, écriture, positionnement d'enregistrement en direct.

1.3.3. Le BIOS

Le BIOS contient l'ensemble des programmes d'entrées-sorties (drivers), spécifiques de la configuration matérielle adoptée.

1.3.3.1. Adaptabilité

L'ensemble de ces programmes liés à l'environnement externe est généralement conçu et écrit par le constructeur du micro-ordinateur. Le "source" du BIOS, écrit en langage machine (assembleur ou macro-assembleur), est en principe fourni à l'acquéreur, dans l'éventualité où celui-ci aurait l'intention de modifier, soustraire ou rajouter des programmes spécifiques en fonction de ses besoins propres (entrées-sorties sur périphériques non standards par exemple).

C'est peut-être une des raisons pour laquelle le système CP/M est si facilement adaptable pour une configuration donnée, bien que cela nécessite quelques connaissances approfondies dans le domaine du fonctionnement des organes d'entrées-sorties, ainsi qu'une pratique de la programmation en assembleur.

Le BIOS est un module indépendant des autres modules (CCP et BDOS) ; pourtant il leur fournit les primitives d'entrées-sorties indispensables à leur fonctionnement.

1.3.3.2. Interfaçage

Une interface entre le BIOS et le reste du système, définie par des spécifications bien précises, permet d'établir la liaison entre les autres modules du système : le BDOS et le CCP.

Cette interface normalisée est constituée d'une séquence de 17 instructions de branchement vers les fonctions spécifiques. La séquence de "JMP" appelée "jump vector" (vecteur de branchement), est située à la base du BIOS. Notons que les paramètres des séquences d'appel au BIOS, ainsi que le retour des résultats, doivent strictement respecter les spécifications imposées par Digital Research.

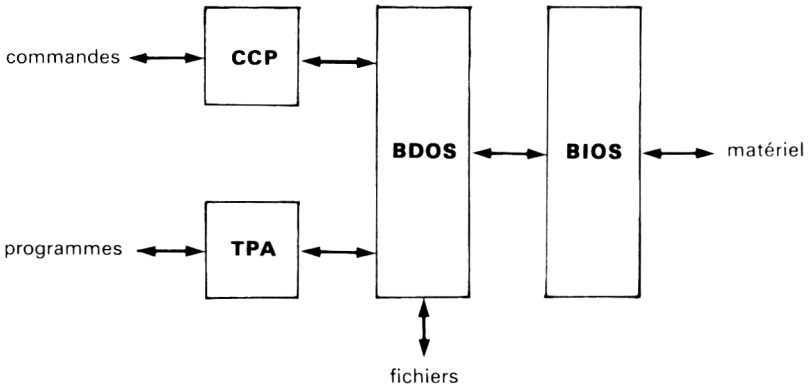


Fig. 4 — *Architecture des modules du CP/M*

1.4. STRUCTURE DU SYSTÈME CP/M

1.4.1. Structure mémoire

La mémoire sous CP/M est divisée en cinq zones, comme le montre la figure ci-après. Le chargeur spécifique du système, implante les trois modules du CP/M en mémoire haute. Le CCP est référencé par l'adresse CBASE, le BDOS par l'adresse BIAS, le BIOS occupant la fin de la mémoire. Le bas de la mémoire, situé de 0000H à 00FFH (soit 256 octets), est réservé au système. Enfin la dernière zone appelée "TPA", qui débute à l'adresse TBASE, est disponible aux programmes utilisateurs.

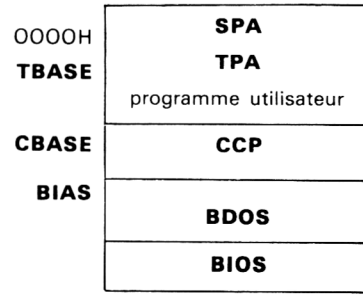


Fig. 5. — *Implantation mémoire des modules de CP/M*

1.4.1.1. Le TPA

Le TPA (Transient Program Area) est la plus grande partie de la mémoire disponible pour l'exécution des programmes. Ces programmes peuvent être soit des programmes utilisateurs, soit encore des programmes système non résidents du type "COM".

Cette zone est située entre les adresses 0100H (hexadécimal) et CBASE (base du CCP). Dans le cas de certains micro-ordinateurs qui possèdent des programmes pré-enregistrés en PROM, l'adresse du début du TPA est alors translatée en 4300H au lieu de 0100H. La taille du TPA est extensible selon la configuration mémoire adoptée 16, 32, 48 ou 64K. L'adresse du début (TBASE = 0100H ou 4300H) restant fixe, c'est l'adresse du CBASE qui varie alors d'un multiple de 16K (4000H).

De plus CP/M offre la possibilité d'implanter des programmes volumineux allant au-delà de l'adresse CBASE en recouvrant le CCP. C'est le "warm start", déclenché à la fin de l'exécution du programme utilisateur ou par l'intervention de l'usager (frappe du caractère spécial <ctrl-C>) qui ramènera automatiquement le CCP en mémoire.

1.4.1.2. Le SPA

Le SPA (System Parameter Area) est la zone réservée au système CP/M pour ranger certains paramètres et vecteurs de branchement.

Les emplacements de code et de donnée de cette zone sont détaillés ci-dessous :

adresses	description des contenus
0000H-0002H	Contient une instruction JMP vers le point d'entrée du "warm start" (réinitialisation à chaud) qui permet à l'utilisateur de programmer un retour moniteur (JMP 0000H) à la fin d'exécution de son programme.
0003H-0003H	Contient l'octet IOBYTE qui permet de réassigner les périphériques. L'octet IOBYTE est constitué de quatre champs distincts, de deux bits chacun, appelés CONSOLE, READER, PUNCH, LIST : 0003H | LIST | PUNCH | READER | CONSOLE | bits 7,6 bits 5,4 bits 3,2 bits 1,0 Le contenu de chaque champ peut prendre quatre valeurs définissant l'assignation source ou destinatrice de chaque périphérique logique : TTY:, CRT:, RDR:, PUN:, LPT:, BAT:, UC1:, UL1:, UP1:, UR1:, etc... Notons que l'implémentation de la fonction IOBYTE est optionnelle et affecte seulement l'organisation du module BIOS.
0004H-0004H	Contient le numéro du disque courant (0=A, ... 15=P).
0005H-0007H	Contient une instruction JMP vers le point d'entrée du BDOS, qui permet de faire appel aux fonctions moniteur, ou de connaître la dernière adresse disponible pour les programmes utilisateurs (LHLD 0006H).
0008H-003FH	Emplacement des interruptions. L'interruption 7 (JMP en 0038H) est utilisée par les programmes "debug" DDT et SID (arrêts sur adresse, trace).
0040H-004FH	Zone de 16 octets réservée au BIOS, mais non utilisée sous CP/M standard.
0050H-005BH	Zone réservée peu utilisée.
005CH-007CH	Bloc de contrôle de fichier (FCB) par défaut, fourni au programme utilisateur par le CCP.
0080H-00FFH	Tampon implicite de 128 octets réservé aux entrées-sorties disque et console.

1.4.2. Structure disque

Pour charger un système CP/M en mémoire ("cold start" ou démarrage à froid, le disque système doit se trouver sur le "drive" A. Ce disque contient le système CP/M proprement dit, le catalogue associé au disque, et les fichiers répertoriés par le catalogue.

La commande DIR permet de lister l'ensemble des noms des fichiers présent sur le disque.

```
A>DIR
A: MOVCPM  COM : SUBMIT  COM : ED      COM : ASM     COM
A: PIP     COM : LOAD    COM : STAT    COM : SYSGEN  COM
A: DDT     COM : MODE    COM : DUMP    ASM : CBIOS   ASM
A: DUMP    COM : CBIOS64 COM : CBIOS48 COM : CBIOS32 COM
A>
```

1.4.2.1. Emplacement du CP/M : les pistes réservées

Le système CP/M résident sur disque n'est pas représenté sous la forme d'un fichier standard ou même particulier. On constate en effet si l'on détruit tous les fichiers présents sur le disque, que l'opération de "bootstrap" du système (démarrage à froid ou "cold start") fonctionne normalement. La commande DIR ne visualisera alors aucun nom de fichier.

```
A>DIR
NO FILE
A>
```

C'est par mesure de protection que le système CP/M n'apparaît pas au niveau utilisateur, pourtant il est bien présent sur le disque.

Les deux premières pistes du disque sont réservées aux modules du système d'exploitation proprement dit. Chaque piste (numérotée à partir de 0), contient 26 secteurs (numérotés à partir de 1), pour une disquette simple densité.

Le secteur 1 de la piste 0 contient le "bootstrap" ou "cold start loader", dont le rôle est de charger le reste du système (CCP, BDOS, BIOS) en mémoire à partir de l'adresse CBASE.

L'adresse CBASE du CCP est calculée, en fonction de l'espace mémoire disponible, de la manière suivante :

capacité	CBASE
16 K	2400H
32 K	6400H
48 K	A400H
64 K	E400H

Les allocations sur disque et en mémoire des différents modules du système CP/M, ainsi que leur correspondance, sont illustrées par le schéma suivant :

piste	secteur	adresse mémoire	module CP/M
00	01	adresse du boot	BOOT
00 .. 00	02 .. 17	CBASE .... CBASE+0780H	CCP
00 .. 01	18 .. 19	CBASE+0800H ..... CBASE+1580H	BDOS
01 .. 01	20 .. 26	CBASE+1600H ..... CBASE+1900H	BIOS
02 .. 76	01 .. 26		Catalogue et fichiers

Fig. 6 — *Correspondance disque-mémoire des modules CP/M*

1.4.2.2. Emplacement du directory et des fichiers

Le catalogue du disque (ou directory) débute à l'adresse piste 2, secteur 1. Il occupe 32 secteurs maximum, découpés en zones de 32 octets appelées "entrées"; soit quatre entrées par secteur ou bien encore 128 entrées par disque. Une entrée étant la copie des 32 premiers octets du FCB lors de la fermeture du fichier (close).

La recherche d'un fichier se fera donc en balayant le catalogue, pour comparer chaque entrée avec le FCB utilisateur.

La zone de stockage des fichiers utilisateur commence à partir de la fin du catalogue.

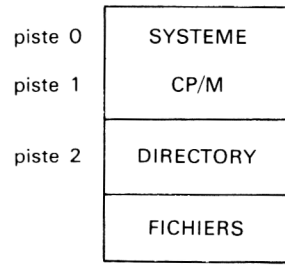


Fig. 7 — *Structure d'un disque système*

1.5. LES FACILITÉS SOUS CP/M : LES COMMANDES

Dans cette partie nous allons quitter les généralités pour nous placer au niveau de l'utilisateur devant son terminal. Nous avons déjà vu qu'une commande ne peut être tapée que si le caractère d'invitation appelé "prompt" est affiché. Une commande peut aussi présenter des arguments sur la même ligne, noms de fichiers en l'occurrence, ou valeurs diverses.

Il existe deux types de commandes sous CP/M : les commandes intégrées au système dans le CCP, dites aussi résidentes, et les commandes standard (livrées avec CP/M) associées à des fichiers exécutables du type "COM". Ces fichiers sont généralement situés sur le disque supportant le système, par raison d'homogénéité.

1.5.1. Les commandes résidentes

Il s'agit de commandes d'intérêt général et d'utilisation très fréquente. De plus, leur faible encombrement a permis de les intégrer avec le CCP en mémoire centrale.

1.5.1.1. DIR

La commande DIR permet de visualiser les noms de fichiers présents sur le disque sélectionné, qui satisfont la référence ambiguë donnée en argument.

Exemples de commandes DIR correctes :

A>DIR	liste tous les noms de fichiers présents sur le disque courant, et est équivalente à la commande : "DIR *.*".
A>DIR B:	liste tous les noms de fichiers présents sur le disque B.
A>DIR *.COM	liste tous les noms de fichiers de type COM.
A>DIR ESS???.B?S	liste les noms de fichiers répondant à cette référence ambiguë. Exemple : "ESSAIS.BAS".
A>DIR B:ESSAIS.*	liste des noms de fichiers génériques du fichier "ESSAIS" présents sur le disque B.

Si aucun fichier n'est trouvé sur le disque sélectionné, le message "NO FILE" ou "NOT FOUND" est affiché à la console.

1.5.1.2. ERA

La commande ERA (Erase) permet de détruire un ou plusieurs fichiers sur le disque sélectionné. Les fichiers à supprimer doivent satisfaire à la référence de fichier donnée en argument sur la même ligne.

En fait les fichiers en cause ne sont pas véritablement effacés physiquement sur le support, mais simplement rendus inaccessibles. La valeur E5H (en hexadécimal) est placée dans le premier octet de l'entrée, dans le catalogue, du fichier à supprimer. Le fait de verrouiller cette entrée permet d'ignorer le nom du fichier et sa "MAP" d'occupation disque ; l'espace occupé sera réutilisable lorsque cette entrée sera prise pour la création d'un autre fichier.

La commande ERA peut s'illustrer avec les exemples donnés ci-dessous :

A>ERA ABC.XYZ	supprime le fichier "ABC.XYZ".
A>ERA *.BAK	supprime tous les fichiers postfixés par BAK.
A>ERA TEST.*	supprime les fichiers de nom principal "TEST".
A>ERA B:EX.ASM	supprime le fichier "EX.ASM" sur le disque B.
A>ERA *.*	supprime tous les fichiers du disque courant. Cette opération peut être fatale si l'usager a tapé cette commande par inadvertance. C'est pour cette raison que le message préventif : "ALL FILES (Y/N)?" est affiché sur la console pour demander la confirmation de l'utilisateur.

1.5.1.3. REN

La commande REN (Rename) permet de changer le nom d'un fichier déjà existant. Les références ambiguës, n'ayant pas de sens dans ce type de commande, ne sont pas autorisées. La syntaxe est la suivante :

REN fichier1=fichier2

"fichier1" représente le nouveau nom du fichier et "fichier2" l'ancien. Ces noms explicites sont du type : "D:FFFFFFFF.TTT" ; avec "D:" représentant le nom du disque (optionnel), "FFFFFFFF" le nom principal du fichier (obligatoire) et ".TTT" le type (optionnel).

Exemple de commandes REN :

A>REN A:NEUF.MAC=A:VIEUX.ASM

A>REN A:APRES.TXT=AVANT

Si le fichier à renommer n'existe pas, le message "NOT FOUND" est renvoyé. De même si le nouveau nom correspond à un fichier déjà présent, le message "FILE EXISTS" est affiché sur la console.

1.5.1.4. SAVE

La commande SAVE permet de sauvegarder les programmes enregistrés en mémoire dans la zone TPA sur le disque, sous la forme d'un fichier. La syntaxe de la commande SAVE est la suivante:

SAVE n fichier

Le premier argument "n" représente le nombre décimal de pages mémoire à sauvegarder à partir du début de la zone TPA. Une page mémoire est équivalente à un bloc contigu de 256 octets ou 100H (en hexadécimal). Le calcul de "n" se fait de la façon suivante: connaissant la dernière adresse du programme en mémoire (voir LOAD, DDT), on obtient la taille en octets en soustrayant 100H à l'adresse de fin, et si le résultat n'est pas un multiple de 100H, on rajoute 1 au chiffre des centaines, et on convertit ce nombre en décimal.

Exemple: soit un programme se terminant en 12C8H, sa taille est 12C8H−100H=11C8H, et le nombre de pages est alors 11+1=12H soit n = 18 en décimal.

Le second argument de la commande représente le nom du fichier dans lequel sera sauvegardé l'image mémoire du programme. En général ce fichier est exécutable et son type devrait être "COM".

Exemples:

A>SAVE 5 PROG.COM — sauvegarde la zone TPA de 100H à 5FFH dans le fichier "PROG.COM" sur le disque courant.

A>SAVE 12 B:SAUVE — sauvegarde la zone TPA de 100H à 0CFFH dans le fichier "SAUVE" sur le disque B.

1.5.1.5. TYPE

La commande TYPE permet de visualiser le contenu d'un fichier "source" codé en ASCII. Le nom donné en argument doit être une référence explicite. Les fichiers du type ASCII sont généralement organisés en suites d'articles séparés par ODH, OAH (retour-chariot, line-feed).

TYPE étend aussi les tabulations horizontales, spécifiées dans le fichier par l'octet 09H ou <ctrl-I>, par huit espaces séparateurs. Les fichiers de type non-ASCII peuvent donner des résultats imprévisibles à l'affichage, selon le type d'écran disponible sur le micro-ordinateur.

Exemples d'utilisation de TYPE:

A>TYPE SOURCE.COB	liste un fichier source écrit en COBOL.
A>TYPE LISTING.PRN	liste le fichier image imprimante.
A>TYPE B:CALCUL.FOR	liste le fichier source "CALCUL" écrit en FORTRAN sur le disque B.

1.5.2. Commandes standard non résidentes

L'ensemble de ces commandes est fourni avec le produit CP/M sur une disquette système "bootable". Ces commandes, comme nous l'avons déjà vu, sont des fichiers du type "COM". Elles ne sont bien sûr pas liées à la disquette système, et peuvent être exploitées sur d'autres disquettes ou disques durs.

Certaines de ces commandes possèdent des sous-commandes qui leur sont propres. On est alors en présence d'utilitaires gérant leur propre dialogue avec l'utilisateur, c'est le cas de ED et DDT.

1.5.2.1. STAT

La commande STAT donne à l'utilisateur des informations sur les fichiers, l'espace de stockage ou les assignations des périphériques. Elle offre en outre la possibilité de changer les affectations des périphériques et les attributs des fichiers. Cette commande admet des arguments sur la même ligne, qui sont fonctions des renseignements que l'utilisateur désire obtenir.

Renseignements sur les supports disque

L'argument est alors le nom du disque (A:,... P:); s'il est absent c'est l'ensemble des disques présents sur la configuration, qui est pris par défaut.

Syntaxes possibles :

STAT calcule l'espace disponible sur les disques actuellement présents sur le système et imprime le message :

```
d: R/W, SPACE: xxxK
```

ou

```
d:R/O, SPACE:xxxK
```

R/W indique que le disque "d:" peut être lu (Read) et écrit (Write), tandis que R/O indique que le disque "d:" est protégé en écriture (Read Only), soit après une demande explicite, soit après un changement de disquette à chaud. La valeur "xxxK"donne la taille de l'espace encore disponible sur la disquette en kilo-octets.

STAT d: calcule uniquement l'espace disponible sur le disque "d:" sélectionné et affiche le message :

```
BYTES REMAINING ON d: xxxK
```

Renseignements sur les fichiers

L'argument qui suit STAT peut être, soit une référence explicite, soit une référence ambiguë de nom de fichier.

```
STAT "nom de fichier"
```

Les noms de fichiers qui satisfont à la référence sont listés par ordre alphabétique, avec leurs caractéristiques de stockage. Une ligne d'en-tête explicite les champs d'information visualisés.

Sous CP/M 1.4 :

```
RECS BYTS EX D:FILENAME.TYP
rrrr  bbbK  ee  d:ffffffff.ttt
```

Le champ "RECS" donne le nombre d'enregistrements "rrrr" de 128 octets attribués au fichier. Le champ "BYTS" donne le nombre de blocs alloués au fichier en kilo-octets : (bbb = rrrr * 128 / 1024). Le champ "EX" donne le nombre d'extensions de 16K du fichier (ee = bbb/16). Vient ensuite le nom du fichier appartenant au disque "d".

En fin d'impression des caractéristiques individuelles des fichiers qui répondent au critère, le total de l'espace encore disponible est renvoyé à l'utilisateur.

Sous CP/M 2.2.:

```
Recs Bytes Ext Acc
rrrr  bbbK   ee   R/W d:ffffffff.ttt
```

Les champs "Recs" "Bytes" et "Ext" s'apparentent aux champs "RECS", "BYTS" et "EX" de la version 1.4. Le champ "Acc" donne en plus le mode d'accès aux fichiers, lequel peut être changé par les commandes de changement d'attribut de fichier, spécifiques à la version 2.2.:

STAT d:fichier.typ $R/W	autorisation d'accès en lecture et écriture pour le fichier nommé.
STAT d:fichier.typ $R/O	protection d'accès en écriture et suppression pour le fichier nommé.
STAT d:fichier.typ $SYS	indicateur "système" pour ne pas faire figurer le nom du fichier lors d'une commande DIR.
STAT d:fichier.typ $DIR	efface l'effet de l'indicateur système "SYS" pour remettre le fichier dans le mode normal.

Lorsqu'un fichier est marqué R/O, toute tentative de suppression du fichier ou d'écriture dans le fichier est rejetée. Le message:

```
Bdos Err on d:fichier R/O
```

est imprimé sur la console, et le système se bloque (attente d'un caractère). La frappe d'un caractère quelconque provoque un "warm start" ainsi que le déblocage.

Assignation de périphériques

Comme dans la plupart des systèmes d'exploitation modernes, l'accès à un organe périphérique ne s'opère pas directement. En particulier, les compilateurs ou autres programmes évolués ne connaissent pas les périphériques physiquement présents sur le système, mais des unités dites "logiques" qui peuvent leur être affectées.

Par exemple, si l'imprimante est en panne ou non connectée, l'usager peut réassigner cette imprimante à la console principale pour visualiser ses résultats, ou bien encore aiguiller la sortie imprimante vers un autre périphérique de son choix.

CP/M sait gérer ces unités logiques, et permet ainsi une plus grande souplesse d'utilisation au niveau des organes d'entrées-sorties.

C'est l'octet IOBYTE à l'emplacement 0003H de la mémoire qui contient les correspondances entre les unités logiques et les unités physiques (voir le chapitre sur le SPA).

A l'exception des disques, il existe quatre périphériques assignables sous CP/M. Ces périphériques ont des noms physiques et logiques prédéfinis.

La commande : "STAT VAL:" (version CP/M 1.4) donne la liste des assignations possibles :

Exemple :

```
A>STAT VAL:
CON := TTY:  CRT :   BAT :  UC1:
RDR := TTY:  PTR :   UR1 :  UR2:
PUN := TTY:  PTP :   UP1 :  UP2:
LST := TTY:  CRT :   LPT :  UL1:
A>
```

où les noms des périphériques logiques sont :

CON	: la console de dialogue avec l'usager,
RDR	: un lecteur papier ou magnétique (cassette),
PUN	: un perforateur ruban ou un enregistreur (cassette),
LST	: une imprimante pour sortie de listings.

et les noms des périphériques physiques :

TTY	: télescriptrice papier (basse vitesse),
CRT	: console (Cathode Ray Tube) écran,
BAT	: organe pour traitement par lots (batch),
PTR	: lecteur de ruban ou de cassette magnétique,
PTP	: enregistreur sur ruban ou sur cassette magnétique,

LPT : imprimante,
UC1 : autre organe du type console,
UR1 : autre organe du type lecteur,
UR2 : autre organe du type lecteur,
UP1 : autre organe du type enregistreur,
UP2 : autre organe du type enregistreur,
UL1 : autre organe du type imprimante.

