ABC INFORMATIQUE

N° 34

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

- Les systèmes d'exploitation
- Gestion de fichiers sur disquette
- Les « XL » d'Atari
- Sautez en parachute

EDITIONS ATLAS

M6062-34-12F 85FB-3,80FS-$1.95

aBC INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 35-40-23. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS. Tél. : (37) 35-40-23.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume III, n° 34.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Jean Riby/Matra-Hachette.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : septembre 1984. 7849. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Opérateurs silencieux

Le système d'exploitation est une partie vitale de l'ordinateur : il assure la liaison entre le logiciel et le matériel. Pour comprendre son importance, exposons son fonctionnement.

Les langages évolués permettent d'ignorer les mécanismes de l'unité centrale, isolant le programmeur de la machine. Ils contribuent à une plus grande portabilité des programmes d'un ordinateur à l'autre. Si un langage est suffisamment standard, les instructions fonctionneront sur toutes les machines l'implémentant. L'interpréteur ou le compilateur qui traite le code source du langage évolué se charge des détails d'allocation mémoire ou autres. L'interpréteur ou le compilateur évolué est également un programme; il doit être chargé en mémoire centrale avant de pouvoir transcrire le code source évolué en instructions code objet exécutables. Le BASIC résidant en ROM est prêt à être utilisé dès l'initialisation de la machine.

Cependant, l'exploitation d'un ordinateur ne se résume pas à un programme central capable de convertir le code source en code machine. Un programme de fond doit assurer la gestion proprement dite de l'ordinateur. Il s'agit de son intendance, comme lorsqu'on frappe une touche au clavier et que le caractère correspondant doit apparaître à l'écran. Alors, quelque part en mémoire existe un programme approprié qui dit à l'UC de prendre en compte toute frappe au clavier. Le programme doit en outre reconnaître le caractère tapé et en informer le circuit de l'unité de visualisation. Cette dernière pourra ainsi créer la matrice de points correspondante selon la séquence d'origine entrée au clavier. Le résultat sera l'affichage de la frappe. Cette gestion est dite « transparente à l'utilisateur ».

De manière semblable, lorsqu'une commande CLOAD est utilisée pour sauvegarder un programme sur cassette, le programmeur n'a pas à se soucier de la manière dont les données sont converties sous une forme appropriée à l'enregistrement sur cassette. Cela incombe au système d'exploitation.

Le système d'exploitation constitue l'ensemble des programmes de fond qui effectuent continuellement les tâches internes à l'ordinateur. Une légère confusion se fait jour lorsque l'ordinateur dispose d'un petit système d'exploitation résidant en ROM avec BASIC résident. En effet, le BASIC et le système d'exploitation figurent alors tous les deux sur le même composant. Sous sa forme la plus simple, la ROM interne comportera donc toute la partie logicielle nécessaire au fonctionnement de l'ordinateur. Elle ne contiendra pas, bien sûr, les programmes d'application (jeux, traitements de texte, etc.), chargés ou écrits par l'utilisateur. Une partie de la ROM (l'*interpréteur*) comportera le code nécessaire pour traduire en code machine les programmes d'application écrits en BASIC. Il y aura également l'*éditeur,* servant à saisir et à modifier les programmes écrits par l'utilisateur. Le reste constituera le *moniteur,* chargé du logiciel d'intendance : gestion du clavier, affichage des caractères et affichage graphique, réception de données depuis une cassette et leur allocation mémoire, etc.

Il ne faut pas confondre le terme « moniteur » employé ici avec un « moniteur de visuali-

Le gestionnaire

Tout ordinateur comporte un système d'exploitation, aussi rudimentaire soit-il. Il s'agit d'un programme qui gère l'exploitation de l'ordinateur et commande les périphériques du système. (Cl. SMT Goupil.)

sation ». En fait, le moniteur à lui seul est pratiquement synonyme de système d'exploitation. Sous sa forme la plus simple, il se contente de recevoir des instructions en code machine, de leur allouer l'adresse mémoire appropriée et de superviser leur exécution. Lorsque la gestion du moniteur prend de l'importance, on le désigne alors par système d'exploitation.

A l'autre extrémité, il y a les ordinateurs dont l'exploitation repose sur un disque. Ils sont souvent utilisés dans des bureaux comme petits systèmes de gestion et disposent de systèmes