La commande: "STAT VAL:" (version CP/M 2.2) imprime le sommaire des commandes disponibles.

Exemple:

```
A>STAT VAL:
Temps R/O Disk: d:=R/O
Set Indicator   : d:filemane. typ $R/O $R/W $SYS $DIR
Disk Status     : DSK: d:DSK:
User Status     : USR:
Iobyte Assign:
CON := TTY: CRT : BAT : UC1:
RDR := TTY: PTR : UR1 : UR2:
PUN := TTY: PTP : UP1 : UP2:
LST := TTY: CRT : LPT : UL1:
A>
```

La syntaxe de l'assignation d'un périphérique physique à un périphérique logique est de la forme:

```
STAT logique=physique
```

Exemple:

```
A>STAT CON:=CRT:, LST:=LPT:
```

Les assignations courantes peuvent être visualisées à tout moment. En particulier pour connaître les assignations par défaut, l'usager doit taper la commande "STAT DEV:".

Exemple :

```
A>STAT DEV:
CON: is CRT:
RDR: is PTR:
PUN: is PTP:
LST : is LPT:
```

Protection d'un disque

A tout moment l'utilisateur peut protéger un disque pour éviter de détruire les informations qu'il contient, en cas d'accident éventuel. La syntaxe est la suivante :

```
STAT d:=R/O
```

Le disque est alors protégé en écriture, et toute tentative d'accès, jusqu'à la prochaine déprotection ou jusqu'au prochain "warm start", se solde par l'affichage du message d'erreur :

```
BDOS ERR ON d: READ ONLY
```

Commandes annexes sous CP/M 2.2

```
STAT USR:
```

donne la liste des numéros d'utilisateurs (voir comande USER), qui possèdent des fichiers sur le disque courant :

Active User	: u	numéro de l'utilisateur actif.
Active file	: u1 u2 ... un	numéros des utilisateurs possédant des fichiers sur le disque courant.

La commande :

```
STAT DSK: ou STAT d:DSK:
```

donne les caractéristiques physiques et logiques du disque sélectionné.

Exemple :

```
A>STAT DSK:

       A  :  Drive Characteristics
    1944  :  128 Byte Record Capacity
     243  :  Kilobyte Drive Capacity
      64  :  32 Byte Directory Entries
      64  :  Checked Directory Entries
     128  :  Records/Extent
       8  :  Records/Block
      26  :  Sectors/Track
       2  :  Reserved Tracks
```

Ce disque possède donc une capacité de 1944 enregistrements de 128 octets, soit 243 Kilo-octets. Le catalogue de fichier comprend un maximum de 64 entrées possibles qui ont été vérifiées. Une extension de fichier est équivalente à 128 enregistrements, et un bloc d'allocation correspond à huit enregistrements. La sectorisation du disque est de 26 secteurs par piste, ce qui correspond généralement à un disque formaté en simple densité, deux pistes sont réservées au système CP/M proprement dit.

1.5.2.2. PIP

PIP (Peripheral Interchange Program, programme d'échange entre périphériques), est un utilitaire très puissant de manipulation de fichiers ou de conversion de support sur les unités périphériques. PIP accepte deux types de syntaxe :

```
    PIP
ou
    PIP <commande interne à PIP>
```

Dans le premier cas PIP invite l'utilisateur à taper ses commandes ligne par ligne en imprimant le caractère d'invitation "*". La sortie de PIP s'effectuant alors en tapant une ligne vide c'est-à-dire un retour-chariot <cr>.

Exemple:

```
A>PIP
*commande interne à PIP
*commande interne à PIP
*<cr>
A>
```

Dans le cas de la seconde syntaxe, PIP n'accepte qu'une seule commande sur la même ligne. La commande est immédiatement exécutée et le contrôle est redonné au système à la fin de son exécution.

Dans les deux cas la syntaxe de la commande interne à PIP est identique; sa forme générale est la suivante:

destination = source1,source2, ... ,sourcen

Le premier champ "destination" représente soit le fichier, soit le périphérique qui doit recevoir les données. Les champs du type "source" représentent un ou plusieurs fichiers ou périphériques à partir desquels seront copiées les données.

Si plusieurs fichiers sources sont donnés, ceux-ci doivent être du type ASCII et se terminer par le caractère de fin de fichier <ctrl+Z>. Une ligne de commande ne doit pas dépasser la taille de 255 caractères.

Les arguments "destination" et "source" peuvent être des références ambiguës, et être précédés du nom du disque comme cela a été dit auparavant.

Copie d'un ou plusieurs fichiers

La syntaxe de copie de fichier est un cas particulier de la syntaxe globale donnée plus haut. La commande ne nécessite en effet qu'un seul champ "source":

destination = source

Seul le champ "source" peut être ambigu. De plus, PIP permet des commandes abréviées pour transférer des fichiers entre disques. Dans ce cas il ne subsiste que le nom du disque dans l'un des deux champs, le fichier "destination" prend alors le même nom que celui du fichier "source":

PIP d:=source	copie sur le disque "d:" le ou les fichiers du disque courant, satisfaisant la référence "source".
PIP d:=s:source	copie sur le disque "d:" le ou les fichiers du disque "s:", qui satisfont la référence "source".
PIP destin=s:	copie le fichier "destin" du disque "s:" vers le disque courant.
PIP d:destin=s:	copie le fichier "destin" du disque "s:" vers le disque "d:".

Exemples de commandes PIP correctes :

A>PIP COPIE=ORIGINAL

A>PIP B:NEW.DOC=A:OLD.TEX

B>A:PIP B:=A:MBASIC.COM

A>PIP B:=*.BAS

Copie de disque à disque

La commande PIP permet de copier tous les fichiers d'un disque sur un autre disque, mais ne permet pas de copier le système CP/M lui-même (voir la commande de génération du système SYSGEN). La copie d'un disque entier est souvent bien utile pour archiver ou sauvegarder les programmes ou données, dans la mesure où les supports disques sont fragiles. En particulier les disquettes ont une durée de vie limitée, car les têtes sont en contact physique avec le support.

A>B:=A:*.*	copie de tous les fichiers de A vers B.

Il est fréquent aussi de trouver avec les produits standard de CP/M des utilitaires de copie totale de disquette en format simple ou double densité :

COPY.COM	copie de disquettes en simple densité
DCOPY.COM	copie de disquettes en double densité

Concaténation de fichiers

La syntaxe de concaténation répond à la syntaxe générale de PIP. Les fichiers "sources" du type ASCII sont lus dans l'ordre de leur définition et rassemblés dans le fichier destinataire unique.

Exemple :

```
A>PIP TOTAL.ASM=PART1.ASM, PART2.ASM, B:PART3.ASM
```

Conversion de supports

PIP permet en outre de transférer des données de fichier à périphérique et vis-versa. Les noms des périphériques logiques et physiques sont les mêmes qui ont été définis avec la commande STAT. PIP reconnaît aussi un certain nombre de noms de périphériques additionnels tels que :

NUL : émet 40 caractères nuls (00H) sur l'organe de sortie (amorce de ruban perforé par exemple).

EOF : marque de fin de fichier (1AH ou <ctrl-Z> sur l'organe destinataire.

INP : entrée spéciale pour un périphérique non standard. PIP fait un "CALL" à l'adresse 103H où l'utilisateur a introduit la routine de lecture en modifiant PIP par DDT. Le caractère lu est retourné à l'adresse 109H.

OUT : sortie spéciale pour un périphérique non standard. PIP fait un "CALL" à l'adresse 106H où l'utilisateur a introduit sa routine d'écriture. Le caractère à émettre doit se trouver dans le registre C.

PRN : nom logique pour une sortie de type imprimante avec prise en compte des tabulations, des sauts de page, de la numérotation des lignes.

Les commandes de PIP acceptent donc des noms de fichiers ou d'unités logiques de périphériques. Dans tous les cas, les données "source" sont lues jusqu'à la rencontre d'une fin de fichier, <ctrl-Z> pour les fichiers ASCII, et une fin réelle de fichier pour les fichiers non-

ASCII sur disque (plus de secteur). La fin de fichier <ctrl-Z> est rajoutée à la fin du fichier destinataire une fois le transfert terminé. Si le fichier destinataire existe déjà, celui-ci est détruit puis reconstruit à partir du fichier source spécifié.

La phase de copie peut être abandonnée à tout moment en frappant sur une touche quelconque du clavier. PIP émet alors le message: "ABORTED" pour indiquer que l'opération de transfert est terminée. De même, s'il se produit une erreur pendant le traitement, PIP abandonne toutes les commandes en attentes (s'il y en a) et redonne le contrôle au système.

Exemples de commandes de conversions entre supports:

A>PIP LST: =LISTING.PRN	copie le fichier LISTING.PRN sur le périphérique assigné à LST:.
A>PIP B: SAISIE.RUB=PTR:	copie les données lues sur le lecteur vers le fichier B:SAISIE.RUB.

Paramètres optionnels

Dans chaque commande l'utilisateur peut spécifier les paramètres supplémentaires. Ces paramètres doivent être entre crochets "[ ]", à la suite du nom de fichier ou de périphérique "source". Certains d'entre eux sont suivis d'une valeur décimale optionnelle.

B : Transfert en mode bloc par bloc: les données sont "bufferisées" jusqu'à la réception du caractère XOFF <ctrl-S> émis par le "source" pour délimiter la taille des blocs. PIP stoppe la réception et vide les données enregistrées en mémoire sur le fichier destinataire, puis se remet en lecture sur l'organe d'entrée. Ce paramètre est utilisé pour transférer sur disque des données d'un support séquentiel (cassette par exemple).

Dn : Détruit les caractères qui dépassent la colonne "n". Cette troncation de ligne est utilisée pour copier des fichiers sur les supports papiers ayant un nombre limité de caractères par ligne.

E : Mode "écho" sur la console de toutes les données transmises.

F : Annule les caractères "form-feed ou 0CH" (sauts de page).

H : Transfert de fichier du type HEX (format Intel), avec contrôle de parité longitudinale sur une ligne ("checksum").

I : Ignore les enregistrements débutant par ":00" dans un fichier au format Intel ou type HEX.

L : Convertit les lettres majuscules en minuscules.

N : Rajoute un numéro de ligne à chaque ligne transférée, en partant de la ligne 1, par pas de 1.

O : Transfert d'un fichier objet (non-ASCII).

Pn : Insère des sauts de page toutes les n lignes. Si n=1 ou est absent, le saut s'effectuera toutes les 60 lignes.

Qs↑Z : Arrête la copie dès que la chaîne <s> suivie de <ctrl-Z> est rencontrée.

Ss↑Z : Démarre la copie lorsque la chaîne <s> suivie de <ctrl-Z> est rencontrée.

Tn : Étend les tabulations à chaque "nième" colonne durant le transfert.

U : Convertit les lettres minuscules en majuscules.

V : Vérifie que les données ont bien été copiées en opérant une lecture comparative après l'écriture.

Z : Force de bit de parité à zéro pour chaque caractère ASCII transféré.

Améliorations sous la version CP/M 2.2

Gn : Lecture d'un fichier appartenant à l'utilisateur "n".

W : Forçage de l'écriture sur les fichiers R/O, protégés en écriture.

R : Lecture de fichier système (indicateur SYS positionné).

1.5.2.3. ED

Généralités sur le fonctionnement de ED

L'éditeur de texte ED est un éditeur orienté ligne par opposition à ceux orientés page ou écran. ED, qui travaille dans le contexte CP/M, permet de construire et modifier des fichiers de type texte codés en ASCII.

Ces fichiers sources sont organisés comme une suite de caractères ASCII, séparés par des caractères de fin d'article (séquence de <CR>, <LF> : carriage-return, line-feed). La taille d'un article n'est pas limitée et peut être quelconque ; elle est égale au nombre de caractères tapés avant le retour-chariot <CR>.

L'éditeur ED dispose d'un ensemble complet de commandes définies par une seule lettre, pour créer et corriger les fichiers texte.

Bien que la taille mémoire reste encore insuffisante sur les petits systèmes gérés par CP/M, ED a la possibilité de traiter des fichiers de taille quelconque dans la limite autorisée par le matériel. Si le fichier à éditer est vraiment trop volumineux pour entrer en mémoire, l'utilisateur peut appeler séparément des parties du fichier qui peuvent être introduites dans la zone mémoire réservée à l'édition.

L'appel à l'éditeur se fait de la façon suivante :

```
A>ED <nom du fichier>
```

Le nom du fichier donné en argument est un nom explicite de la forme :

```
d:fichier.typ
```

Si le fichier spécifié n'existe pas, il est créé et ouvert vide en écriture. Le message NEW FILE est alors affiché, suivi du caractère "*" "prompt" de ED, invitant l'utilisateur à taper une commande de l'éditeur. En principe cette commande doit être "I" (Insert) qui fait passer l'éditeur en mode d'insertion dans lequel l'utilisateur peut introduire son texte. Le passage du mode insertion au mode commande de ED se fait en tapant sur la touche "Control-Z" du clavier. Le caractère d'invitation "*" est alors renvoyé et ED attend une nouvelle commande. La sortie de l'éditeur se fait par la commande "E" (End). Le fichier source est dupliqué dans un autre fichier du type BAK. Ce fichier cons-

titue une sauvegarde du dernier source avant édition. L'utilité du "Backup" est indispensable pour éviter toute fausse manipulation ; la version précédente du fichier est toujours conservée dans ce fichier.

Exemple :

```
A>ED A:ESSAI.TEX

NEW FILE
   : *I
  1: CECI EST LA PREMIERE LIGNE DU FICHIER
  2: ON EST PASSE A LA SECONDE LIGNE EN TAPANT <cr>
  3: A LA LIGNE SUIVANTE ON VA TERMINER L'INSERTION
  4: <ctrl-Z>
   : *E        (sortie de ED)
A>
```

Dans le cas de figure où l'utilisateur sollicite l'éditeur, alors que le fichier existe déjà, et c'est le cas le plus fréquent, le fichier spécifié est ouvert. L'utilisateur peut alors amener une partie ou la totalité du fichier en mémoire par la commande "A" (append), pour le lister ou le modifier à sa guise. En fin de traitement, la zone mémoire appelée "buffer mémoire" est vidée sur un pseudo-fichier temporaire de type "**$$$**", suivi par le reste du fichier source non lu. Le nom du fichier source est alors renommé avec le type BAK et le fichier temporaire "fichier.**$$$**" est à son tour renommé avec son nom originel "fichier.typ".

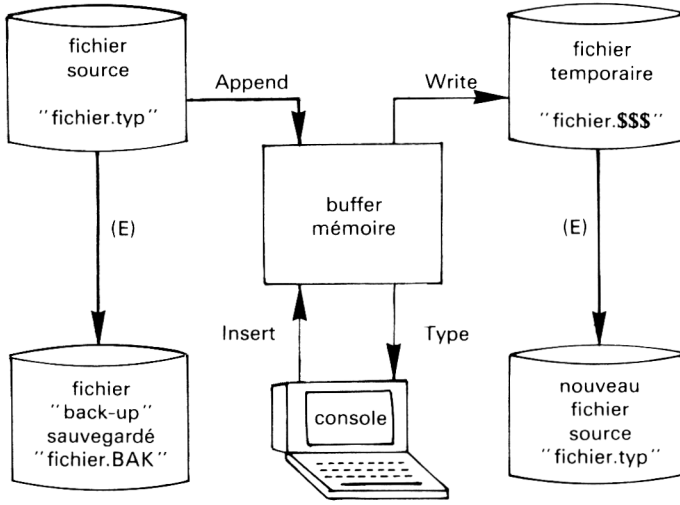


Fig. 8 — *Fichiers manipulés par ED*

Organisation du buffer mémoire

Le buffer mémoire est une zone réservée dans laquelle le texte source est rangé, après y avoir été introduit pas les commandes "A" (Append) ou "I" (Insert). Ce buffer est associé au pointeur du caractère courant "CP" appelé aussi "curseur". A chaque introduction d'une nouvelle ligne, le buffer mémoire se dilate de la taille de la ligne, de même à chaque suppression de ligne ou de caractère, le buffer se retasse. Ce buffer constitue donc un flot continu de caractères.

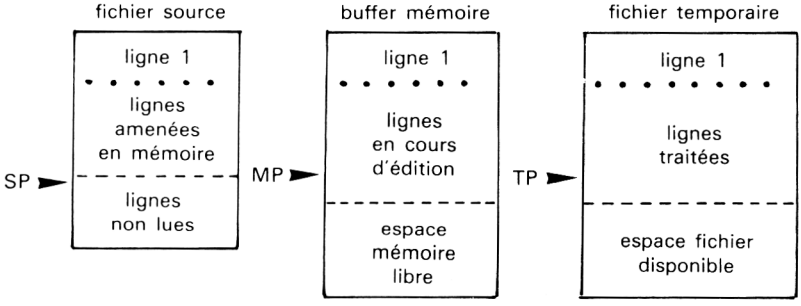


Fig. 9 — *Aspect du buffer mémoire par rapports aux fichiers*

Les trois pointeurs SP (pointeur du source, "source pointer"), MP (pointeur mémoire, "memory pointer"), TP (pointeur fichier temporaire, "temporary pointer"), sont mis à jour à chaque transfert du texte (commandes: A, W, E, H, O, R).

Le pointeur CP (pointeur du caractère courant, "character pointer"), est, quant à lui, mis à jour à chaque opération sur le texte dans le buffer mémoire (listage, modification, positionnement, etc...). CP, qui évolue entre le début et la fin du buffer (MP), est associé à la ligne courante CL (current line) du texte.

Opérations de transfert du texte

nA (append) Amène les "n" premières lignes non traitées du fichier source (localisées par SP), à la suite du buffer mémoire (localisé par MP). Les pointeurs SP et MP sont incrémentés de "n".

nW (write) Écrit les "n" premières lignes du buffer mémoire dans le fichier temporaire à partir de la localisation TP. L'espace correspondant en mémoire est libéré, et les lignes suivantes sont translatées vers le début du buffer, ce qui provoque un retassement en mémoire. Le pointeur MP est décrémenté de "n", tandis que le pointeur TP est incrémenté de "n".

+/—U (upper) Convertit les minuscules en majuscules si "+U", sinon n'opère aucune conversion.

La valeur de "n" peut varier de 1 (valeur par défaut qui peut être omise) à 65535, aussi symbolisée par le caractère "#". Ainsi la commande "#A" est couramment utilisée pour amener l'ensemble du fichier source en mémoire. Si le fichier source est trop volumineux, l'utilisateur peut appeler son fichier morceau par morceau en utilisant la commande "0A" qui remplit à moitié le buffer mémoire.

Opérations de transfert de fichier

E (end) Sortie normale de l'éditeur. Le buffer mémoire est recopié dans le fichier temporaire, ainsi que le reste des lignes non lues du fichier source. L'ancien fichier source est renommé avec le type BAK, et le fichier temporaire **$$$** est renommé du nom du fichier source. Le contrôle est ensuite redonné au CCP.

H (head) Identique à la commande E avec attente d'une nouvelle commande de l'éditeur sur le même fichier source.

O (original) Abandon de l'édition en cours et rappel du fichier source original non encore modifié. Tous les pointeurs sont remis à zéro.

Q (quit) Abandon du travail sans altérer aucun fichier et retour au moniteur.

Opérations sur le buffer mémoire

Toute opération sur le buffer mémoire met en jeu le pointeur de caractère courant CP, ainsi que la ligne courante CL.

I (insert) Insère une ou plusieurs lignes devant la ligne courante CL. Chaque ligne tapée doit se terminer par un retour-chariot <CR>; le caractère <LF> "line-feed" est

rajouté automatiquement. L'arrêt de l'insertion se matérialise par la frappe du caractère <ctrl-Z>, et l'éditeur renvoie alors le prompt "*".

+B (begin) Positionne le curseur CP au début du buffer mémoire.

−B (bottom) Positionne le curseur CP à la fin du buffer mémoire.

+/−nC (cursor) Déplace le curseur CP de "n" caractères vers l'avant si "+", ou "n" caractères vers l'arrière si "−". Les caractères de fin de ligne <CR> et <LF> sont comptabilisés comme les autres caractères.

+/−nD (delete) Supprime les "n" caractères après le pointeur CP si "+", sinon supprime les "n" caractères qui précèdent CL si "−".

+/−nK (kill) Supprime les "n" lignes, suivantes si "+" ou précédentes si "−", de texte source relatives au pointeur CP.

+/−nL (line) Positionne le curseur CP au début de la "nième" ligne relative à la ligne courante CL. Si n=0 le positionnement s'effectue sur la ligne courante. Pour une valeur de "n" en dehors des limites du buffer mémoire, le positionnement a lieu au début ou à la fin du buffer.

+/−nT (type) Affiche le contenu des (n−1) lignes relatives au positionnement du curseur CP. Si n=0 ce sont les caractères du début de la ligne courante jusqu'au curseur CP qui sont imprimés. Si n=1 (valeur par défaut) ce sont les caractères situés entre le curseur CP et la fin de la ligne courante qui sont imprimés.

+/−n Est équivalent à la commande "+/−nLT". Si n est omis, c'est la ligne suivante qui est listée (signe +), sinon c'est la ligne précédente (signe −).

+/−nP (page) Imprime "n" séquences de 24 lignes (page d'écran) par rapport à la ligne courante. P seul imprime la page suivante, −P imprime la page précédente et 0P imprime la page courante.

Recherche et modification de chaînes

nF<chaîne> (find)	Recherche de la chaîne de caractère <chaîne> dans le buffer mémoire à partir du positionnement courant du curseur CP. Si "n" est spécifié le positionnement s'effectuera sur la "nième" chaîne rencontrée. Si la recherche est infructueuse le CP restera à son emplacement initial, et le message "BREAK at #" sera renvoyé sur le terminal.
I<chaîne> (insert)	Insère la chaîne de caractères spécifiée à la suite du caractère pointé par CP.
nN<chaîne> (nième)	Recherche la "nième" occurrence de la chaîne spécifiée non pas dans le buffer mémoire, mais directement dans le fichier source. Dans le cas ou le buffer mémoire se sature lors de la recherche, une recopie automatique de celui-ci s'effectue dans le fichier temporaire (commande # W). Dès que la chaîne a été trouvée on remplit à moitié le buffer mémoire par les lignes suivantes du fichier source.
nS<chaîne 1> ↑ Z <chaîne 2> (substitute)	Opère une substitution de la nième chaîne <chaîne 1> par la chaîne <chaîne 2>
nJ<chaîne 1> ↑Z <chaîne 2> ↑Z <chaîne 3> (juxtaposition)	Juxtapose la chaîne <chaîne 2> à la chaîne <chaîne 1> et détruit tous les caractères jusqu'à rencontrer <chaîne 3>. Si <chaîne 3> est omis, aucune suppression n'est faite, c'est le cas d'une juxtaposition simple.

Opérations annexes

Un nombre quelconque de commandes peuvent être tapées sur la même ligne. Elles ne seront exécutées que lorsque le retour-chariot sera

rencontré. Si l'utilisateur fait une faute de frappe il a la possibilité d'utiliser les caractères spéciaux d'édition au clavier :

<rubout> : efface le dernier caractère tapé ;
<ctrl-U> : efface toute la ligne en cours de frappe ;
<ctrl-E> : efface toute la ligne et renvoie un retour-chariot ;
<ctrl-C> : ré-initialise le système CP/M (identique à "Quit").

Les commandes qui peuvent être combinées ensemble sont celles qui opèrent sur le buffer mémoire, (commandes : B, C, D, K, L, T, P).

Exemple de chaîne de commandes :

:*B3L-T position sur la ligne 3 (3L) par rapport au début (B) et impression de la ligne (–T).

Une "macro-commande" est aussi disponible pour exécuter de manière répétitive une séquence de commandes de ED ; il s'agit de la commande "M" dont la syntaxe est :

nM<chaîne de commandes>

La chaîne de commandes est exécutée "n" fois si n>1, sinon elle est exécutée de manière répétitive jusqu'à ce qu'une condition anormale se présente (fin de buffer par exemple).

Messages d'erreur

En phase d'édition (commandes tapées au clavier) l'éditeur de texte ED imprime le dernier caractère lu suivi d'un code d'erreur symbolisé par l'un des caractères suivants :

? commande inexistante ;
> buffer mémoire saturé (pour les commandes : D, K, N, S) ou chaînes trop longues (pour les commandes : F, N, S) ;
ne satisfait pas toutes les occurrences liées à la commande ;
0 fichier bibliothèque de type LIB non trouvé.

D'autres erreurs, non liées directement à l'édition de texte, peuvent survenir. En particulier, si une erreur physique sur disque est détectée, le

message suivant est imprimé :

PERM ERR DISK d

où "d" est le nom du disque courant sélectionné. L'utilisateur a alors le choix d'ignorer l'erreur en tapant n'importe quel caractère au clavier, ou de réinitialiser le système par un ↑C.

Amélioration sous la version CP/M 2.2

L'accès aux numéros de lignes de manière relative, par rapport à un autre numéro de ligne, est peu souple et fastidieux. La version CP/M 2.2 offre la possibilité d'accéder aux numéros absolus des lignes. Le numéro de ligne doit alors être suivi du caractère ":".

:*10:T	liste la ligne numéro 10.

De même l'utilisateur peut faire une référence absolue à une ligne, à partir du positionnement courant du curseur ; dans ce cas le numéro absolu est précédé du caractère ":".

:*:30T	liste les lignes de la ligne courante jusqu'à la ligne 30.
:*15::20K	supprime les lignes 15 à 20.

1.5.2.4. ASM

L'assembleur ASM traduit un fichier source écrit dans le langage symbolique du 8080, et produit un fichier objet au format Intel 8080 en hexadécimal. L'appel à l'assembleur ASM accepte deux types de syntaxe :

ASM <nom du fichier>

ou

ASM <nom du fichier>.paramètres

Le nom du fichier est implicitement du type ASM et ne doit pas être suivi de son type ASM. Les paramètres optionnels définissent les localisations (numéros des disques) des fichiers concernés par l'assemblage. Ces fichiers sont au nombre de trois :

le fichier source	"fichier.ASM"
le fichier objet	"fichier.HEX"
le fichier listing	"fichier.PRN"

Ainsi dans la forme complète de la syntaxe d'appel :

ASM <nom du fichier>.<s><h><p>

les paramètres <s>, <h>, <p> représentent successivement les noms des disques (sans le :), du fichier source pour <s>, du fichier objet pour <h>, et du fichier image imprimante <p>. Ces noms peuvent être, soit les symboles de A à P (du disque 0 au disque 15), soit les symboles X pour <p> (listing sur la console) ou Z pour <h> et <p> (inhibition du fichier de sortie).

Exemple :

A>ASM PROGR.BAB

va chercher le fichier source PROGR.ASM sur le disque B:, produit un fichier objet PROGR.HEX, résultat de l'assemblage, sur le disque A:, et génère un fichier image imprimante PROGR.PRN sur le disque B:.

De même :

A>ASM PROGRAMM.AZX

assemble le fichier PROGRAMM.ASM sur le disque A: en produisant un listing résultat sur le terminal, sans construire le fichier objet.

Format d'un programme assembleur

Comme dans la plupart des langages d'assemblage, chaque ligne symbolique peut se décomposer en quatre parties appelées "zones".

étiquette opération opérande ; commentaire

Toute ligne en langage assembleur doit se terminer par la séquence <CR><LF>. Le caractère "!" joue aussi le rôle d'un séparateur de ligne assembleur, et permet de mettre plusieurs instructions sur la même ligne physique.

La zone étiquette n'est pas obligatoire ; elle sert de référence symbolique pour les variables ou repère une destination de branchement dans le programme (aiguillages, boucles, etc...). Si elle est présente, elle commence en colonne 1, et est constituée d'une chaîne de 1 à 16 caractères alphanumériques, ne commençant pas par un chiffre. Pour des raisons de compatibilité avec d'autres assembleurs 8080, la zone étiquette peut se terminer par le caractère ":".

étiquettes correctes	étiquettes incorrectes
VARIABLE ETIQ18: boucle: LABEL$1982 symboletreslong:	"VARIABLE" 18ETIQ: (boucle): LABEL $ 1982 etiquettetroplongue:

La zone opération contient soit une directive d'assemblage, soit le mnémonique d'une instruction en langage machine 8080. La liste des codes instruction est fournie en annexe.

La zone opérande est réservée à la partie variable de l'instruction. Elle permet de définir une adresse, une valeur numérique, une référence symbolique (symbole), ou encore une expression calculable. Certaines instructions simples n'utilisent pas cette zone.

La zone commentaire est facultative et est ignorée par l'assembleur. Elle débute par le caractère ";" et se termine à la fin de la ligne physique. Des commentaires peuvent être introduits à n'importe quel emplacement dans le texte source. Par souci de compatibilité le caractère "*", s'il est placé en première colonne, indique que toute la ligne est en commentaire.

Mots réservés

Certains symboles ne peuvent pas être redéfinis par le programmeur, car ils sont utilisés pour référencer les registres du microprocesseur 8080 ou Z 80.

A	registre A
B	registre B
C	registre C

D	registre D
E	registre E
H	registre H
L	registre L
M	contenu de la paire de registre HL
SP	registre pointeur de pile
PSW	registre mot d'état programme (program status word).

Opérateurs arithmétiques et logiques

Les opérandes peuvent être des expressions combinant plusieurs autres opérandes par des opérations arithmétiques ou logiques. Les opérateurs reconnus sont :

+	somme arithmétique
–	différence arithmétique, ou moins unaire
*	multiplication entière non signée
/	division entière non signée
MOD	reste de la division
NOT	inverse logique ou complémentation à un
AND	intersection logique
OR	réunion logique
XOR	OU exclusif
SHL n	décalage vers la gauche de "n" bits avec remplissage de 0
SHR n	décalage vers la droite de "n" bits avec remplissage de 0.

Dans le cas où il y a plusieurs opérateurs le parenthésage est conseillé. En effet l'évaluation d'une expression procède de la gauche vers la droite et applique à certains opérateurs une priorité appelée précédence d'opérateur. Les opérateurs d'égale précédence sont évalués de la gauche vers la droite lorsqu'ils sont rencontrés dans une expression. L'ordre hiérarchique des précédences est défini comme suit :

```
* / MOD SHL SHR (plus forte priorité)
– +
NOT
AND
OR XOR (plus faible priorité)
```

Directives d'assemblage

Les directives d'assemblage se présentent sous la forme de mnémoniques dans la zone opération, mais ne sont pas exécutables à l'inverse des instructions. Les directives reconnues par l'assembleur ASM sont détaillées ci-dessous.

ORG

Définit l'adresse d'implantation de l'ensemble ou d'une partie du programme qui suit cette directive. La syntaxe générale de la directive ORG est :

étiquette ORG expression ; commentaire

Seul le champ expression est obligatoire. Il peut y avoir plusieurs directives ORG au sein d'un même programme. Un programme sous CP/M débute généralement par ORG 100H.

END

Indique à l'assembleur la fin du programme à assembler.

étiquette END expression ; commentaire

Si l'expression est absente, l'adresse de début d'exécution prise par défaut est 0000H, sinon l'expression après être évaluée définit l'adresse de départ du programme. Pour un programme CP/M standard on aura : END 100H.

EQU

Permet d'établir une équivalence ou un synonyme entre l'expression située à droite et le symbole en zone étiquette. La syntaxe est :

étiquette EQU expression ; commentaire

L'utilisation de la directive EQU permet au programmeur de mieux clarifier son programme, de lui donner plus de lisibilité par le choix de symboles personnalisés, et aussi de le rendre facilement paramétrisable.

SET

Cette directive est identique à la directive EQU, excepté que le symbole situé en zone étiquette, peut

être redéfini à plusieurs reprises au cours du programme.

IF,ENDIF

Ces deux directives permettent de faire de l'assemblage conditionnel. Leur syntaxe est la forme:

```
IF      expression
        ...
séquence d'instructions
        ...
ENDIF
```

L'expression qui suit le IF est évaluée et si sa valeur est différente de zéro (condition vraie) la séquence d'instructions qui suit est assemblée; si la valeur vaut zéro (condition fausse) l'assembleur saute la séquence entre le IF et le ENDIF et reprend l'assemblage après le ENDIF.

DB, DW, DS

Ces directives permettent d'initialiser et de réserver des zones mémoires.

DB : Initialise une zone mémoire octet par octet (8 bits).
DW: Initialise une zone mémoire mot par mot (16 bits).
DS : Réserve une zone mémoire de taille spécifiée en octets.

Les syntaxes possibles sont:

```
étiquette   DB   expr1,expr2,expr3...   ;commentaire
étiquette   DW   expr1,expr2,expr3...   ;commentaire
étiquette   DS   expression             ;commentaire
```

Les expressions expr1, expr2, expr3 peuvent être des constantes numériques, des constantes de type chaîne de caractères (textes) ou encore des symboles. La directive DS n'admet qu'une seule expression qui représente la taille en octets de la zone à réserver. La directive DW génère la valeur des expressions,

cadrée à droite, sur des mots de 16 bits, l'octet de poids faible étant rangé le premier et l'octet de poids fort étant rangé le second.

Messages d'erreur

On distingue deux catégories de diagnostics d'erreurs : les erreurs dues à l'assembleur lui-même et les erreurs dues à l'assemblage du programme source. Dans la première catégorie on peut signaler les messages d'erreurs suivants :

NO SOURCE FILE PRESENT	Le nom du fichier spécifié à la suite de la commande ASM n'existe pas. Vérifier s'il est sur le bon disque ou s'il n'y a pas eu une erreur de frappe.
NO DIRECTORY SPACE	Le catalogue du disque est saturé. Il est conseillé alors de détruire des fichiers dont l'utilisation n'est pas primordiale (fichiers de type PRN ou BAK).
SOURCE FILE NAME ERROR	Le nom du fichier à assembler doit être explicite. Ne pas utiliser les symboles "?"et "*".
SOURCE FILE READ ERROR	Une erreur physique sur un secteur du disque a été détectée dans le fichier source.
OUTPUT FILE WRITE ERROR	Soit il n'y a plus d'espace disponible sur le disque, soit le disque est protégé. Agir en conséquence : changer ou déprotéger le disque.
CANNOT CLOSE FILE	Le disque est protégé en écriture.

Dans la seconde catégorie il s'agit d'erreurs de syntaxe au niveau du programme source. On peut trouver les types d'erreurs suivants :

D : La valeur de l'expression ne convient pas (trop longue).
E : L'expression est invalide, ou sa valeur est trop grande.
L : La zone étiquette est incorrecte.
N : Instruction non implémentée sur ASM (voir MAC-80).
O : Overflow : l'expression est trop compliquée pour être traitée.
P : La valeur de l'étiquette a changé en cours du programme.
R : Le registre spécifié n'est pas compatible avec les mnémoniques.
U : Le symbole en zone opérande n'est défini nulle part ailleurs.
V : L'expression en zone opérande est incorrecte.

1.5.2.5. LOAD

La commande LOAD produit un fichier exécutable de type COM à partir d'un fichier objet, résultat d'un assemblage de type HEX au format Intel 8080 (binaire codé hexadécimal). C'est le fichier du type HEX qui se place en argument de la commande LOAD, sans donner son type qui est implicite. La syntaxe est en effet très simple :

LOAD <nom du fichier>

LOAD édite alors une séquence de renseignements de la forme :

```
FIRST    ADDRESS 0100      0100= adresse début du programme
LAST     ADDRESS  aaaa       aaaa= adresse fin du programme
BYTES    READ     nnnn       nnnn= nombre d'octets lus
RECORDS  WRITTEN    rr         rr= nombre d'enregistrement lus
```

Si le fichier spécifié n'est pas du type HEX un message d'erreur est renvoyé à l'utilisateur. Dans le cas où l'opération s'est bien passée, le fichier exécutable de type COM et de même nom principal est créé et le fichier de type HEX est conservé.

Le fichier de type COM peut alors être directement exécuté dans le TPA, en tapant tout simplement son nom (sans y adjoindre son type qui est implicite). C'est le module CCP du CP/M qui contient le chargeur pour implanter en zone TPA les programmes de type COM.

1.5.2.6. DDT

Le programme DDT (Dynamic Debugging Tool) est un outil indispensable pour la mise au point et les tests de programmes évoluant dans un environnement CP/M. La commande d'appel au metteur au point peut prendre deux formes :

```
    DDT
ou
    DDT d:nom du fichier.type
```

où d: est le nom du disque supportant le fichier spécifié. Le type peut être quelconque s'il s'agit de modifier le fichier en question sinon c'est le type COM qui est le plus généralement utilisé en vue d'une exécution contrôlée par DDT. Si le type HEX est spécifié DDT convertit le format Intel en binaire comme le faisait la commande LOAD.

La première forme syntaxique est équivalente à la seconde si la séquence de commande suivante est donnée :

```
A>DDT                          appel de DDT
-Id:nom du fichier.type        ouverture du fichier en entrée
-R                             chargement du fichier en mémoire
```

un message de bienvenue : nnK DDT VER v.v
est envoyé à la console, où "nn" est la taille mémoire en Kilo-octets, et "v.v" le numéro de version. DDT envoie ensuite :

```
NEXT  PC
nnnn  pppp
```

où "nnnn" est la première adresse disponible et "pppp" l'adresse de début d'exécution du programme (compteur ordinal).

DDT émet un "prompt" qui lui est caractéristique : le caractère "–", invitant l'utilisateur à taper une commande propre à DDT. A tout moment l'utilisateur peut quitter DDT en tapant <ctrl-C> ou G0 (branchement à l'adresse 0000H). L'image mémoire de son programme peut alors être sauvergardée par la commande :

```
SAVE nn "nom du fichier.COM"
```

où "nn" représente le nombre de pages de 256 octets du programme. Le fichier ainsi sauvé pourra à nouveau être exécuté soit directement en

tapant le nom du fichier après le prompt A>, soit à nouveau par DDT pour une nouvelle mise au point.

Les commandes de DDT

Avant de taper une commande interne à DDT, l'opérateur sur la console doit attendre le caractère d'invitation "–" indiquant que DDT est prêt. Les commandes de DDT sont formulées par une seule lettre assortie ou non d'arguments (valeur numérique ou adresse).

Ces commandes permettent d'assembler des instructions 8080, de désassembler du binaire, de visualiser et de modifier des zones mémoires, de tracer ou de s'arrêter sur des instructions. L'ensemble de ces fonctions est détaillé ci-dessous.

A (Assemble) — Aa — a = adresse héxadécimale

Cette commande permet d'assembler "en ligne" des instructions 8080. La différence avec le produit ASM est qu'il n'y a pas de champ étiquette et donc pas de gestion de symboles. Toutes les références d'adresse sont donc représentées en absolu et en héxadécimal. Une ligne d'assemblage se termine par un <cr>, l'adresse suivante est alors affichée pour accepter une nouvelle ligne d'assembleur. L'assemblage prend fin lorsque le programmeur tape une ligne vide, <cr> seul.

Exemple:
```
–A100
100 JMP 31A
103 <cr>
```

D (Display)

D
Dd — d = adresse début
Dd,f — f = adresse fin

Dans le premier cas le contenu des 192 octets, à partir de l'adresse courante, est visualisé sur la console, soit douze lignes. Chaque ligne de seize octets en hexadécimal est précédée d'une adresse en héxadécimal, et est suivie de la traduction ASCII de ces

octets. Les octets non imprimables sont remplacés par un ".".

F (Fill)

Fd,f,v v = valeur de l'octet

La commande F garnit la mémoire entre les adresses début et fin avec la valeur hexadécimale "v".

Exemple : —F1800,1A7F ,FF

G (Go)

G	
Gd	d = adresse de départ
Gd,a	a = adresse d'arrêt
Gd,a,b	b = seconde adresse d'arrêt
G,a	a = adresse d'arrêt
G,a,b	a,b = adresses d'arrêt

Dans le premier cas l'exécution du programme débute à partir de l'adresse courante dans le compteur ordinal, par défaut 100H. Dans le second cas l'adresse de début d'exécution est celle qui est spécifiée. Le troisième cas demande qu'un arrêt sur adresse (breakpoint) soit effectué lorsque le compteur ordinal atteindra cette valeur. Le quatrième cas permet de spécifier deux points d'arrêt. Enfin dans les deux derniers cas l'adresse de début d'exécution est l'adresse courante, c'est-à-dire l'adresse où l'utilisateur s'était par exemple arrêté précédemment. Les points d'arrêt sont automatiquement effacés lorsque le programme y est passé.

Exemple : —G100,158

H (Hexa)

Ha,b a,b = valeurs hexadécimales

Cette commande permet de faire la somme et la différence des deux valeurs numériques "a" et "b" codées en hexadécimal.

Exemple :—H1580,724
1CA4 0E5C

I (Input)

Id:nom du fichier.type

Cette commande permet d'ouvrir un fichier en lecture et de placer son FCB en 5CH de la mémoire (FCB implicite).

Exemple: —IB:TEST.COM

L (List)

L
Ld — d = adresse début
Ld,f — f = adresse fin

Cette commande permet de désassembler le code machine 8080 en assembleur symbolique 8080. Si L n'a pas d'argument, douze instructions sont désassemblées à partir de l'adresse courante, ce qui représente une moitié d'écran vidéo; deux commandes L successives remplissent l'écran du terminal.
Une adresse début et une adresse fin peuvent aussi être précisées.

Exemple:
```
—L100,108
100 LXI D,128
103 DAD D
104 SHLD 0122
107 MVI A,0
108 MOV B,E
109
```

M (Move)

Md,f,a — a = adresse destination

Cette commande permet de transférer une zone mémoire comprise entre les adresses début et fin, vers une autre zone mémoire définie à l'adresse destination. L'adresse début est incrémentée de 1 à chaque transfert d'octet et si elle dépasse l'adresse finale le transfert s'arrête.

Exemple: —M1200,127F,0080

R (Read)

R
Rd d = déplacement

Cette commande permet de charger en mémoire le fichier décrit dans le FCB implicite en 005CH de la mémoire. Les fichiers à charger en mémoire sont du type COM ou HEX. Si l'argument "d" est spécifié cette valeur est rajoutée à chaque adresse dans le programme à charger. La commande R ne peut être effective que si la commande I a été mise en œuvre auparavant.

S (Set)

Sa a = adresse mémoire

Cette commande permet d'examiner le contenu de la mémoire à l'adresse spécifiée et éventuellement de la modifier. DDT renvoie l'adresse mémoire suivie de son contenu. Si l'utilisateur tape un <cr>, le contenu n'est pas altéré et l'adresse suivante ainsi que son contenu sont affichés. La modification d'un octet en mémoire se fait en tapant une valeur hexadécimale en face du contenu actuel. Cette opération se termine par la frappe du caractère "." qui permet de revenir au niveau des commandes de DDT.

Exemple: -S103
103 4D<cr>
104 34 1A<cr>
105 18 .<cr>

T (Trace)

T
Tn n = nombre de pas

Cette commande permet une trace sélective du programme en cours d'exécution. L'argument "n" détermine le nombre d'instructions à tracer; s'il est absent une seule instruction est tracée. Le format d'une instruction tracée est le suivant:

CfZfMfEflf A=aa B=bbcc D=ddee H=hhll S=ssss P=pppp instruction

ou "f" est la valeur (0 ou 1) du flag associé :

C flag Carry
Z flag Zero
M flag Minus
E flag Parité paire
I flag Carry Interdigit.

A registre accumulateur sur un octet de huit bits
B paire de registres : B (poids forts),C (poids faibles)
D paire de registres : D (poids forts),E (poids faibles)
H paire de registres : H (poids forts),L (poids faibles)
S registre pointeur du sommet de la pile (16 bits)
P compteur ordinal de l'instruction en cours (16 bits)
instruction : mnémonique de l'instruction en cours.

Dans le cas où la trace s'arrête, l'adresse de la prochaine instruction à exécuter est notifiée à la fin de la dernière ligne après l'instruction désassemblée sous la forme : *<adr>.

Une trace peut être interrompue à tout moment en frappant sur la touche <rubout> ; le contrôle est alors redonné à DDT.

U (Untrace)

identique à la commande T.

Cette commande est la même que la commande T excepté que les étapes intermédiaires du programme en cours d'exécution ne sont pas tracées. C'est uniquement la dernière ligne de trace qui apparaît sur la console. Cette commande est une variante de la combinaison de la commande G avec un point d'arrêt et de la commande X.

X (Examine)

X
Xr r = registre du CPU

Cette commande permet de visualiser le contenu de tous les registres du CPU, c'est le cas de la première

forme, le format de la visualisation étant le même que celui de la commande T.

Dans le cas de la seconde forme le nom du registre ou du flag est précisé, et son contenu est affiché en hexadécimal sur 1, 8 ou 16 bits selon qu'il s'agit d'un flag, de l'accumulateur A ou d'une paire de registres. Ce contenu peut être modifié en donnant une nouvelle valeur en face de l'ancien contenu.

Fonctionnement de DDT

Lorsqu'un programme utilisateur est contrôlé par le metteur au point DDT, la zone TPA est alors occupée par deux programmes : le programme utilisateur et le programme DDT. Au moment de l'appel à DDT, celui-ci se charge comme un programme standard en zone TPA à l'adresse 100H, puis "s'autotranslate" de telle sorte que la fin de DDT soit contiguë avec la base du BDOS. L'espace mémoire utilisateur est ainsi libéré pour introduire un éventuel programme à tester. L'adresse du point d'entrée des services du BDOS, en 0006H de la mémoire, est remplacée par l'adresse du début du noyau de DDT, qui représente alors l'adresse de fin logique de la mémoire utilisable par le programme.

DDT possède une structure de recouvrement qui permet à certains programmes volumineux de venir "écraser" la partie non vitale de DDT. En effet DDT est découpé en deux modules : le noyau indispensable à la mise au point, et le module assembleur-désassembleur qui, lui, peut être recouvert par la fin d'un programme dans le TPA. Les commandes A et L deviennent alors inutilisables, et les commandes de trace donnent le contenu en héxadécimal de l'instruction.

Les commandes qui agissent sur le compteur ordinal P (G, T, U), utilisent l'instruction d'interruption logiciel RST 7 et son emplacement dans la zone SPA de la mémoire.

1.5.2.7. SUBMIT

SUBMIT est un utilitaire qui permet d'exécuter une séquence de commandes CP/M. L'utilisateur constitue un fichier de commandes, à

l'aide de l'éditeur de texte, lui évitant ainsi de retaper chaque fois ses commandes séparément.

Des paramètres formels peuvent éventuellement être utilisés afin que l'utilisateur puisse introduire ses propres paramètres au niveau de l'exécution de la commande.

La syntaxe générale de la commande est :

SUBMIT <nom du fichier> <paramètres>

Le fichier référencé doit être du type SUB, mais n'apparaît pas dans le nom du fichier donné en argument de la commande. Le champ paramètre facultatif, peut être constitué d'un ou de plusieurs paramètres sur la même ligne. Chacun de ces paramètres peut référencer un nom de fichier ou une information nécessaire au fichier de commande SUB.

A ces paramètres spécifiés dans la commande tapée au clavier, sont associés, dans le fichier de commande, les symboles formels :

$1 $2 $3 ... $n

Chaque symbole formel numéroté de cette manière référence le paramètre correspondant à la position où il se trouve dans la liste ; le premier paramètre est associé à $1, le second à $2, etc...

Soit le fichier de commande JOB.SUB préalablement construit par l'éditeur de texte ED:

```
ASM $1
LOAD $1
ERA $1.HEX
PIP $2=$1.PRN
ERA $1.PRN
$1
```

La commande :

A>SUBMIT JOB PROGRAM CON:

aura pour effet d'assembler le fichier PROGRAM.ASM, de construire le fichier exécutable PROGRAM.COM, de détruire le fichier objet intermédiaire PROGRAM.HEX, de lister sur la console le fichier impri-

mante PROGRAM.PRN, de détruire ce fichier, et enfin de charger et d'exécuter le fichier PROGRAM.COM.

SUBMIT crée un fichier intermédiaire $$$.SUB dans lequel il opère toutes les substitutions des paramètres formels $i par les paramètres effectifs tapés au clavier. Chaque ligne de ce fichier représentant une commande, SUBMIT exécute alors séquentiellement ces commandes jusqu'à épuisement du fichier.

1.5.2.8. DUMP

DUMP permet de visualiser en code hexadécimal le contenu d'un fichier quelconque sur la console.

L'affichage du fichier se fait ligne par ligne. Chaque ligne contient une suite de seize octets traduits en hexadécimal, et est précédée par une adresse relative par rapport au début du fichier (codée hexadécimale) sur quatre octets.

Le déroulement de l'affichage peut être suspendu à tout moment, si celui-ci est trop rapide (sur un écran à 9600 bauds par exemple), en frappant sur la touche <ctrl-S> (XOFF), pour faire une sorte d'arrêt sur image. La reprise de l'affichage se déclenchant par la frappe d'un caractère quelconque du clavier. De même, l'interruption définitive de la visualisation se réalise par la frappe d'un caractère quelconque (si le fichier est trop long par exemple).

Exemple d'affichage avec DUMP :

```
A>DUMP PROG.COM
0000  42 49 4F 53 22 20 3D 20 24 02 C3 87 0C CD 68 21
0010  00 00 39 22 DE 01 31 00 02 2A 06 00 22 48 01 23
0020  23 23 22 4A 01 2A 01 00 23 5E 23 56 1B 1B 1B EB
0030  22 4C 01 11 03 01 0E 09 CD 05 0D 2A 48 01 CD A9
```

1.5.2.9. MOVCPM

Le programme MOVCPM permet de reconfigurer le système CP/M pour n'importe quelle taille mémoire. Deux paramètres optionnels peuvent être mentionnés pour indiquer la taille désirée, en Kilo-octets, du nouveau système et l'attitude à prendre à la fin du traitement.

Si le second paramètre (caractère "*") est omis, MOVCPM reconstruit une nouvelle image mémoire du CP/M à la taille désirée (premier paramètre), et donne le contrôle à ce nouveau CP/M sans le sauvegarder sur disque. C'est un moyen pas dangereux de tester une nouvelle configuration du CP/M.

Par contre si le second paramètre "*" est présent, la nouvelle image du CP/M représente une image disque du CP/M prête à être sauvegardée sur disque à l'aide des commandes SAVE ou SYSGEN. Le message suivant est alors imprimé:

READY FOR "SYSGEN" OR
"SAVE 34 CPMnn.COM" où "nn" représente la taille mémoire en Kilo-octets (premier argument de MOVCPM).

L'image mémoire située entre les adresses 0900H et 2300H, constitue l'image disque du CP/M: le BOOT en 900H, le CCP en 980H, le BDOS en 1180H et le BIOS en 1F80H. Cette image mémoire peut être sauvegardée par la commande SAVE dans le fichier "CPMnn.COM", pour être éventuellement modifiée par DDT, et engendrer un nouveau système CP/M par la commande SYSGEN.

Type de commandes valides:

MOVCPM	Construit une image du nouveau CP/M en mémoire pour une taille maximale et donne le contrôle à ce CP/M.
MOVCPM nn	Construit une image d'un CP/M configuré à "nn" K-octets en mémoire et lui donne le contrôle.
MOVCPM * *	Construit une image disque en mémoire d'un CP/M configuré pour une taille maximale, en vue d'une sauvegarde sur disque par SAVE ou SYSGEN.
MOVCPM nn *	Construit une image disque en mémoire d'un CP/M configuré pour une taille de "nn" K-octets, pour être sauvé sur disque par SAVE ou SYSGEN.

1.5.2.10. SYSGEN

La commande SYSGEN permet de générer un système CP/M sur un nouveau disque. En fait SYSGEN copie le CP/M sur un autre disque. Ce genre d'opération se fait généralement pour créer des disquettes CP/M "bootables" soit à partir du CP/M courant, soit à partir d'une image mémoire d'un CP/M reconfiguré par MOVCPM, soit encore à partir d'un système CP/M modifié en mémoire par DDT.

SYSGEN fabrique donc un système CP/M sur la disquette destinatrice, qui pourra par la suite être utilisée comme disquette système.

Cette commande dialogue avec l'utilisateur de manière interactive par des questions claires afin d'éviter toute fausse manipulation.

Exemple de la mise en œuvre de SYSGEN:

```
A>SYSGEN
SYSGEN VER x.x
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
SOURCE ON A:,THEN TYPE RETURN <cr>
FONCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B:,THEN TYPE RETURN <cr>
FONCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>
A>
```

1.6. SERVICES SYSTÈME OFFERTS

On peut considérer que CP/M offre deux catégories de services système: les fonctions du BDOS et les primitives du BIOS.

1. Les fonctions logiques offertes par le BDOS sont indépendantes de l'environnement matériel, et sont accessibles à n'importe quel programmeur. En effet, l'accès à ces fonctions se fait par un passage obligé, c'est ce qu'on appelle un "guichet", où des contrôles stricts sont effectués sur la validité et la cohérence des paramètres transmis.

2. Les primitives d'entrées-sorties du BIOS sont liées à l'environnement matériel: console, imprimante, disquette ou disque dur.

Ces primitives doivent être manipulées avec la plus grande précaution et leur utilisation est déconseillée pour les programmeurs non avertis.

Nous allons analyser dans la suite de ce chapitre l'ensemble de ces services offerts.

1.6.1. Les fonctions système du BDOS

1.6.1.1. Le point d'entrée des services système

L'accès aux fonctions du CP/M est obtenu en passant le numéro de la fonction et éventuellement une information supplémentaire à travers le point d'entrée situé à l'adresse 0005H de la mémoire, par l'intermédiaire de l'instruction d'appel CALL.

Le numéro de fonction est transmis dans le registre "C", et l'information supplémentaire dans la paire de registres "DE". L'information retournée après le "CALL" est soit située sur un octet dans le registre "A", soit délivrée sur un mot dans la paire de registres "HL".

Exemple d'appel d'un service moniteur :

```
BDOS  EQU  0005H          ; au début du programme
      ...
      LXI  D,<adresse>    ; argument dans la paire de registre DE
      MVI  C,<numéro>     ; numéro de la fonction dans le registre C
      CALL BDOS           ; appel du service moniteur
```

Nous allons analyser avec plus de détails l'ensemble des services offerts par le BDOS que l'on peut classifier en quatre groupes : opérations sur les organes asynchrones (console, imprimante,lecteur rapide, etc...), opérations sur les disques, opérations sur les fichiers, et opérations diverses.

Certaines fonctions, en particulier celles relatives aux notions de numéro d'utilisateur, d'attribut de fichier ou d'accès direct, ne sont disponibles qu'en CP/M 2.2.

1.6.1.2. Réinitialisation du système

Il existe deux méthodes pour terminer normalement un programme qui s'exécute dans la zone TPA. La première consiste à provoquer une réinitialisation du système par un branchement à l'adresse 0000H, ou par un appel au BDOS avec le code de fonction C=0. Ces deux possibilités sont strictement équivalentes :

```
BOOT  EQU   0000H   ou   BDOS  EQU   0005H
      ...                      ...
      ...                      MVI   C,0
      JMP   BOOT               CALL  BDOS
```

Une seconde méthode, plus rapide, et possible sur des programmes qui ne recouvrent pas le CCP, consiste à restituer la valeur du pointeur de pile du début du programme et d'exécuter une instruction RET.

```
OLDSP  DS    2        ; mémoire de sauvegarde de SP
DEB    LXI   H,0      ; valeur du pointeur de pile au début
       DAD   SP       ;
       SHLD  OLDSP    ; sauvegardée dans la variable OLDSP
       ...
EXIT   LHLD  OLDSP    ; restitution du pointeur de pile
       SPHL           ; dans le registre SP
       RET            ; retour au programme appelant CCP
```

1.6.1.3. Opérations sur la console et l'imprimante

— Lecture d'un caractère de la console (Console input) :

```
appel  :   C= 1
retour :   A= caractère en ASCII
```

Le caractère est renvoyé en écho sur la console, excepté <lf> s'il suit un <cr>. Les caractères de tabulation sont remplacés par des espaces pour des taquets de huit colonnes. S'il s'agit d'un caractère de contrôle excepté <cr> *et* <lf>, il est renvoyé en écho précédé du caractère "↑".

— Écriture d'un caractère sur la console (Console output) :

```
appel  :   C= 2
           E= caractère en ASCII
```

Les caractères de tabulation sont étendus en espaces pour des taquets de huit colonnes. Si le caractère <ctrl-S> a été tapé au cours d'une sortie sur console, cette sortie est suspendue jusqu'à ce qu'un autre caractère quelconque soit tapé au clavier.

— Lecture d'un caractère sur le lecteur (Reader input):

appel : C= 3
retour : A= caractère ASCII

La lecture s'effectue sur le périphérique assigné au lecteur RDR: (voir la commande STAT et l'octet IOBYTE).

— Écriture d'un caractère sur le perforateur (Punch output):

appel : C= 4
E= caractère ASCII

Le caractère ASCII dans le registre E est émis vers le périphérique assigné à PUN:.

— Écriture d'un caractère sur l'imprimante (List output):

appel : C= 5
E= caractère ASCII

Le caractère ASCII dans le registre E est envoyé sur le périphérique assigné à LST: (imprimante).

— Entrée-sortie directe sur la console (Direct console I/O):

appel : C= 6
E= OFFH lecture d'un caractère
E= OFEH état du port asynchrone
E= caractère ASCII à écrire
retour : A= caractère si lecture
A= status si état demandé

Les entrées-sorties directes sur la console sont utilisées pour des applications qui demandent des temps de réponse très courts. Les caractères de contrôle ne sont pas filtrés et leur gestion est laissée libre à l'utilisateur.

— Lecture de l'octet IOBYTE (Get I/O byte):

appel : C= 7
retour : A=octet IOBYTE

L'octet IOBYTE, qui définit les assignations des périphériques physiques aux périphériques logiques, est retourné dans l'accumulateur A.

— Modification de l'octet IOBYTE (Set I/O byte):

appel : C= 8
E= nouvel octet IOBYTE

— Impression d'un message sur la console (Print string):

appel : C= 9
DE= adresse de la chaîne de caractères

Les caractères sont imprimés à partir de l'adresse définie dans la paire de registres DE, jusqu'à ce qu'un caractère nul 00H ou un caractère "$" soit rencontré.

Exemple:

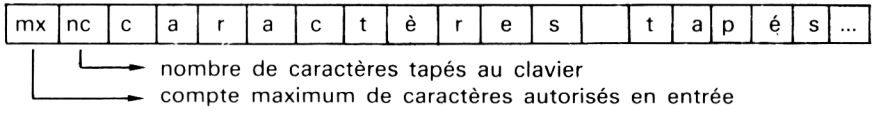
```
BDOS  EQU   5
MESS  DB    'A VOTRE SERVICE',0DH,0AH,'$'
      ...
      MVI   C,9
      LXI   D,MESS
      CALL  BDOS
```

— Lecture d'une ligne console complète (Read console buffer):

appel : C = 10 ou 0AH
DE = adresse du buffer de réception de l'utilisateur, le compte maximum de caractères autorisés en lecture doit être placé dans le premier octet du buffer.

retour : le nombre de caractères tapés est situé en buffer+1 les caractères tapés sont situés à partir de buffer+2

Format du buffer de lecture :



Cet appel permet de remplir directement un buffer mémoire à partir d'une ligne tapée au clavier se terminant par <cr>. L'écho des caractères introduits est envoyé sur la console. Les tabulations sont étendues et les caractères spéciaux sont pris en compte : effacement d'un caractère, d'une ligne, réimpression d'une ligne...

— Interrogation du "status" de la console (Get console status) :

appel : C= 11 ou 0BH
retour : A= 0 ou Z-flag=1 si absence de caractère
A= FF ou Z-flag=0 si présence d'un caractère

Cette fonction est très utile pour tester si un caractère est présent sur le port console.

1.6.1.4. Opérations sur les disques

— Réinitialisation du disque système (Reset disk system) :

appel : C= 13 ou 0DH

Cette fonction est utilisée pour remettre le disque système dans un état normal : protection R/W et adresse DMA en 0080H de la mémoire, après un changement de disque sans opération de redémarrage à froid.

— Sélection d'un disque (Select disk) :

appel : C= 14 ou 0EH
E= numéro du disque à sélectionner

Les numéros des disques 0, 1, ..., 15 correspondent aux noms des disques A, B, ..., P.

— Interrogation des disques disponibles (Return login vector):

appel : C= 24 ou 18H
retour : HL= vecteur de login

Le vecteur de "login" est un mot de 16 bits, dont chaque bit correspond à un disque (0 à 15). Le bit de plus faible poids dans le registre L est associé au disque A, et ainsi de suite jusqu'au bit de plus fort poids pour le disque P. Le bit indique que le disque n'est pas connecté (bit = 0) ou que le disque est présent (bit = 1).

— Numéro du disque courant (Return current disk):

appel : C= 25 ou 19H
retour : A= numéro du disque courant

Cette fonction retourne un numéro de 0 à 15, qui est celui du disque actuellement sélectionné.

— Protection d'un disque (Write protect disk):

appel : C= 28 ou 1CH

Cette fonction protège en écriture la totalité du disque courant. Sa déprotection sera effective au cours de la prochaine réinitialisation du système (démarrage à chaud ou à froid).

— Lecture du vecteur de protection (Get Read/Only vector):

appel : C= 29 ou 1DH
retour : HL= vecteur R/O

Cette fonction retourne un vecteur de 16 bits, où chaque bit est associé à un disque (A poids faible, et P poids fort). Un bit à 1 indique que le disque associé est temporairement protégé.

1.6.1.5. Opérations sur les fichiers

— Ouverture d'un fichier déjà existant (Open file):

appel : C= 15 ou 0FH
DE= adresse du FCB
retour : A= code directory (0, 1, 2, 3) si fichier trouvé
A= 0FFH si fichier non trouvé

Un balayage du directory est effectué en comparant les positions 1 à 14 du FCB avec celles de chaque entrée. Si le FCB comporte des "?" dans le champ symbolique du nom de fichier, le balayage s'arrête à la première coïncidence rencontrée. Le code retourné dans l'accumulateur A correspond à l'index de l'entrée dans le secteur courant du directory (0, 1, 2, 3). Avant de faire un "open" l'enregistrement courant "cr" dans le FCB doit être mis à zéro par le programmeur, pour que le fichier soit accessible séquentiellement à partir du premier enregistrement.

— Fermeture d'un fichier (Close file):

appel	:	C= 16 ou 10H
		DE= adresse du FCB
retour	:	A= code directory (0, 1, 2, 3) si le fichier existe
		A= 0FFH si le fichier n'existe pas

La fermeture d'un fichier en lecture n'est pas indispensable, par contre un fichier en création ou en mise à jour doit être fermé pour que les enregistrements écrits sur disque soient comptabilisés dans la "map" de l'entrée du fichier. Les codes retournés sont identiques à ceux de la fonction "open".

— Recherche du premier fichier ambigu (Search for first):

appel	:	C= 17 ou 11H
		DE= adresse du FCB
retour	:	A= code directory (0, 1, 2, 3) si coïncidence trouvée
		A= 0FFH si aucun fichier n'est trouvé

Les caractères "?" dans le nom du fichier du FCB expriment que le fichier est ambigu. Si une coïncidence est trouvée dans le balayage du catalogue, l'index de l'entrée correspondante est retourné dans le registre A, et le secteur courant du directory se trouve en mémoire à l'adresse 0080H.

— Recherche du fichier ambigu suivant (Search for next):

appel	:	C= 18 ou 12H
		DE= adresse du FCB
retour	:	identique à la fonction de recherche du premier fichier

— Destruction d'un fichier (Delete file):

appel : C= 19 ou 13H
DE= adresse du FCB
retour: A= code directory (0, 1, 2, 3) si le fichier existait
A= 0FFH si le fichier n'existe pas

Cet appel détruit toutes les entrées, si le fichier a plusieurs extensions, dans le directory du disque. Tous les blocs utilisés par le fichier sont rendus disponibles.

— Lecture séquentielle d'un enregistrement (Read sequential):

appel : C= 20 ou 14H
DE= adresse du FCB
retour : A= 0 si la lecture s'est bien passée
A= 1 si une erreur de lecture s'est produite

Les 128 octets de l'enregistrement courant du fichier sont lus dans le buffer disque DMA. Le compteur d'enregistrement "cr" progresse d'une unité à chaque transfert, et, s'il déborde, l'extension "ex" suivante du fichier est ouverte et la lecture reprend avec un compteur d'enregistrement remis à zéro.

— Écriture séquentielle d'un enregistrement (Write sequential):

appel : C= 21 ou 15H
DE= adresse du FCB
retour : A= 0 si l'écriture s'est bien passée
A non nul si erreur d'écriture ou saturation disque

Cette fonction écrit un enregistrement de 128 octets, mémorisé à partir de l'adresse DMA, à la suite de l'enregistrement précédent. Le compteur d'enregistrement "cr" est incrémenté à chaque écriture, et, s'il déborde, une extension "ex" est automatiquement ouverte dans le catalogue et le compteur "cr" repart à zéro.

— Création d'un fichier (Make file):

appel : C= 22 ou 16H
DE= adresse du FCB
retour : A= code directory (0, 1, 2, 3) si normal
A= 0FFH si le directory est saturé

Cette fonction est similaire à la fonction "open", à la seule différence que le fichier est à créer même s'il existe déjà. Une entrée est initialisée dans le catalogue directory, de telle sorte que le fichier soit vide.

— Changement de nom d'un fichier (Rename file):

appel : C= 23 ou 17H
DE= adresse d'un FCB spécial
retour : A= code directory (0, 1, 2, 3) si le fichier existe
A= 0FFH si le fichier n'existe pas

Cet appel change le nom et le type de l'ancien fichier, spécifiés dans les 16 premiers octets du FCB, par le nom et le type du nouveau fichier, spécifiés dans les 16 octets suivants.

— Initialisation de l'adresse buffer disque (Set DMA address):

appel : C= 26 ou 1AH
DE= adresse du buffer disque DMA

Même si la configuration physique ne supporte pas le mécanisme d'accès direct mémoire DMA (Direct Memory Access), le buffer DMA est une zone mémoire de 128 octets destinée à recevoir des données en provenance du disque, ou qui contient des données à expédier sur le disque. L'adresse DMA implicite en 0080H de la mémoire est disponible au programmeur, et est automatiquement forcée à chaque réinitialisation du système.

— Modification des attributs d'un fichier (Set file attributes):

appel : C= 30 ou 1EH
DE= adresse du FCB
retour : A= code directory

Les attributs de fichier (R/O, R/W) et (DIR, SYS) sont représentés par le bit de poids fort des octets respectifs t1' et t2' du type de fichier dans le FCB. Le type de fichier n'est pas modifié car les caractères ASCII sont traités sur les sept bits de poids faible.

— Lecture directe d'un enregistrement (Read random):

appel : C= 33 ou 21H
DE= adresse du FCB
retour : A= code

L'opération de lecture a lieu pour un numéro d'enregistrement spécifié dans les octets "r0", "r1", "r2" aux positions 33, 34, 35 du FCB. L'adresse de l'enregistrement sur disque est donc codée sur 24 bits, pour pouvoir atteindre la capacité maximale de 8 Mégaoctets. L'octet "r0" représente les poids faibles, et l'octet "r2" les poids forts.

— Écriture directe d'un enregistrement (Write random):

appel : C= 34 ou 22H
DE= adresse du FCB
retour : A= code d'erreur

L'écriture directe fonctionne selon les mêmes principes énoncés plus haut avec la lecture directe.

— Calcul de la taille d'un fichier direct (Compute file size):

appel : C= 35 ou 23H
DE= adresse du FCB
retour : le champ r0, r1, r2 du FCB est initialisé

La taille virtuelle d'un fichier construit en mode séquentiel est égale à sa taille physique. Si le fichier est construit en mode direct, des "trous" vont apparaître dans l'allocation de l'espace disque, et la taille utile ne sera pas la même que la taille demandée.

— Positionnement sur un enregistrement direct (Set random record):

appel : C= 36 ou 24H
DE= adresse du FCB
retour : le champ r0, r1, r2 du FCB est initialisé

Cette fonction permet de se positionner sur un enregistrement très précis d'un fichier construit en mode séquentiel.

1.6.1.6. Opérations diverses

— Numéro de version CP/M (Return version number):

appel : C= 12 ou 0CH
retour : HL= numéro de version

Le registre H indique s'il s'agit d'un MP/M (H = 01) ou d'un CP/M (H = 00). Le registre, s'il est nul, indique qu'il s'agit de versions antérieures à CP/M 2.0, sinon il contient les valeurs 20H, 21H, ..., 2FH pour les différentes versions de CP/M 2.0.

— Obtention des adresses d'allocation (Get addr alloc):

appel : C= 27 ou 1BH
retour : HL= adresse du vecteur d'allocation

Un vecteur d'allocation est conservé en mémoire principale pour chaque disque connecté. Il permet par exemple de connaître la taille restante de l'espace de stockage sur disque (commande STAT).

— Obtention de l'adresse des paramètres disques (Get addr Disk parms):

appel : C= 31 ou 1FH
retour : HL= adresse du DPB (bloc de paramètres du disque)

Cette fonction retourne l'adresse d'une table où ont été rangées les caractéristiques des disques de la configuration: simple ou double densité, nombre de faces, nombre de pistes, nombre de secteurs par piste, taille d'un secteur...

— Initialisation ou lecture du numéro utilisateur (Set/Get user code):

appel : C= 32 ou 20H
E= 0FFH obtention du numéro d'utilisateur
E= code utilisateur pour modification
retour : A= numéro d'utilisateur si obtention

Cette fonction permet de lire le numéro d'utilisateur courant ou de le modifier. Seize utilisateurs sont autorisés sous CP/M 2.2, auxquels sont attribués les numéros 0 à 15.

1.6.2. Les primitives système du BIOS

1.6.2.1. Les points d'entrées des primitives système

A l'opposé du BDOS où l'accès aux fonctions se fait par un passage obligé en 0005H de la mémoire, l'accès aux primitives d'entrées-

sorties du BIOS se fait par un tableau ou vecteur de branchement situé à la base du BIOS pour des raisons de commodité d'interface avec les autres modules (CCP et BDOS). La base du BIOS se déduit à partir du contenu de l'adresse 0001H de la mémoire qui pointe sur le point d'entrée du "warm start" en BIOS+3.

Ce vecteur de branchement comporte 17 instructions de branchement vers les sous-programmes spécifiques du BIOS. Ces derniers peuvent être classés en trois catégories :

— réinitialisations du système,
— entrées-sorties mode caractère,
— entrées-sorties disque.

Les principales fonctions traitées par le BIOS sont les suivantes :

BIOS	JMP boot	: réinitialisation après un "cold start"
BIOS+3	JMP wboot	: réinitialisation après un "warm start"
BIOS+6	JMP const	: test du "status" console
BIOS+9	JMP conin	: lecture d'un caractère à la console
BIOS+12	JMP conout	: écriture d'un caractère à la console
BIOS+15	JMP list	: écriture d'un caractère sur imprimante
BIOS+18	JMP punch	: écriture d'un caractère sur le "puncher"
BIOS+21	JMP reader	: lecture d'un caractère sur le lecteur
BIOS+24	JMP home	: positionnement sur la piste 00 du disque
BIOS+27	JMP seldsk	: sélection d'un disque "drive"
BIOS+30	JMP settrk	: initialise le numéro de piste
BIOS+33	JMP setsec	: initialise le numéro de secteur
BIOS+36	JMP setdma	: initialise l'adresse du buffer DMA
BIOS+39	JMP read	: lecture d'un secteur disque
BIOS+42	JMP write	: écriture d'un secteur disque
BIOS+45	JMP listst	: test du "status" imprimante
BIOS+48	JMP sectran	: translation de numéros de secteurs

1.6.2.2. Primitives de réinitialisation

BOOT — est sollicitée uniquement par le chargeur du système d'exploitation lors du démarrage à froid c'est-à-dire lors de l'opération de "bootstrap" après une "RAZ" manuelle du système. Certaines variables dans la zone SPA en bas de mémoire sont initialisées, et le contrôle est alors donné au module de dialogue CCP.

WBOOT — est sollicitée à chaque réinitialisation du système, qu'elle soit provoquée par la frappe d'un caractère <ctrl-C> ou programmée (JMP 0000H ou CALL 0005H avec C = 0). Son rôle est de ramener le système CP/M en mémoire et de redonner le contrôle au CCP.

1.6.2.3. Primitives d'entrées-sorties mode caractère

CONST — retourne le "status" du périphérique assigné à la console dans l'accumulateur A. Si A = 0 aucun caractère n'est présent, si A = 0FFH un caractère est présent.

CONIN — lit un caractère sur le périphérique console dans le registre A. Le bit de parité est supprimé.

CONOUT — émet le caractère présent dans le registre C vers la console.

LIST — émet le caractère présent dans le registre C vers le périphérique assigné à l'imprimante.

PUNCH — émet le caractère présent dans le registre C vers le périphérique assigné au perforateur rapide.

READER — lit un caractère sur le périphérique assigné au lecteur rapide dans le registre A.

1.6.2.4. Primitives d'entrées-sorties disque

HOME — ramène les têtes de lecture du disque sélectionné au début du disque, sur la piste 0.

SELDSK — sélectionne le disque dont le numéro est dans le registre C : 0 pour A, 1 pour B,15 pour P.

SETTRK — permet de se positionner sur le numéro de piste indiqué dans la paire de registres BC.

SETSEC — permet de sélectionner le numéro de secteur dans la piste, sur lequel se réalisera le transfert ; ce numéro est dans le registre C.

SETDMA — permet d'initialiser l'adresse DMA contenue dans la paire de registres BC, pour les transferts suivants.

READ — se positionne sur les numéros de piste et de secteur définis préalablement par SETTRK et SETSEC, et lit le secteur pour transférer son contenu à l'adresse initialisée par SETDMA. READ retourne un code de terminaison dans l'accumulateur A ; si A = 0 la lecture s'est bien passée, sinon, si A = 1, une erreur irrécupérable s'est produite.

WRITE — écrit les informations, 128 octets en mémoire, situées à l'adresse DMA, sur le secteur sélectionné.

SECTRAN — calcule le numéro du secteur physique à partir du numéro du secteur logique donné dans le registre C. En effet, pour des raisons de performances, sur certaines disquettes, les secteurs logiques ne sont pas contigus, afin d'éviter de perdre un tour de disque pour se positionner sur le secteur suivant. Un facteur de "vissage" égal à 6 est appliqué sur le numérotage des secteurs. Les premiers secteurs logiques (1, 2, 3, 4, ...) correspondent par exemple aux secteurs physiques (1, 7, 13, 19, ...). De manière générale il existe une table de translation dans le BIOS par type de disque, son adresse est donnée dans la paire DE, et le résultat rendu dans la paire HL.

2

Le système MP/M

2.1. GÉNÉRALITÉS SUR MP/M

Le système d'exploitation MP/M (Multi-Programming Monitor) est un système "multi-usagers", sur-ensemble de CP/M qui, lui, est mono-usager.

Le but de ce système est de supporter un accès multi-terminal avec possibilité de "multi-tasking" au niveau de chaque terminal.

MP/M peut gérer de 1 à 16 utilisateurs, chacun d'eux étant associé à un poste de travail : une console physique.

Sous MP/M apparaît également la notion de propriétaire, matérialisée par un numéro d'utilisateur de 0 à 15 (ne pas confondre la notion de poste de travail ou console avec celle de propriétaire ou numéro d'usager).

Tout utilisateur de 1 à 15 a accès en lecture et en exécution à tous les utilitaires et bibliothèques de sous-programmes stockés sous le numéro utilisateur commun 0 (numéro du système). Par contre, d'un utilisateur à un autre, les programmes sont indépendants, et deux utilisateurs différents peuvent créer des fichiers de même nom sans pour cela qu'il y ait un mélange. Chaque utilisateur reste propriétaire de ses informations et n'interfère pas sur celles de son voisin.

Les utilitaires généralement stockés sous le numéro utilisateur commun permettent aux usagers de partager les commandes et ainsi ne pas encombrer leur propre espace disque par les duplications abusives de programmes.

Le passage d'un utilisateur à un autre se fait par un mécanisme de partage du temps CPU appelé "Time-slicing". Une tranche de temps de 20 milli-secondes maximum est attribuée à chaque programme actif (processus). Pour améliorer l'enchevêtrement des processus, tout processus qui initialise une entrée-sortie ou sollicite une fonction moniteur, est désactivé au profit du processus le plus prioritaire en attente.

Tout se passe comme si chaque utilisateur travaillait seul avec la machine. Un inconvénient normal mais inévitable subsiste: plus le nombre d'usagers ou de processus s'accroît, plus le temps de réponse se dégrade (overhead-time).

2.2. ENVIRONNEMENT MATÉRIEL

Les configurations matérielles avec lesquelles MP/M peut fonctionner sont les suivantes:

— microprocesseurs : 8080, 8085 ou Z 80
— mémoire : de 32 à 400 k-octets, avec ou sans banc
— consoles : 1 à 16 consoles du type "CRT" (écran + clavier)
— disques : 1 à 16 disques souples ou durs
— base de temps : une horloge programmable sous interruption

2.3. DESCRIPTION FONCTIONNELLE DU MP/M

Le système d'exploitation MP/M est construit autour d'un noyau temps réel multi-tâches. Ce noyau logique est indépendant de l'environnement matériel, mais dépend du CPU car il est écrit en langage assembleur 8080.

Le système MP/M est divisé en trois parties, chacune d'elles représentant un niveau moniteur.

— XIOS (eXtended Input Output Supervisor)
— BDOS (Basic Disk Operating System)
— XDOS (eXtended Disk Operating System)

A cet ensemble de modules s'ajoute une zone réservée à l'interface système-utilisateur. Cette zone a une taille variable selon la configuration adoptée. Elle est située vers les adresses hautes de la mémoire, et se subdivise en trois parties :

— SYSTEM.DAT : paramètres de génération
— CONSOLE.DAT : tampons d'entrées-sorties consoles
— USERSYS.STK : pile de travail

MP/M permet également, si la taille du moniteur rentre dans les 16K, d'allouer le reste de la mémoire commune à des processus utilisateurs qui sont déclarés résidents lors de la génération système. Il s'agit des programmes de type "RSP" (processus système résidents).

Le reste de la mémoire disponible étant ce que nous avons déjà appelé dans CP/M, le "TPA".

2.3.1. Le XIOS

Le module XIOS n'est autre qu'une extension du module BIOS sous CP/M. Il contient tous les programmes spécifiques (drivers) liés à l'environnement matériel. XIOS constitue l'interface logiciel-matériel entre les autres modules du système et les coupleurs des périphériques.

Les principales primitives préprogrammées permettent :

— la gestion des ports série et parallèle ;
— la sélection de disque, de face, de piste et de secteur ;
— la sélection et la protection des bancs mémoires ;
— l'initialisation des transferts physiques sur disque ;
— le traitement des interruptions ;
— la gestion de la base de temps (compteur).

Ces sous-programmes peuvent être modifiés, changés ou supprimés à la guise de l'utilisateur, selon la configuration qu'il désire adopter (le "source" est fourni).

2.3.2. Le BDOS

Le module "BDOS" contient les fonctions logiques de gestion des fichiers disque et des consoles conversationnelles (gestion de la procédure asynchrone).

Ce module traite tous les "appels système" offerts aux utilisateurs. En effet, un point d'entrée (guichet) est implanté en bas de mémoire 0005H pour permettre l'accès à un service moniteur quelconque.

Sous CP/M le module "BDOS" est entièrement résident; par contre, sous MP/M, dans le cas de certaines configurations à mémoires organisées par bancs, le module "BDOS" se divise en deux parties:

— ODOS : partie réentrante en mémoire commune
gestion du dialogue avec la console;
— BNKBDOS : partie non-réentrante qui s'étend sur le banc zéro
gestion des fichiers.

2.3.3. Le XDOS

Le module XDOS contient le noyau logique du MP/M, ainsi que des extensions de la gestion des fichiers.

Le module XDOS comprend les fonctions suivantes:

— gestion des processus par le distributeur "dispatcher";
— gestion des queues ou files d'attentes;
— gestion des événements ou flags;
— gestion de la mémoire;
— gestion de la base de temps;
— le processus "terminal" (TMP);
— l'interpréteur de commandes (CLI), avec le chargeur;
— les fonctions supplémentaires de MP/M.

2.3.3.1. Gestion des processus

La ressource "CPU" est attribuée au processus candidat qui a la plus forte priorité. La détermination de la priorité est donnée par le module de gestion des processus, appelé "dispatcher" ou distributeur.

Chaque processus possède un descripteur de processus "PD"; c'est une table qui contient toutes les informations dont le système a besoin pour connaître les caractéristiques propres du processus:

— numéro de l'utilisateur;
— numéro de console;
— numéro du disque courant;
— priorité du processus;
— nom du processus;
— segment mémoire utilisé;
— zone de travail: chaînons, buffers, pile, etc...

Ces informations sont utilisées par le "dispatcher" pour sauver l'état du processus en cours, déterminer quel processus doit être lancé, et restaurer l'état du processus.

Le module de gestion des processus est sollicité:

— à chaque appel système (CALL BDOS),
— à chaque interruption externe,
— à chaque "top" d'horloge (toutes les 20 ms).

Les processus ayant une même priorité sont régulés cycliquement (round-robin), par le distributeur qui leur octroie des tranches de temps CPU (slice) égales.

2.3.3.2. Gestion des queues

Les queues ou files d'attentes (FIFO: First In First Out) sont principalement utilisées pour la communication de messages entre processus, mais aussi pour synchroniser les processus, et pour résoudre l'exclusion mutuelle (accès à une ressource unique non partageable).

La gestion des queues sous MP/M a été conçue à la fois pour les processus système et les programmes utilisateurs.

En fait, les queues sont traitées d'une manière similaire aux fichiers disque mais sont toujours résidentes en mémoire. Les queues peuvent être créées, ouvertes, écrites, lues et détruites.

Une queue peut être lue ou écrite de manière conditionnelle ou inconditionnelle. Si une lecture est effectuée sur une queue dans laquelle le message est absent, le processus appelant est suspendu par le "dispat-

cher" jusqu'à ce que le message soit écrit dans la queue par un autre processus. De même, si un message est écrit de manière inconditionnelle dans une queue déjà pleine, le processus émetteur attendra que l'autre processus lise les messages stockés pour libérer de l'espace nécessaire au message en attente. C'est ce qu'on appelle la synchronisation de processus par le mécanisme du "producteur-consommateur".

Le système des queues garantit aussi l'exclusion mutuelle entre processus concurrents, en s'assurant que les phases critiques ne peuvent pas être interrompues durant leur exécution (queues de type MX).

La structure de donnée d'une queue comporte à la fois le bloc de contrôle de queue "QCB" qui doit être résident, et le bloc utilisateur de contrôle de queue "UQCB" situé dans le programme utilisateur.

Il existe deux types de "QCB" : les queues circulaires et les queues chaînées. Ces types dépendent de la taille du message : les messages de 0 à 2 octets utilisent les queues circulaires, tandis que les messages de 3 octets ou plus utilisent les queues chaînées.

2.3.3.3. Gestion des événements

La gestion des événements logiques est utilisée par MP/M pour synchroniser les tâches qui ont lancé des processus asynchrones (entrées-sorties par exemple). Les événements logiques se substituent aux événements physiques (interruptions), car MP/M est indépendant de l'environnement matériel, donc des interruptions, simulant ainsi l'environnement physique. MP/M supporte 32 "flags" du type événement.

Les opérations sur les événements sont :

— attente d'un événement (Wait),
— activation d'un événement (Flag).

2.3.3.4. Gestion mémoire

MP/M gère la mémoire dans des segments prédéfinis à la génération du système. Huit segments mémoire de 0 à 64 Kilo-octets peuvent être gérés, que la mémoire physique soit monolithique ou structurée en "bancs", avec ou sans protection. Les segments mémoire sont décrits de manière interne par des descripteurs de mémoire MD (Memory Descriptor).

Les bancs sont des partitions fixes et totalement indépendantes. On remarque qu'à un instant "t" le CPU n'adresse qu'un banc, c'est-à-dire 16 + 48 K = 64 Kilo-octets. Les programmes ne sont pas réentrants d'une partition à l'autre.

2.3.3.5. Gestion du temps

La gestion du temps sous MP/M se répartit selon deux processus: le processus "Tick" et le processus "Clock".

Le processus "Tick", qui est réveillé toutes les 20 milli-secondes, détermine la période allouée au CPU pour chacun des processus actifs. Cette fréquence ne doit pas être trop élevée car une augmentation significative de l'"overhead-time", due à une commutation excessive des processus, serait à craindre. Inversement, avec une fréquence trop basse, les processus garderaient la ressource CPU pendant une trop longue période, pénalisant ainsi les autres processus.

Le processus "Clock", quant à lui, est activé par le processus "Tick" toutes les secondes. Sa fonction est de maintenir et de fournir la date (jour, mois, année, heure, minute, seconde).

De plus, des primitives système offrent la possibilité de réguler des programmes à charger et à exécuter à un instant choisi à l'avance, ainsi que de déclencher l'exécution d'un processus pendant une certaine période.

2.3.4. Zones système de travail

2.3.4.1. SYSTEM.DAT

Ce segment situé au fond de la mémoire occupe 256 octets (une page). Il contient les informations nécessaires au "chargeur du système" pour reconfigurer dynamiquement le système. Ces informations sont introduites par le générateur de système GENSYS.

00-00	dernière page de la mémoire physique (00 = toute la mémoire)
01-01	nombre de consoles connectables
02-02	numéro de l'interruption réservée au debug DDT (breakpoint)
03-03	booléen d'allocation de pile pour les "CALL" système
04-04	booléen indiquant si la mémoire est organisée en bancs
05-05	booléen indiquant si le CPU est un Z 80
06-06	booléen indiquant si le BDOS est banqué (BNKBDOS)
10-2F	table des segments mémoire initiaux
30-4F	table des points d'arrêt de DDT
50-6F	pile pour les appels système des utilisateurs

Après chargement, ce segment est utilisé pour y ranger certains paramètres de fonctionnement.

2.3.4.2. CONSOLE.DAT

La taille de ce segment est proportionnelle au nombre de consoles déclarées à la génération du système. Chaque console requiert 256 octets qui contiennent : le descripteur du processus terminal (TMP), une pile et des tampons pour les entrées-sorties console.

1.3.4.3. USERSYS.STK

Ce segment optionnel contient 64 octets d'espace pile par segment mémoire utilisateur. Il est utilisé comme une pile temporaire lorsque les programmes utilisateurs font des appels système (CALL BDOS).

2.4. STRUCTURE DU SYSTÈME MP/M

2.4.1. Structure mémoire

MP/M étant un système multi-utilisateur, il implique la présence simultanée de plusieurs programmes en mémoire.

Le système occupe au minimum 15 Kilo-octets et au maximum 20 Kilo-octets en mémoire. Le reste de la mémoire, découpé en segments (huit au maximum), est réservé aux programmes utilisateurs, (segment = TPA). Chaque segment mémoire est, de plus, divisé en deux régions : le SPA (System Parameter Area) et L'UCA (User Code Area).

2.4.1.1. Le SPA

La première région SPA (zone des paramètres système), occupe les 256 premiers octets du segment mémoire. Cette zone est aussi nommée "page de base du segment".

2.4.1.2. L'UCA

La seconde région du segment mémoire utilisateur l'UCA (zone de code utilisateur), débute à l'adresse 0100H relative à la base du segment mémoire. Lorsqu'un programme est chargé, son code est placé dans le segment mémoire au début de l'UCA.

Les programmes utilisateurs sont chargés en mémoire par l'interpréteur de commande "CLI" (Commande Line Interpreter). CLI reçoit les commandes du "TMP" (Terminal Message Process), lequel les lit sur l'entrée console.

TMP est un programme réentrant avec lequel les utilisateurs peuvent communiquer en tapant des lignes de commandes.

Chaque commande est précédée d'un préfixe "prompt" ou vecteur de "login" qui indique le numéro de l'utilisateur suivi du nom du disque courant. Les différentes formes syntaxiques acceptées pour une commande sont :

```
commande
commande fichier1
commande fichier1 fichier2
```

où "commande" est soit une "queue" soit un programme utilisateur.

Lorsque CLI reçoit une commande il fait l'analyse syntaxique de la première partie et essaye d'ouvrir la "queue". Si l'ouverture de la "queue" réussit, la suite de la commande est recopiée dans la "queue" et l'opération CLI est terminée.

Si la "queue" n'existe pas, CLI tente d'ouvrir un fichier du type "PRL" (programme relogeable) dont le nom est celui de la commande. Si l'ouverture du fichier réussit, l'en-tête du fichier PRL est lu pour déterminer les arguments mémoire (taille du programme par exemple). Une demande d'allocation mémoire relogeable est faite pour obtenir un segment mémoire dans lequel sera chargé et exécuté le programme. Si la

demande est satisfaite le fichier PRL est chargé dans le segment et est exécuté, terminant ainsi l'opération CLI.

Si le fichier PRL n'existe pas, alors CLI tente d'ouvrir un fichier de même nom de type COM. Si l'ouverture réussit, alors une demande d'allocation mémoire absolue est faite, à partir du TPA basé en 0100H. Si cette demande est satisfaite, le fichier de type COM est lu dans le segment absolu du TPA et est exécuté, terminant le processus CLI.

CLI crée un descripteur de processus pour chaque programme qu'il charge, initialise une pile de travail, et lui octroie une priorité implicite.

2.4.2. Structure disque

Comme pour CP/M les deux premières pistes du disque (pistes 0 et 1) sont réservées, mais le système MP/M, trop volumineux, qui ne peut y contenir, est alors placé dans un fichier appelé "MPM.SYS". Ce fichier représente l'image mémoire exacte du MP/M, qui sera chargée lors de l'opération de démarrage à froid (cold start ou cold boot).

Les deux pistes réservées contiennent un "bootstrap" sur le secteur 1, et le chargeur du système MP/M qui est découpé en deux parties: MPMLDR et LDRBIOS.

MPMLDR est le chargeur logique du système MP/M, indépendant de l'environnement matériel, qui charge le fichier "MPM.SYS" en mémoire et adapte sa configuration à l'aide du fichier "SYSTEM.DAT".

LDRBIOS est par contre un "mini-BIOS", dépendant de l'environnement physique (disque souple ou dur, sectorisé "soft" ou "hard"). Les primitives de LDRBIOS, situées à des adresses bien précises, sont utilisées par MPMLDR.

piste	secteur	adresse mémoire	nom du module
00	01	adresse du boot	BOOT
00 .. 00	02 .. 25	0100H 0C80H	MPMLDR
00 01 .. 01	26 01 .. 19	0D00H 0D80H 1680H	LDRBDOS
01 .. 01	20 .. 26	1700H 1A00H	LDRBIOS

Fig. 10 — *Implantation disque des modules de chargement*

2.5. LES COMMANDES SOUS MP/M

La compatibilité au niveau du dialogue homme-machine a été maintenue entre CP/M et MP/M. Il en résulte que toutes les commandes déjà disponibles sous CP/M restent valables sous MP/M.

Cependant une différence subsiste sous MP/M qui peut accepter deux catégories de commandes : les commandes de type COM, entièrement compatibles avec celles de CP/M, qui s'exécutent à partir de l'adresse 0100H de la mémoire dans le TPA, et les commandes de type PRL (Program relocatable) mieux adaptées au contexte MP/M, qui s'exécutent dans l'un des segments mémoire définis à la génération du système. Des utilitaires permettent de convertir des fichiers exécutables de type HEX en type PRL (GENMOD), ou de type PRL en type COM (PRLCOM).

Le lancement de l'exécution d'un programme sous MP/M se fait en tapant son nom après avoir reçu le prompt d'invitation. Celui-ci, appelé aussi "vecteur de login", est constitué du numéro de l'utilisateur suivi du nom du disque. Le "prompt" se termine toujours par le caractère ">".

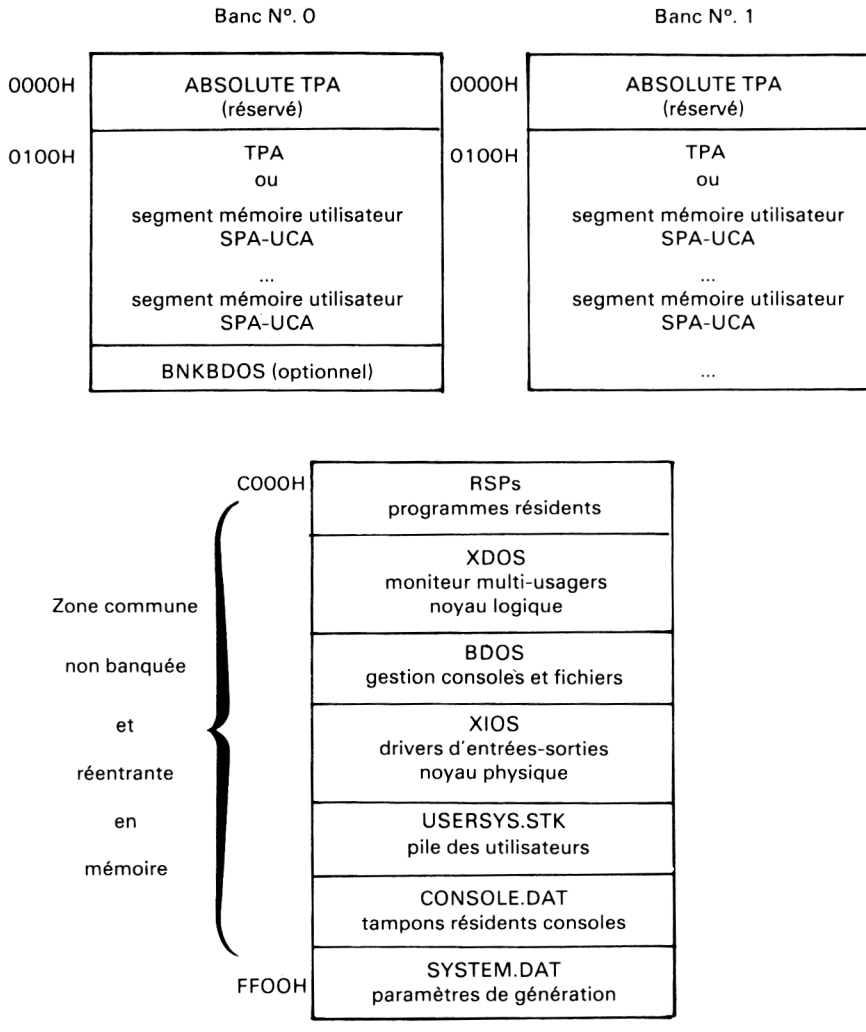


Fig. 11 — *Implantation mémoire de MP/M*

Exemple :

```
0A>commande
```

Le caractère de fonction <ctrl-C> tapé au clavier permet d'abandonner autoritairement l'exécution du programme en cours, on dit aussi "avorter" un programme.

La possibilité de détacher de la console un programme en cours d'exécution est donnée à l'utilisateur par le caractère de fonction <ctrl-D>. L'usager peut alors demander l'exécution d'un autre programme à condition que le premier, qui s'exécute normalement, contrôle l'état (status) de la console. Pour revenir au programme précédemment détaché, il suffit de taper la commande ATTACH suivie du nom du programme à rattacher à la console.

Exemple :

```
0A>PIP                    appel de PIP
*                         préfixe de PIP
↑D                        détachement de PIP
0A>PROG                   appel de PROG
. . . . . .               exécution de PROG
0A>ATTACH PIP             PIP est rattaché à la console
*                         préfixe de PIP
```

2.5.1. Les commandes compatibles CP/M

Comme nous l'avons vu plus haut, la plupart des commandes CP/M, qu'elles soient intégrées au CCP ou non résidentes (fichiers COM), ont été transportées sous MP/M, avec le type PRL.

DIR
ERA
REN
TYPE
STAT
PIP
ED
LOAD
DDT
SUBMIT
DUMP

2.5.2. Les commandes supplémentaires de MP/M

Par rapport au système CP/M, MP/M apporte de nouveaux utilitaires liés à la spécificité d'un contexte multi-usagers: CONSOLE, USER, DSKRESET, ERAQ, DDT-MPM, MPMSTAT, TOD, ABORT, SPOOL, STOPSPLR, SCHED, PRLCOM, GENHEX, GENMOD, GENSYS, MPMLDR.

2.5.2.1. CONSOLE

CONSOLE permet de visualiser le numéro de la console courante.

Exemple:

```
0A>CONSOLE
Console = 1
0A>
```

2.5.2.2. USER

Cette commande est utilisée, soit pour visualiser le numéro utilisateur courant, soit pour changer de numéro de compte. Dans les deux cas le "prompt" ou préfixe d'invitation à taper une commande est visualisé sur le terminal.

Exemple:

```
0A>USER
User Number = 0
0A>USER 5
User Number = 5
5A>                                  préfixe de l'utilisateur no. 5
```

2.5.2.3. DSKREST

Cette commande est utilisée à la suite d'un changement de disque, en particulier de disquette. En effet lorsqu'on introduit une nouvelle disquette, celle-ci est automatiquement protégée en écriture R/O. DSKRESET réalise un "Reset Disk" qui redonne le statut R/W à la disquette à condition qu'aucun autre utilisateur ne l'utilise. Sinon l'opération est inefficace et le message suivant est affiché:

```
Disk reset denied, Drive "d:" Console "c" Program "pppp"
```

où "d" est le nom du disque, "c" le numéro de console et "pppp" le nom du programme en cours d'exécution.

Si aucun argument n'est donné, l'ensemble des disques connectés sera réinitialisé, sinon les arguments représentent les noms des disques séparés par une virgule.

Exemple:

```
0A>DSKRESET
0A>DSKRESET A:, B:
```

2.5.2.4. ERAQ

Destruction d'un ou plusieurs fichiers avec validation par l'utilisateur.

Exemple:

```
OA>ERAQ B:*.BAK
  B:ESSAI    BAK ?Y
  B:TEST     BAK ?Y
  B:SOURCE   BAK ?N
  OA>
```

2.5.2.5. DDT

Le metteur au point DDT de MP/M conserve toutes les fonctionnalités offertes par le DDT de CP/M, avec les quatre commandes additionnelles suivantes: W, V, N, B.

W (Write) — Wn — n = nombre de secteurs à écrire

Cette commande se substitue à la commande SAVE de CP/M qui n'existe pas sous MP/M. La valeur "n" peut être calculée par la commande V. Pour que l'écriture soit effective, il est conseillé d'utiliser la commande "Inom du fichier" qui spécifie sur quel fichier doit s'opérer la sauvegarde de l'image mémoire.

Exemple:

```
-ISAUVE.COM
-W8
```

V (Value)

V
Vv v = taille en octets du programme

Dans sa première forme (V seul) l'adresse du NEXT est retournée, et peut servir pour la seconde forme. Celle-ci renvoie le nombre de secteurs correspondant au programme, qui pourra par la suite être réutilisé par la commande W.

Exemple:

```
-V
NEXT PC
0280 0100
-V280
0003
```

N (Normalize)

N

Cette commande translate un fichier exécutable de type PRL, après avoir été lu par la commande R.

B (Bitmap)

Ba,0 ou Ba,1 a = adresse dans la "MAP"

Cette commande permet de modifier, mise à un ou à zéro, les bits de la "MAP" d'un fichier translatable de type PRL.

2.5.2.6. MPMSTAT

Visualisation de l'état courant du système en fonctionnement, en particulier: nombre de consoles actives, nombre de processus, état des processus, état des queues, état des événements, répartition des processus vis-à-vis des consoles, localisation des processus en cours dans les segments mémoires, etc...

2.5.2.7. TOD

TOD (Time Of Date) permet d'initialiser et de visualiser la date du jour (année, mois, jour, heure, minute, seconde). Cet utilitaire possède un calendrier universel qui permet d'afficher le nom du jour.

Exemple :

```
0A>TOD 05/17/82 10:21:00          17 Mai 1982 10h 21min
Strike key to set time <cr>
Mon 05/17/82 10:21:00
0A>TOD
Mon 05/17/82 10:22:38
```

2.5.2.8. ABORT

Cette commande permet à l'utilisateur d'avorter l'exécution d'un programme qu'il donne en argument :

```
0A>ABORT PROGTEST
```

2.5.2.9. SPOOL

Cette commande permet d'envoyer un fichier texte en ASCII sur le périphérique assigné à l'imprimante, même si celle-ci est occupée. Le fichier à imprimer est placé dans une file d'attente qui est automatiquement vidée dès que l'imprimante devient libre.

Exemple :

```
0A>SPOOL B:EDITION1.PRN,EDITION2.TEXT,EDITION3.DOC
```

La commande STOPSPLR permet d'avorter la sortie sur imprimante.

2.5.2.10. SCHED

Cette commande permet de demander l'exécution d'un programme à partir d'une certaine date et heure définie par l'usager :

```
2A>SCHED 5/17/82 11:15 MAJOUR
```

Le programme MAJOUR sera lancé le 17 mai 1982 à 11h 15min.

2.5.2.11. PRLCOM

Cette commande permet de transformer un fichier exécutable de type PRL en fichier exécutable de type COM.

Exemple :

```
OA>PRLCOM PROGMPM.PRL B:PROGCPM.COM
```

2.5.2.12. GENHEX

Cette commande permet de reconstruire un fichier objet au format Intel hexadécimal de type HEX à partir d'un fichier exécutable de type COM. Un déplacement peut être donné en paramètre pour spécifier l'adresse de début.

Exemple :

```
OA>GENHEX PROGRAM. COM 100
```

2.5.2.13. GENMOD

Cette commande permet de générer un fichier exécutable de type PRL à partir d'un fichier constitué par la concaténation de 2 images d'un même objet de type HEX décalées de 0100H l'un par rapport à l'autre.

Exemple :

```
OA>GENMOD CONCATEN.HEX B:RESULT.PRL
```

Les fichiers de type PRL sont des fichiers exécutables dans un segment mémoire, contrairement aux fichiers de type COM qui sont exécutables dans la zone TPA.

La concaténation des deux fichiers décalés de 100H permet de déterminer les champs adresse du programme, et de les repérer dans une "MAP de bits" contiguë au binaire objet. Pour chaque octet binaire généré est associé un bit qui indiquera au chargeur de programme, s'il est à 1, de translater l'adresse (poids forts) d'une valeur égale à l'adresse début du segment mémoire alloué.

Un espace de 256 octets est aussi réservé en tête de tout fichier PRL où est mémorisée la taille du programme.

Fig. 12 — *Structure d'un fichier PRL*

2.5.2.14. GENSYS

GENSYS configure le système d'après l'environnement matériel et selon les choix de l'utilisateur: nombre de consoles, mémoire avec ou sans banc, numéro de l'interruption utilisée par le DDT pour les "breakpoints", noms des processus résidents du type RSP, etc... L'exemple ci-dessous nous montre le déroulement interactif de la génération:

```
OA>GENSYS
MP/M System Generation

Top page of memory = FF                    taille maximum en pages
Number of consoles = 2                     deux consoles
Breakpoint RST # = 6                       numéro de RST pour DDT
Add system call user stacks (Y/N)? Y
Z 80 CPU (Y/N)? Y
Bank switched memory (Y/N)? Y              bancs mémoire
Banked BDOS file manager (Y/N) N
Enter memory segment table: (ff terminates list)
Base,size,attrib,bank = 0,BC,0,0           segment 0 = BC00H octets
Base,size,attrib,bank = 0,C0,0,1           segment 1 = C000H octets
Base,size,attrib,bank = FF                 fin de description
Select Resident System Processes: (Y/N) Y
SPOOL     ?N
TIME      ?Y
OA>
```

2.5.2.15. MPMLDR

MPMLDR charge en mémoire le fichier "MPM.SYS" après la génération système faite par GENSYS. MPMLDR ne peut s'exécuter que sous CP/M ou par l'opération de "bootstrap" (cold start loader) de MP/M. Au cours du chargement la configuration du système est visualisée comme suit:

```
MP/M Loader

Number of consoles      = 2
Breakpoint RST #        = 6
Z 80 CPU
TOP of memory           = FFFFH
Memory Segment Tables:

SYSTEM      DAT         FF00H       0100H
CONSOLE     DAT         FD00H       0200H
USERSYS     STK         FC00H       0100H
XIOS        SPR         F600H       0600H
BDOS        SPR         E200H       1400H
XDOS        SPR         C300H       1F00H
TIME        RSP         C600H       0300H

Memseg      Usr         0000H       C000H       Bank 01H
Memseg      Usr         0000H       BC00H       Bank 00H

MP/M
0A>
```

2.6. SERVICES SYSTÈME OFFERTS

De la même manière que les commandes les fonctions système offertes par le BDOS du CP/M sont conservées à quelques exceptions près, et de nouvelles fonctions spécifiques à MP/M y ont été rajoutées.

2.6.1. Fonctions supplémentaires du XDOS

— demande et libération d'espace mémoire absolue ou relogeable,
— scrutation de périphériques (mode "polling"),
— attente et activation d'évènements,
— création, ouverture, suppression d'une queue,
— lecture et écriture conditionnelle ou non d'une queue,
— mise en attente d'un délai,
— appel du "dispatcher" de processus,
— création et terminaison d'un processus,

— initialisation de la priorité,
— assignation d'une console,
— envoi d'une commande,
— analyse syntaxique d'un nom de fichier,
— obtention du numéro de console,
— initialisation et restitution de la date,
— abandon d'un processus spécifique.

2.6.2. Primitives système du XIOS

Le XIOS est une extension du BIOS de CP/M ; en particulier on retrouve les mêmes primitives. Des primitives spécifiques au contexte MP/M y ont été rajoutées. Les points d'entrée sont situés à la suite du vecteur de branchement du BIOS.

```
XIOS+51   JMP SELMEMORY     ; sélection d'un banc mémoire
XIOS+54   JMP POLLDEVICE    ; interrogation d'un périphérique
XIOS+57   JMP STARTCLOCK    ; démarrage de l'horloge
XIOS+60   JMP STOPCLOCK     ; arrêt de l'horloge
XIOS+63   JMP EXITREGION    ; sortie d'une phase critique
XIOS+66   JMP MAXCONSOLE    ; nombre maximum de consoles
XIOS+69   JMP SYSTEMINIT    ; initialisation du système
XIOS+72   JMP IDLE          ; mise à l'état repos
```

3

Extension de la famille CP/M

3.1. SYSTÈMES POUR MICROPROCESSEURS 16 BITS CP/M-86, MP/M-86

L'apparition sur le marché, au début des années 80, de microprocesseurs 16 bits a amené Digital Research à développer les systèmes d'exploitation CP/M-86 et MP/M-86 sur le 8086 d'Intel. Une des améliorations qu'apporte le 8086 est la capacité mémoire qui peut atteindre plus d'un Méga-octets.

3.1.1. CP/M-86

La plupart des facilités du CP/M-80 ont été conservées et des améliorations y ont été portées. D'une manière générale, le système CP/M-86 maintient la compatibilité au niveau des fichiers, avec toutes les versions précédentes de CP/M. Le microprocesseur étant différent, cette compatibilité n'est pas assurée au niveau du code machine et gestion mémoire. Pourtant, si les binaires sont différents, des utilitaires permettent de transcoder un fichier exécutable 8080 en un fichier exécutable 8086.

3.1.1.1. Architecture de CP/M-86

A l'inverse de CP/M, CP/M-86 ne réside pas sur les deux premières pistes réservées, mais dans un fichier de nom "CPM.SYS". En effet,

comme il n'y a plus de problème de taille mémoire, CP/M-86 est trop volumineux pour contenir dans ces deux pistes. Celles-ci contiennent le "Cold Start Loader" LDBIOS dont le rôle est de charger le fichier CPM.SYS en mémoire.

Sur le plan de la structure interne de CP/M-86 on retrouve les mêmes modules qui existaient sous CP/M :

— le CCP : interface utilisateur-machine
— le BDOS : gestion des fichiers et fonctions système
— le BIOS : noyau physique lié à l'environnement.

Le CCP et le BDOS occupent environ 10 K-octets, tandis que la taille du BIOS varie selon les installations. Le CCP ne peut être recouvert par des programmes en zone TPA, ce qui rend le CP/M-86 totalement résident, alors que ce n'était pas le cas sous CP/M.

3.1.1.2. Les commandes CP/M-86

Le code binaire étant différent par rapport à CP/M, les fichiers exécutables sont reconnus par le type "CMD". Comme pour CP/M c'est le prompt "A>" qui invite l'utilisateur à taper une commande.

Les commandes internes au CCP telles que :

DIR ERA REN TYPE USER

offrent les mêmes fonctionnalités que sous CP/M. De même les utilitaires non résidents tels que :

STAT PIP END SUBMIT

opèrent de manière identique.

Par contre, les produits : ASM86 et DDT86, tout en gardant la philosophie des produits ASM et DDT, ont été largement modifiés.

ASM86 lit un source en assembleur 8086 de type A86 et produit trois fichiers résultats : un objet de type H86 au format Intel, un listing de type LST, et une table des symboles de type SYM.

La syntaxe de la commande ASM86 accepte des arguments si le caractère "$" est présent. Ces arguments sont constitués de deux lettres :

la première représente le type de fichier (A = source, H = objet, P = listing, S = symbole) et la seconde le nom du disque associé.

Exemple d'appel :

```
A>ASM86 MODULE $AB HB PC SB
```

le programme "MODULE.A86" qui se trouve sur le disque B, est assemblé et produit un objet de type H86 sur le disque B, un listing de type LST sur le disque C, et la table des symboles sur le disque B.

La commande GENCMD (Generate CMD) se substitue également à la commande LOAD de CP/M, en créant un fichier exécutable de type CMD à partir d'un fichier objet au format Intel 8086 de type H86.

La commande LDCOPY (Loader Copy) remplace aussi la commande SYSGEN.

Les commandes ASM86, GENCMD sont aussi fournies avec le type COM pour faire du développement croisé sous CP/M-80.

3.1.2. MP/M-86

Le système MP/M-86 constitue le haut de gamme de la famille des systèmes d'exploitation de Digital Research. En effet il rassemble les fonctionnalités spécifiques au microprocesseur 16 bits 8086 issues de CP/M-86, et celles du MP/M multi-utilisateurs, multi-postes, multi-tâches.

3.1.2.1. Architecture de MP/M-86

Comme pour MP/M, le MP/M-86 est organisé en plusieurs modules :

- le TMP : interface de dialogue avec les utilisateurs
- le SUP : superviseur et traitement des appels système
- le RTM : distributeur de processus et gestion des queues
- le MEM : module de gestion de la mémoire
- le CIO : module de gestion des entrées-sorties caractères
- le BDOS : système de gestion des fichiers
- le XIOS : noyau physique de traitement des entrées-sorties

On remarque que l'ancien XDOS de MP/M a été éclaté en quatre modules indépendants (SUP, RTM, CIO, BDOS) sous MP/M-86

SUP — Le superviseur gère les interactions entre les processus utilisateurs et les autres modules du système. Tous les appels système provenant des utilisateurs ou des modules internes passent par le superviseur qui joue le rôle d'un guichet. L'interpréteur de commande CLI, qui peut être appelé à n'importe quel niveau, fait aussi partie du superviseur.

RTM — Le moniteur temps réel RTM est le noyau de "multi-tasking" du système. Il permet de gérer la commutation et l'allocation CPU des processus. En plus de la distribution des processus, RTM assure la gestion des queues, la gestion des événements logiques (flags), l'interrogation cyclique des périphériques (polling), et la gestion des bases de temps.

CIO — Ce module gère les entrées-sorties mode caractère pour les consoles ou les imprimantes. A chaque périphérique est associé un CCB (bloc de contrôle caractère) qui contient des informations concernant le propriétaire, le périphérique et les caractèristiques d'édition.

BDOS — Ce module représente le système de gestion de fichiers classique déjà connu sous les autres systèmes. Il offre à l'utilisateur toutes les fonctions de gestion de fichiers, et gère l'allocation et la libération de l'espace disque. Des améliorations ont été portées par rapport à MP/M pour résoudre les conflits d'accès au niveau des fichiers (verrouillage, partage d'accès, réservation d'article, etc...).

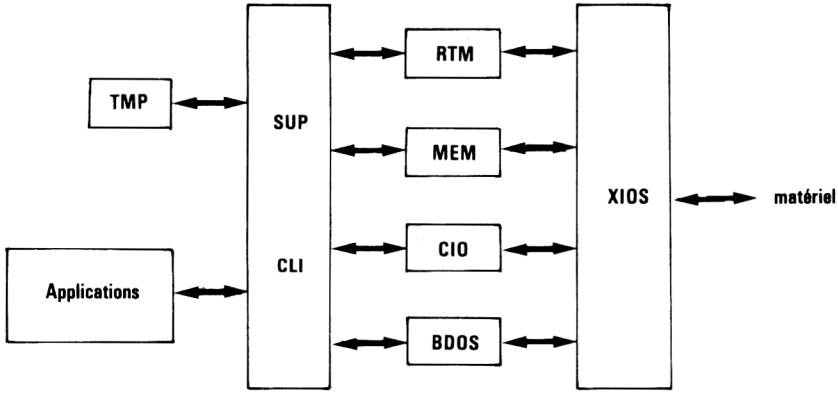


Fig. 13 — *Architecture des modules de MP/M-86*

3.1.2.2. Les commandes MP/M-86

Le "prompt" invitation à taper une commande est le même que sous MP/M, ainsi que la syntaxe de la commande. MP/M-86 apporte en plus la notion de protection individuelle au niveau d'un fichier. Un mot de passe peut être spécifié dans une référence de fichier selon la syntaxe :

d:nom du fichier.type;mot de passe

Exemple :

A:DOCUMENT.DST;SECRET

Les commandes elles-mêmes, qui sont des fichiers exécutables de type CMD (voir CP/M-86), peuvent être assorties d'un mot de passe si l'on désire qu'elles ne soient pas publiques (commandes privilégiées).

Exemples :

OA>GENSYS;SESAM

4C>B:ESPION;SPY

Les commandes de MP/M-86 regroupent la plupart des commandes de MP/M et CP/M-86. Quatre nouvelles commandes sont rajoutées :

PRINTER	permet de visualiser ou de changer les numéros des imprimantes connectées au système.
SDIR	permet de visualiser les catalogues de fichiers (directory) avec de nombreuses options.
SET	permet de modifier les niveaux de protection (mot de passe), la rétention et les attributs d'un fichier.
SHOW	permet de visualiser l'état d'un disque et ses protections.

3.2. SYSTÈMES ORIENTÉS RÉSEAUX : CP/NET, CP/NOS, MP/NET

CP/NET, CP/NOS et MP/NET, développés par Digital Research, sont des systèmes d'exploitation orientés "réseaux". Ils permettent de faire communiquer entre eux plusieurs micro-ordinateurs à base de CP/M et MP/M, d'accéder à des ressources communes, et de partager les organes d'entrées-sorties.

3.2.1. CP/NET

CP/NET est le premier système de la famille "réseau". Il permet d'interconnecter deux micro-ordinateurs, dont l'un fonctionne sous MP/M (système maître) et l'autre sous CP/M (système esclave). C'est le système maître qui à l'initiative du dialogue et gère les ressources partageables. Cependant l'accessibilité aux ressources communes est possible à partir des deux systèmes maître et esclave.

3.2.2. CP/NOS

CP/NOS est un système qui permet de faire communiquer un système MP/M maître avec un système esclave CP/M sur PROM et RAM réduit à sa plus simple expression, c'est-à-dire un CP/M sans support disque. Seules, la gestion de la console et de l'imprimante sont supportées.

3.2.3. MP/NET

MP/NET représente le haut de gamme de la famille "réseaux". Il permet de connecter un ou plusieurs systèmes MP/M entre eux. Il n'y a plus de distinction entre système maître et système esclave, le réseau devenant alors totalement symétrique. L'ensemble constituant un environnement "multi-micro-ordinateurs".

3.2.4. Architecture du système CP/NET

Le système esclave CP/M de CP/NET est logiquement divisé en quatre modules.

— BIOS : identique à celui du CP/M
— BDOS : identique à celui du CP/M
— SNIOS : (Slave Network I/O System) extension réseau du BIOS
— NDOS : (Network Disk Operating System) extension réseau du BDOS

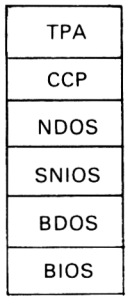


Fig. 14 — *Implantation mémoire des modules de CP/NET esclave*

Le système maître MP/M de CP/NET est bâti autour d'un MP/M qui possède deux modules supplémentaires : un module logique SLVSP résident de type RSP qui gère les fonctions logiques d'échange de messages avec le système esclave, et le module NETWRKIF qui contient les sous-programmes de gestion des entrées-sorties spécifiques à l'environnement "réseau". Ce dernier module, comme le module SNIOS du système esclave, est fourni à l'utilisateur sous la forme d'un "source", dans l'éventualité où des modifications liées au matériel ou à la procédure de communication seraient à rajouter.

3.2.5. Les commandes CP/NET

CP/NET inclut de nouvelles commandes spécifiques "réseau" par rapport à CP/M et MP/M que nous allons détailler ci-dessous.

LOGIN — permet à un utilisateur de se connecter sur le réseau pour accéder aux ressources communes. Un mot de passe facultatif peut être donné.

A>LOGIN <mot de passe>

LOGOFF — déconnecte l'utilisateur d'un système esclave du système maître.

A>LOGOFF

SNDMAIL — permet d'envoyer un message d'un système esclave vers un système esclave ou maître.

A>SNDMAIL <destinataire> "message à envoyer"

RCVMAIL — permet d'obtenir tous les messages postés par un système maître.

A>RCVMAIL

NETWORK — permet pour un utilisateur esclave de mettre à jour la table de configuration des ressources du système esclave.

A>NETWORK <périph.local> = <périph.maître>

LOCAL — permet de réassigner au système local les périphériques déjà assignés au réseau.

```
A>LOCAL <périph.local>
```

DSKRESET — est identique à celle de MP/M sur le système local.

```
A>DSKRESET
A>DSKRESET A:,B:
```

ENDLIST — permet d'envoyer une fin de fichier <ctrl-Z> sur le périphérique assigné à l'imprimante.

```
A>ENDLIST
```

CPNETLDR — permet de charger en mémoire les modules du système CP/NET : SNIOS.SPR et NDOS.SPR.

```
A>CPNETLDR
```

CPNETSTS — visualise la table de configuration du système esclave.

```
A>CPNETSTS
```

BROADCST — permet à un utilisateur d'un système maître d'envoyer un message à tous les autres utilisateurs.

```
OA>BROADCST "message à diffuser à tous"
```

MSNDMAIL — permet à un utilisateur du système maître d'envoyer un message vers un système esclave.

```
3A>MSNDMAIL <destinataire> "message"
```

MRCVMAIL — permet d'obtenir tous les messages postés par les systèmes esclaves.

```
2B>MRCVMAIL
```

SPOOL — permet aux utilisateurs d'utiliser l'imprimante du système maître.

```
1D>SPOOL fichier1,fichier2,...
```

4

Produits développés autour de CP/M et MP/M

4.1. LE PHÉNOMÈNE CP/M

Parallèlement au développement de CP/M et de ses extensions, de nombreuses sociétés ont produit des logiciels compatibles CP/M. C'est cette vaste bibliothèque de programmes qui a fait de CP/M un véritable phénomène, et par là un Standard "de facto". Ces produits tels que macro-assembleurs, interpréteurs, compilateurs, éditeurs de documents, utilitaires, ne sont pas fournis avec le système standard CP/M, mais peuvent être disponibles sous forme d'options, ou achetés dans de véritables supermarchés de logiciel.

4.2. MACRO-ASSEMBLEURS

4.2.1. MAC

En plus de l'assembleur ASM qui fait partie du "package" CP/M, Digital Research a développé le macro-assembleur MAC qui a de nombreuses similitudes avec le macro-assembleur ASM-80 d'Intel.

MAC produit en sortie les fichiers classiques du type PRN (imprimante) et HEX (objet), ainsi que le fichier contenant la table des symboles de type SYM, utilisable par le metteur au point SID.

Le "package" MAC inclut en outre un ensemble de macro-instructions dans le fichier SEQIO.LIB qui simplifie l'interface entre le programme assembleur et les services offerts par le système. Un autre fichier appelé Z 80.LIB offre à MAC la possibilité d'assembler les instructions du Z 80. Exemple de mise en œuvre :

```
A>MAC B:UTIL
```

4.2.2. MACRO-80

Le macro-assembleur MACRO-80 ou M80 de Microsoft est également conforme aux spécifications du macro-assembleur ASM-80 d'Intel. MACRO-80 permet d'assembler des programmes écrits en 8080 ou Z 80 et produit en sortie un fichier objet du type REL (objet relogeable), qui, traité par LINK-80 donne un fichier exécutable de type COM.

Les autres langages de Microsoft (BASIC-80, FORTRAN-80 et COBOL-80), peuvent facilement s'interfacer à l'aide de références globales avec des sous-programmes écrits en MACRO-80.

L'éditeur de liens LINK-80, l'utilitaire de références croisées CREF-80 et le gestionnaire de librairie LIB-80 font partie intégrante du "package" MACRO-80, et sont livrés avec.

L'appel au macro-assembleur peut se formuler de deux manières :

```
    M80
ou
    M80 fichier-objet,fichier-listing=fichier-source
```

Dans le premier cas, le prompt "*" est renvoyé à l'utilisateur qui peut spécifier ses fichiers comme ils sont exprimés dans la seconde forme. Seulement le signe égal et le nom du fichier source qui le suit obligatoires ; le nom du fichier objet prend alors le type REL par défaut, et le type PRN est associé au listing. Exemples :

```
A>M80
*OBJET,LISTING=SOURCE        assemble SOURCE.MAC
                             produit LISTING.PRN et OBJET.REL
A>M80 PROG,PROG=PROG         assemble PROG.MAC
                             produit PROG.PRN et PROG.REL
```

MACRO-80 accepte aussi les options à la fin de la chaîne de commande, précédées par le caractère "/".

O représentation en octal du code généré
H représentation en hexadécimal du code généré
R forçage de la génération d'un fichier objet de type REL
L forçage de la génération d'un fichier listing de type PRN
C forçage de la génération d'un fichier références croisées
Z assemblage d'un programme écrit en Z 80.

Exemple:

A>M80 PROG/R/L/Z — le source est en assembleur Z 80 (Z) produit PROG.REL (R) et PROG.PRN (L)

4.3. INTERPRÉTEURS

4.3.1. MBASIC

L'interpréteur BASIC version 5 de Microsoft, appelé MBASIC, est devenu un véritable standard. A l'inverse des autres interpréteurs, MBASIC traduit chaque ligne source en code binaire intermédiaire "dans la foulée", ce qui rend le temps de traduction pratiquement inperceptible.

Le code intermédiaire extrêmement compact est interprété lorsque l'utilisateur tape la commande RUN. De plus les programmes sont faciles et rapides à développer. MBASIC dispose en effet d'un éditeur incorporé et permet de stopper l'exécution du programme à tout moment pour visualiser ou changer les valeurs des variables, et relancer l'exécution. Les principales améliorations apportées à MBASIC concernent:

— noms des variables plus longs (jusqu'à 40 caractères);
— types de variables: INTEGER, REAL, DOUBLE PRECISION, STRING;
— programmation structurée des boucles: instructions WHILE/WEND;
— numérotage et renumérotage automatique des lignes (AUTO/RNUM);

— représentation en octal, hexadécimal et binaire de l'information ;
— plusieurs instructions par ligne, séparées par un ":" ;
— trace des numéros de ligne durant l'exécution (commandes TRON/TROFF).

Exemple :

A>MBASIC	appel de BASIC
LOAD "B:PROGBAS"	chargement du programme en mémoire
RUN	lancement de l'exécution
SAVE "PROBASIC",A	sauvegarde du source en ASCII
SYSTEM	retour au système
A>	

4.3.2. CBASIC

CBASIC de Compiler Systems est un interpréteur BASIC non interactif. Trois phases sont nécessaires pour exécuter un programme :

— la construction du programme par un éditeur de texte indépendant ;
— la traduction du source en un fichier intermédiaire de type INT ;
— l'exécution proprement dite par interprétation du code intermédiaire ;

La commande CBASIC réalise la traduction, et la commande CRUN lance l'interprétation du fichier INT contenant le code intermédiaire.

Une des caractéristiques de CBASIC est que les numéros de lignes ne sont pas obligatoires.

Syntaxe de mise en œuvre :

CBASIC source	compile le programme "source.BAS" et produit un fichier de type INT.
CRUN objet	interprète le fichier de type INT.

4.3.3. PASCAL/M

PASCAL/M de Sorcim est un interpréteur de P-code PASCAL. Le P-code est un langage intermédiaire indépendant de la machine cible.

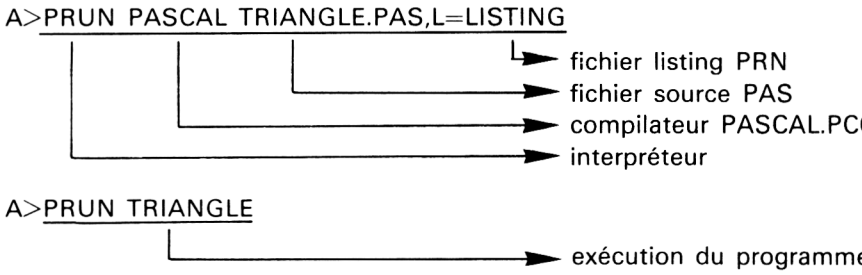
Le produit PASCAL/M comprend trois fichiers :

— PASCAL.PCO : compilateur PASCAL sous forme de P-code ;
— PASLIB.PCO : librairie PASCAL (Run-time) sous forme de P-code ;
— PRUN.COM : interpréteur PASCAL de P-code.

La syntaxe de PASCAL/M est :

PRUN PASCAL source	interprète le compilateur PASCAL et traduit le source PAS en P-code PCO
PRUN objet (P-code)	interprète le programme utilisateur traduit en P-code.

Exemple :



4.3.4. CIS-COBOL

CIS-COBOL, conforme à la norme ANSI-74, produit à partir d'un source de type COB, un code intermédiaire compacté de type INT exécutable par l'interpréteur RUN. Ce compilateur est fortement orienté vers le traitement conversationnel, et de nombreux dispositifs lui permettent de gérer un écran vidéo : positionnement du curseur, verbes ACCEPT et DISPLAY. CIS-COBOL permet aussi la gestion des fichiers séquentiels, directs et indexés, et le recouvrement automatique de programmes.

Syntaxe de mise en œuvre :

COBOL source	compilation d'un source de type COB
RUN objet.INT	interprétation du programme compilé

4.3.5. MU-LISP

MU-LISP de Microsoft est un interpréteur de traitement de listes orienté "intelligence artificielle". Il contient 83 fonctions LISP et peut manipuler des nombres avec une précision infinie.

4.4. COMPILATEURS

4.4.1. BASCOM

Le compilateur BASCOM de Microsoft est le prolongement de l'interpréteur MBASIC, et entièrement compatible avec lui sur le plan du langage.

BASCOM produit, à partir d'un programme source BASIC de type BAS, un fichier intermédiaire relogeable de type REL, qui repris par l'éditeur de liens LINK-80 fournit du code machine 8080 ou Z 80 dans un fichier de type COM.

Le caractère interactif de mise au point qui existait sous MBASIC est perdu au profit de la vitesse d'exécution du programme compilé par BASCOM. Il est donc fortement conseillé de développer un programme avec MBASIC, et une fois au point, de le compiler par BASCOM pour l'exécuter.

Syntaxe de mise en œuvre :

```
BASCOM objet,listing=source/options
```

Se rapporter à la syntaxe de MACRO-80 entièrement compatible avec celle de BASCOM. Exemple :

```
A>BASCOM PROGBIN,PROGLIS=PROGBAS
```

4.4.2. SBASIC

SBASIC est un véritable compilateur structuré qui engendre du code 8080 dans un fichier de type HEX. Les instructions de contrôle du type PASCAL y ont été rajoutées : WHILE...DO, IF...THEN...ELSE,

BEGIN...END. Les "fonctions" et les "procédures" constituent une autre caractéristique de SBASIC.

4.4.3. PASCAL/MT+

Le PASCAL/MT+ est issu de la combinaison du PASCAL standard (norme ISO), du PASCAL/MT (sous-ensemble du PASCAL standard), et d'un metteur au point symbolique. Il accepte les variables réelles en virgule flottante ou en BCD. Le langage d'assemblage est interfacé ainsi que les entrées-sorties et les procédures d'interruption.

PASCAL/MT+ est un compilateur qui produit un module relogeable de type ERL compatible avec l'éditeur de liens LINK MT.

4.4.4. PASCAL/Z

Le PASCAL/Z d'Ithaca Intersystems est un compilateur qui produit uniquement du code machine Z 80. En fait la compilation se fait en deux étapes: la traduction du source PASCAL en code mnémonique assembleur Z 80 dans un fichier de type SRC, et l'assemblage proprement dit qui produit le binaire Z 80 dans un fichier au format Intel de type HEX.

Syntaxe de mise en œuvre:

```
PASCALZ source.options
```

où "source" représente le nom du fichier PASCAL de type PAS, et "options" une liste optionnelle de trois lettres (noms des disques pour les fichiers de type PAS, SRC, LST).

```
ASMBL source1.opt1, source2.opt2,...
```

où les "sourcen" représentent les noms des fichiers traduits en assembleur Z 80 de type SRC, et "optn" la liste optionnelle des noms des disques supportant les fichiers de type SRC, HEX, LST.

Exemple :

A>PASCALZ PROPASCA	compilation proprement dite
A>ASMBL LIBS,PROPASCA	assemblage de programme PROPASCA.SRC avec la librairie PASCAL :LIBS.SRC

4.4.5. FORTRAN-80

Le FORTRAN-80 de Microsoft est conforme à la norme ANSI-66 excepté pour les "complexes". FORTRAN-80 est distribué avec sa librairie FORLIB.REL ainsi que LINK-80 et les produits qui vont avec.

Syntaxe de mise en œuvre :

```
F80 objet,listing=source
```

Le source doit être du type FOR et le fichier objet produit est du type REL. Cet objet est ensuite édité par LINK-80 pour engendrer un fichier exécutable de type COM.

Exemple :

```
A>F80 CALCUL,LST:=CALCUL
```

4.4.6. COBOL-80

Le compilateur COBOL-80 est conforme à la norme ANSI-74, et aux spécifications de Microsoft : source de type COB et production d'un objet relogeable de type REL. COBOL-80 permet de gérer des fichiers séquentiels, relatifs et indexés. La gestion intéractive de l'écran est facilitée avec les verbes ACCEPT et DISPLAY qui permettent de réaliser des transferts de pleine page. Le morcellement des programmes qui dépassent la taille mémoire disponible est possible.

Ce compilateur étant très volumineux, il est lui-même morcelé en plusieurs parties :

COBOL.COM	programme principal du compilateur résident
COBOLn.OVR	quatre overlays de n=1 à n=4
COBLIB.REL	librairie "Runtime"
CRTDRV.REL	"driver" pour terminal vidéo

La compilation se réalise en cinq phases. "COBOL.COM" compile l'IDENTIFICATION DIVISION et l'ENVIRONNEMENT DIVISION. "COBOL1.OVR" compile la DATA DIVISION et "COBOL2.OVR" la PROCEDURE DIVISION. Ces trois phases constituent la première passe de compilation et un fichier traduit en code intermédiaire "STEXT.INT" est créé. "COBOL3.OVR" traduit alors ce fichier intermédiaire en un fichier relogeable de type REL. Et enfin "COBOL4.OVR" alloue les FCB nécessaires et détruit le fichier temporaire.

Syntaxe de mise en œuvre:

```
COBOL objet,listing=source
```

Exemple:

```
A>COBOL GESTION,TTY:=GESTION
```

4.4.7. C

Le langage C initialement développé par les Bell Laboratories pour le système d'exploitation UNIX, est un langage de haut niveau dont la structure ressemble à celle de PASCAL sans les types. Ce langage orienté "développement de logiciel", manipule les pointeurs, les structures, permet de faire des inclusions de segments "source" et des compilations séparées. La formulation des expressions lui permet d'engendrer un code objet beaucoup plus compact, certaines variables peuvent par exemple être introduites dans les registres du CPU.

Il existe plusieurs compilateur C disponibles sous CP/M: le "C" de Whitesmiths, le "C" de BDS, le "Small C", le "Tiny C", etc...

Le C de Whitesmiths est l'outil le plus puissant en matière de "Langage C". Il dispose de plus de 75 fonctions pour gérer les entrées-sorties, les chaînes de caractères et l'allocation mémoire. Le code produit est du type REL compatible avec l'éditeur de liens LINK-80. Le compi-

lateur est lancé par un appel à SUBMIT sur une commande cataloguée de type SUB. Le fichier résultat de type COM produit après édition de liens est directement exécutable.

4.5. ÉDITEUR DE LIENS LINK-80

L'éditeur de liens LINK-80 de Microsoft permet de relier entre eux des modules objets de type REL, et de construire un fichier unique exécutable de type COM. En fait, les objets de type REL sont indépendants du langage machine de la machine cible, et constituent un code objet intermédiaire spécifique des compilateurs de Microsoft: BASIC-80, FORTRAN-80, COBOL-80 et MACRO-80. C'est LINK-80 qui est chargé d'interpréter ce code et de générer le code machine associé.

La syntaxe d'appel à l'éditeur de liens peut prendre deux formes:

```
    L80
ou
    L80 objet1,objet2,...,objetn
```

Dans le cas de la première forme, un prompt "*" est envoyé sur la console, où l'utilisateur peut taper la suite des fichiers objets comme elle est définie dans la seconde forme.

Quelques paramètres supplémentaires précédés par un "/" peuvent être rajoutés derrière chaque nom de fichier.

E Génération du code binaire en mémoire et retour système
F Demande d'exécution du programme après l'édition de liens
M Édition de la table des références globales (librairies)
N Le nom du fichier exécutable est sauvé avec le type COM

Exemple:

```
A>L80 PROGMAC,PROGCOB,B:PROGBAS,PROGRAM/M/N/E
```

4.6. TRAITEMENT DE TEXTE

4.6.1. WORDSTAR

Le logiciel WORDSTAR de Micropro est un éditeur orienté "traitement de texte", qui combine à la fois l'édition de texte classique et l'édition de documents. WORDSTAR est un éditeur dit "pleine page" qui nécessite un écran vidéo avec gestion du curseur en X et Y.

Cet éditeur visualise directement le document sur l'écran au moment où l'utilisateur le demande. Les lignes du texte sont automatiquement justifiées par rajout de séquences <cr> <lf> à l'endroit adéquat, et par insertion d'espaces supplémentaires relativement bien répartis entre les mots de la ligne. Des paramètres de "formattage" du texte peuvent à tout moment être définis par l'utilisateur: détermination des marges à droite ou à gauche, de la longueur des pages, des en-têtes de haut et de bas de page, de la double impression (caractères renforcés, du soulignement, etc...

La première moitié de l'écran est réservée aux explications de chacune des commandes. Quatre niveaux d'aide à l'utilisation peuvent être sélectionnés sous la forme d'un menu.

Un programme séparé appelé "INSTALL" permet de configurer WORDSTAR pour un terminal vidéo donné. INSTALL possède une liste de plusieurs dizaines de terminaux avec leurs caractéristiques. Il suffit de sélectionner dans la liste le numéro correspondant au terminal de votre système. S'il n'existe pas, INSTALL offre la possibilité de définir vous-même les caractéristiques du terminal: se rapporter à la notice explicative du produit. Exemple d'appel:

```
A>WS
```

ou

```
A>WS B:RAPPORT.DOC
```

4.6.2 WORDMASTER

WORDMASTER constitue un sous-ensemble de WORDSTAR. Toute la partie d'édition de document y est amputée. WORDMASTER

est donc un éditeur de texte "pleine page" qui comprend toutes les commandes de l'éditeur standard ED de CP/M.

4.6.3. TEXWRITER III

Ce produit est un "formatteur" de texte pour cadrer ou paginer des documents. D'autres documents déjà préparés et stockés sur disque peuvent s'insérer dans le document en cours de constitution (lettres, préimprimés, "mailing", contrats,...).

4.7. PROGICIELS DE GESTION

4.7.1. DATASTAR

DATASTAR de Micropro est un progiciel de saisie, d'interrogation et de mise à jour de fichiers. C'est un produit orienté "écran vidéo" qui permet à l'utilisateur de définir le format de ses grilles de saisie. Celles-ci peuvent avoir plusieurs pages de longueur et jusqu'à trois pages en largeur. Des masques de saisie permettent la protection et le contrôle de zones (taille, caractères numériques ou alphabétiques,...). Un menu d'aide à l'utilisation, représentant l'ensemble des commandes disponibles, est affiché sur l'écran.

4.7.2. SUPERSORT

SUPERSORT est un logiciel de tri, fusion et "formattage" de fichiers. Il permet de trier ou fusionner jusqu'à 32 fichiers en entrée, qu'ils soient sous forme ASCII, BCD ou binaire. Les articles peuvent être de longueur fixe ou variable jusqu'à 2048 caractères. Les clés de tris sont indépendantes, et leur nombre peut aller de 1 à 32.

4.7.3. SUPERCALC

Le logiciel SUPERCALC de Sorcim est similaire au produit VISICALC bien connu sur Apple-2. SUPERCALC est un logiciel orienté vers la gestion financière, la prévision des tendances, l'aide à la décision.

En fait c'est une gigantesque grille de calcul qui utilise un écran vidéo. Les informations sont localisées dans des cellules référencées par une lettre pour les colonnes et un chiffre pour les rangées. Ces cellules peuvent contenir des valeurs numériques, des chaînes de caractères alphanumériques, ou des formules. Chaque modification d'une valeur provoque l'ajustage et le recalcul automatique, là où apparaît cette valeur, dans toutes les cellules du modèle.

SUPERCALC permet de gérer jusqu'à 63 colonnes (A-Z, AA-AZ, BA-BK) et 254 rangées (1-254). La matrice des cellules peut se déplacer horizontalement et verticalement sur l'écran, l'utilisateur n'apercevant qu'une "fenêtre" de la matrice.

4.7.4. DBASE-II

DBASE-II est un système de gestion de base de données (SGBD) interactif, relationnel et non hiérarchique. Il possède son propre langage de manipulation de structures. Des commandes simples permettent la modification, le changement ou l'extension des bases de données.

ANNEXE A

Table des codes ASCII

	0x	1x	2x	3x	4x	5x	6x	7x
x0	NUL	DLE	SP	0	@	P	`	p
x1	SOH	DC1	!	1	A	Q	a	q
x2	STX	DC2	"	2	B	R	b	r
x3	ETX	DC3	#	3	C	S	c	s
x4	EOT	DC4	$	4	D	T	d	t
x5	ENQ	NAK	%	5	E	U	e	u
x6	ACK	SYN	&	6	F	V	f	v
x7	BEL	ETB	'	7	G	W	g	w
x8	BS	CAN	(	8	H	X	h	x
x9	HT	EM	)	9	I	Y	i	y
xA	LF	SUB	*	:	J	Z	j	z
xB	VT	ESC	+	;	K	[	k	{
xC	FF	FS	,	<	L	\	l	|
xD	CR	GS	-	=	M	]	m	}
xE	SO	RS	.	>	N	^	n	~
xF	SI	US	/	?	O	_	o	DEL

NUL	caractère nul = 00
SOH	début d'en-tête de bloc
STX	début de texte
ETX	fin de texte
EOT	fin de transmission
ENQ	invitation (Enquiry)
ACK	accusé de réception positif
BEL	alarme sonore
BS	éffacement caractère
HT	tabulation horizontale
LF	ligne suivante (Line-Feed)
VT	tabulation verticale
FF	saut de page (Form-Feed)
CR	retour chariot
SO	sélection en sortie
SI	sélection en entrée
SP	espace ou blanc

DLE	caractère d'échappement
DC1	asservissement périphérique XON
DC2	télécommande moteur TAPE-ON
DC3	asservissement périphérique XOFF
DC4	télécommande moteur TAPE-OFF
NAK	accusé de réception négatif
SYN	caractère de synchronisation
ETB	fin de transmission de bloc
CAN	annulation (Cancel)
EM	fin de message
SUB	caractère de substitution
ESC	caractère d'échappement
FS	séparateur de fichier
GS	séparateur de groupe
RS	séparateur d'enregistrement
US	séparateur d'unité
DEL	caractère de remplissage

ANNEXE B

Récapitulatif des commandes

nom de la commande	CP/M	MP/M	CP/M-86	MP/M-86
ABORT		X		X
ASM	X	X		
ASM86	X	X	X	X
ATTACH		X		X
CONSOLE		X		X
DDT	X	X		
DDT86			X	X
DIR	X	X	X	X
DSKRESET		X		X
DUMP	X	X	X	X
ED	X	X	X	X
ERA	X	X	X	X
ERAQ	X	X	X	X
GENCMD	X	X	X	X
GENHEX		X		
GENMOD		X		
GENSYS		X		X
LDCOPY			X	
LOAD	X	X		
MOVCPM	X			
MPMLDR	X			
MPMSTAT		X		X
PIP	X	X	X	X
PRINTER				X
PRLCOM		X		
REN	X	X	X	X
SAVE	X		X	
SCHED		X		X
SDIR				X
SET				X
SHOW				X
SPOOL		X		X
STAT	X	X	X	X
STOPSPLR		X		X
SUBMIT	X	X	X	X
SYSGEN	X			
TOD		X		X
TYPE	X	X	X	X
USER	X	X	X	X
XSUB	X	X	X	X

ANNEXE C

Récapitulatif des fonctions système

Numéro fonction	Signification de la fonction CP/M	Paramètre en entrée	Paramètre de retour
C = 0	réinitialisation du système	-	-
C = 1	lecture d'un caractère à la console	-	A=caractère
C = 2	écriture d'un caractère sur la console	E=caractère	-
C = 3	lecture d'un caractère au lecteur	-	A=caractère
C = 4	écriture d'un caractère au perforateur	E=caractère	-
C = 5	écriture d'un caractère à l'imprimante	E=caractère	-
C = 6	lecture directe sur la console	E=FF(hexa)	A=caractère
C = 6	écriture directe sur la console	E=caractère	-
C = 7	lecture de l'octet IOBYTE	-	A=IOBYTE
C = 8	initialisation de l'octet IOBYTE	E=IOBYTE	-
C = 9	impression d'une chaîne ASCII	DE=adresse	-
C = 10	lecture du buffer console	DE=adr.buf	-
C = 11	lecture du "Status" console	-	A=0 ou FFH
C = 12	obtention du numéro de version système	-	HL=A=vers.
C = 13	réinitialisation du disque système	-	-
C = 14	sélection d'un disque	E=No.disque	-
C = 15	ouverture d'un fichier	DE=adr.FCB	A=code dir.
C = 16	fermeture d'un fichier	DE=adr.FCB	A=code dir.
C = 17	recherche du premier fichier ambigu	DE=adr.FCB	A=code dir.
C = 18	recherche du fichier ambigu suivant	-	A=code dir.
C = 19	suppression d'un fichier	DE=adr.FCB	A=code dir.
C = 20	lecture séquentielle d'un enregistrement	DE=adr.FCB	A=code err.
C = 21	écriture séquentielle d'un enregistrement	DE=adr.FCB	A=code err.
C = 22	création d'un fichier	DE=adr.FCB	A=code dir.
C = 23	changement du nom du fichier	DE=adr.FCB	A=code dir.
C = 24	lecture du vecteur de "Login"	-	HL=login
C = 25	lecture du numéro du disque courant	-	HL=disque
C = 26	initialisation de l'adresse buffer DMA	DE=adr.DMA	-
C = 27	retour de l'adresse d'allocation	-	HL=vecteur
C = 28	protection en écriture du disque	-	-
C = 29	lecture du vecteur de protection	-	HL=vecteur
C = 30	modification des attributs du fichier	DE=adr.FCB	A=code dir.
C = 31	adresse des paramètres du disque	-	HL=adr.DPB
C = 32	lecture du numéro de l'utilisateur	E=FF(hexa)	A=numéro
C = 32	forçage du numéro de l'utilisateur	E=numéro	-
C = 33	lecture directe sur disque	DE=adr.FCB	A=code err.
C = 34	écriture directe sur disque	DE=adr.FCB	A=code err.
C = 35	calcul de la taille du fichier	DE=adr.FCB	-
C = 36	positionnement sur un enregistrement	DE=adr.FCB	-

Numéro	Signification de la fonction MP/M	entrée	retour
C= 37	réinitialisation d'un "drive" disque	DE=vecteur	A=0/FFH
C= 38	accessibilité d'un "drive" disque	DE=vecteur	-
C= 39	libération d'un "drive" disque	DE=vecteur	-
C= 40	écriture directe d'un enregistrement nul	DE=adr.FCB	A=code ret.
C=128	demande d'allocation de mémoire absolue	DE=adr.MD	A=0/FFH
C=129	demande d'allocation de mémoire relogeable	DE=adr.MD	A=0/FFH
C=130	libération d'un segment mémoire	DE=adr.MD	
C=131	interrogation d'un périphérique (polling)	E=No.péri.	-
C=132	mise en attente d'un évènement logique	E=No.flag	A=0/FFH
C=133	activation d'un évènement logique	E=No.flag	A=0/FFH
C=134	création d'une queue (QCB)	DE=adr.QCB	-
C=135	ouverture d'une queue (UQCB utilisateur)	DE=ad.UQCB	A=0/FFH
C=136	suppression d'une queue (QCB)	DE=adr.QCB	A=0/FFH
C=137	lecture inconditionnelle d'une queue	DE=ad.UQCB	message lu
C=138	lecture conditionnelle d'une queue	DE=ad.UQCB	A=0/FFH
C=139	écriture inconditionnelle dans une queue	DE=ad.UQCB	-
C=140	écriture conditionnelle dans une queue	DE=ad.UQCB	A=0/FFH
C=141	postage d'un délai dans le temps	DE=Nb.tops	-
C=142	appel du "dispatcher" pour régulation	-	-
C=143	terminaison normale d'un processus	DE=code	-
C=144	création d'un processus (Process Descrip.)	DE=adr.PD	PD créé
C=145	modification de la priorité d'un processus	DE=prior.	-
C=146	attachement d'un processus à la console	-	-
C=147	détachement du processus de la console	-	-
C=148	détachement et attachement à la console	-	-
C=149	assignation de la console à un processus	DE=adr.APB	A=0/FFH
C=150	envoi d'une commande à l'interpréteur CLI	DE=CLICMD	-
C=151	appel d'un processus résident de type RSP	DE=adr.CPB	HL=0/FFH
C=152	analyse syntaxique du nom de fichier	DE=ad.PFCB	-
C=153	obtention du numéro de console		A=numéro
C=154	adresse de la zone des paramètres système	-	HL=adresse
C=155	obtention de la date et de l'heure	DE=adr.TOD	-
C=156	adresse du descripteur de processus courant	-	HL=adr.PD
C=157	avortement d'un processus	DE=adr.APB	A=code

Numéro	Signification de la fonction CP/NET	entrée	retour
C= 64	demande de connexion à un système maître	DE=message	A=0/FFH
C= 65	déconnexion d'un utilisateur du réseau	E=maître	A=0/FFH
C= 66	envoi d'un message sur le réseau	DE=adr.mess	-
C= 67	réception d'un message du réseau	-	DE=adr.buf
C= 68	état du réseau	E=syst.ID	A=état
C= 69	adresse de la table de configuration	-	HL=adresse

ANNEXE D

Liste des instructions machine du 8080

MOV r1,r2	transfert registre à registre
MOV M,r	transfert registre à mémoire
MOV r,M	transfert mémoire à registre
HLT	halte du CPU
MVI r,v	chargement immédiat registre
MVI M,v	chargement immédiat mémoire
INR r	incrémentation registre
DCR r	décrémentation registre
INR M	incrémentation mémoire
DCR M	décrémentation mémoire
ADD r	addition registre à A
ADC r	addition dans A avec carry
SUB r	soustraction registre de A
SBB r	soustraction avec retenue
ANA r	intersection registre et A
XRA r	OU exclusif registre et A
ORA r	réunion registre avec A
CMP r	comparaison registre avec A
ADD M	addition mémoire avec A
ADC M	addition mémoire avec carry
SUB M	soustraction mémoire de A
SBB M	soustraction avec retenue
ANA M	ET mémoire avec A
XRA M	OU exclusif mémoire avec A
ORA M	OU mémoire avec A
CPM M	comparaison mémoire avec A
ADI v	addition immédiate à A
ACI v	addition immédiate à avec CY
SUI v	soustraction immédiate à A
SBI v	soustraction avec retenue
ANI v	ET immédiat avec A
XRI v	OU exclusif immédiat avec A
ORI v	OU immédiat avec A
CPI v	comparaison immédiate avec A
RLC n	rotation à gauche de A
RRC n	rotation à droite de A
RAL n	rotation gauche à travers CY
RAR n	rotation droite à travers CY
JMP adr	branchement inconditionnel
JC adr	branchement si carry vrai
JNC adr	branchement si carry faux
JZ adr	branchement si résultat nul
JNZ adr	branchement si non nul
JP adr	branchement si positif
JM adr	branchement si négatif
JPE adr	branchement si parité paire
JPO adr	branchement si parité impaire
CALL adr	appel de sous-programme
CC adr	appel si carry vrai
CNC adr	appel si carry faux
CZ adr	appel si résultat nul
CNZ adr	appel si résultat non nul
CP adr	appel si résultat positif
CM adr	appel si résultat négatif
CPE adr	appel si parité paire
CPO adr	appel si parité impaire

RET	retour de sous-programme
RC	retour si carry
RNC	retour si carry faux
RZ	retour si zéro
RNZ	retour si non nul
RP	retour si positif
RM	retour si négatif
RPE	retour si parité paire
RPO	retour si parité impaire
RST	retour sous interruption
IN	lecture périphérique
OUT	écriture périphérique
LXI B,v	chargement immédiat de BC
LXI D,v	chargement de DE
LXI H,v	chargement de HL
LXI SP,v	chargement de SP
PUSH B	empilement de BC
PUSH D	empilement de DE
PUSH H	empilement de HL
PUSH PSW	empilement de A et PSW
POP B	dépilement de BC
POP D	dépilement de DE
POP H	dépilement de HL
POP PSW	dépilement de A et PSW
STA adr	rangement direct de A
LDA adr	chargement direct de A
XCHG	échange de DE et de HL
XTHL	échange de SP et de HL
SPHL	force HL dans SP
PCHL	force le compteur ordinal
DAD B	addition de BC à HL
DAD D	addition de DE à HL
DAD H	addition de HL à HL
DAD SP	addition de SP à HL
STAX B	rangement indirect de A
STAX D	rangement indirect de A
LDAX B	chargement indirect de A
LDAX D	chargement indirect de A
INX B	incrémentation de BC
INX D	incrémentation de DE
INX H	incrémentation de HL
INX SP	incrémentation de SP
DCX B	décrémentation de BC
DCX D	décrémentation de DE
DCX H	décrémentation de HL
DCX SP	décrémentation de SP
CMA	complémentation de A
STC	forçage du carry
CMC	complémentation du carry
DAA	addition décimale
SHLD adr	rangement direct de HL
LHLD adr	chargement direct de HL
EI	autorise les interruptions
DI	inhibe les interruprtions
NOP	pas d'opération

ANNEXE E

Liste des principaux types de fichiers

```
COM : fichier exécutable dans la zone TPA (CP/M et MP/M)
ASM : fichier source en assembleur 8080 (CP/M et MP/M)
HEX : fichier objet au format Intel 8080 en hexadécimal
PRN : fichier image imprimante
BAK : fichier image de la dernière version traitée par ED
$$$ : fichier temporaire de travail
BAS : fichier chargeable par l'interpréteur MBASIC
COB : fichier source écrit en langage COBOL
FOR : fichier source écrit en langage FORTRAN
PAS : fichier source écrit en langage PASCAL
PCO : fichier intermédiaire en P-code PASCAL
INT : fichier en code intermédiaire
ASC : fichier source écrit en langage BASIC
DAT : fichier de données
DOC : fichier documentation
LIB : fichier bibliothèque
MAC : fichier source pour le macro-assembleur MAC-80
SUB : fichier catalogue de commandes à soumettre
PRL : fichier relogeable exécutable dans un segment mémoire sous MP/M
REL : fichier objet en code intermédiaire relogeable par LINK-80
TXT : fichier texte
RSP : fichier programme système pouvant être résident en mémoire (MP/M)
SPR : fichier module du système MP/M
SYS : fichier image disque du système (MP/M,CP/M-86,MP/M-86)
A86 : fichier source en assembleur 8086
H86 : fichier objet au format Intel 8086 en hexadécimal
LST : fichier image imprimante d'un assemblage sous ASM86
CMD : fichier exécutable sous CP/M-86 et MP/M-86
```

ANNEXE F

Liste des principaux produits sous CP/M

Langages de programmation

Langage	Nom du Produit	Description du produit	Concepteur
MACRO	MAC	macro-assembleur 8080	Digital-Research
	MACRO-80	macro-assembleur 8080/Z80	Microsoft
BASIC	MBASIC	interpréteur	Microsoft
	CBASIC	interpréteur	Compiler Systems
	KBASIC	interpréteur	Eidos
	XY BASIC	interpréteur	M. Williams Co
	BASCOM	compilateur	Microsoft
	SBASIC	compilateur structuré	Topaz Programming
PASCAL	PASCAL/M	interpréteur (P-code)	Sorcim
	PASCAL/MT+	compilateur (norme ISO)	MT Microsystems
	PASCAL/Z	compilateur pour Z80	Ithaca Intersystems
	JRT PASCAL	interpréteur temps réel	
FORTRAN	FORTRAN-80	compilateur (ANSI-66)	Microsoft
	SSS FORTRAN	compilateur	Supersoft
COBOL	COBOL-80	compilateur (ANSI-74)	Microsoft
	CIS-COBOL	interpréteur (ANSI-74)	Micro-Focus
	NEVADA-COBOL	compilateur (ANSI-74)	Ellis
C	C COMPILER	compilateur	Whitesmiths
	BDS-C	compilateur	BDS
	C	compilateur	Supersoft
	Tiny C	interpréteur	Tiny C Associated
	C86	compilateur pour 8086	Computer Innovation
PL1	PL/1-80	compilateur sous ensemble	Digital Research
PL/M	PL/M	compilateur	Digital Research
LISP	mu-LISP	interpréteur	Microsoft
	LISP	interpréteur	Supersoft
APL	uAPL	interpréteur	Softronics
ALGOL	ALGOL-60	compilateur	Resrch
ADA	ADA	compilateur sous-ensemble	Supersoft
FORTH	FORTH	interpréteur	Supersoft
RATFOR	RATFOR	traducteur en FORTRAN	Supersoft

Outils de développement

Nom du produit	Description du produit	Concepteur
SID, ZSID	"debuggers" symboliques 8080 et Z80	Digital Research
EDIT-80	éditeur de texte mode ligne	Microsoft
VEDIT	éditeur de texte mode page	CompuView Products
IBMCPM	conversion de disquettes IBM-3740	Lifeboat Ass.
TRANS86	traducteur de binaires 8080/8086	Sorcim
DISTEL, DISILOG	désassembleurs 8080 et Z80	Lifeboat Ass.

Traitement de texte

Nom du produit	Description du produit	Concepteur
WORDSTAR	éditeur de texte et de documents	Micropro
TEX	éditeur de texte mode page	Digital Research
TEXWRITER III	éditeur de texte et de documents	Organic Software
WORDMASTER	éditeur de texte mode page	Micropro
SPELLBINDER	éditeur de document, photocomposition	Lexisoft
SPELLSTAR	vérificateur de documents	Micropro
SPELLGUARD	vérificateur de documents	Innovative Software

Progiciels de gestion

Nom du produit	Description du produit	Concepteur
DATASTAR	saisie et interrogation de fichiers	Micropo
CALCSTAR	gestion financière et prévisions	Micropo
SUPERCALC	gestion financière et prévisions	Sorcim
DBASE II	système de base de données	Ashton-Tate
MAIL-MERGE	système de "mailing"	Microsoft

ANNEXE G

Liste de quelques micro-ordinateurs

CP/M	CP/M et MP/M	CP/M-86 et MP/M-86
Apple II + Softcard Z80 Cromenco System 3 Datapoint 1550/2150 Dec VT18X Rainbow Heath H8/H89 Hewlett-Packard HP125 IBM personnal computer IF-800 ITT 3030 JB 3000 Panasonic Logabax LX500 Micropolis mod I/II Mostek MDX Ohio Scientific C3 Osborne 1 North star Horizon Pertec PCC-1000/2000 Radio Shack TRS80 I/II Sanco 2000/7000 Stratos TKL-10, TKL-20 Xerox 820 microcomputer Zenith Z89	Addx systèmes SM1/SM2 Altos ACS8000/TKL8000 Dynabyte DB 8/4 Intel MDS/800 Micromation Onyx C8001 REE Quasar System 2800 Zobex Z-Plus	IF86 Infor Française Altos ACS8600 Micromachine 4000 NSC 6604/6608 Sirius 1 GPS Système 101

Imprimerie de la Manutention à Mayenne
Dépôt légal : septembre 1982
N° d'éditeur : 3812

Ce livre s'adresse à tous ceux qui veulent comprendre et pratiquer CP/M, MP/M et leurs extensions.

CP/M est devenu un véritable "standard" pour les micro-ordinateurs.

Ce succès, CP/M le doit à lui-même, et aussi à la grande variété de logiciels qui ont été développés autour de lui.

Vous trouverez dans cet ouvrage non seulement la description complète du fonctionnement et des commandes de CP/M et MP/M, illustrée de nombreux exemples, mais aussi un guide des extensions actuelles nées de CP/M, de MP/M et de l'avènement des microprocesseurs 16 bits : CP/M-86, MP/M-86, CP/NET, MP/NET...

Enfin, un chapitre particulier est consacré aux produits développés autour de CP/M et MP/M : Langages de programmation, traitement de texte, progiciels de gestion.

EYROLLES

8617

EYROLLES

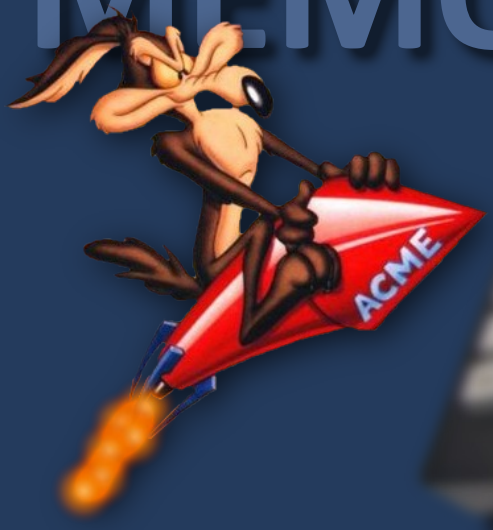
CP/M ET SA FAMILLE

Philippe DAX

Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/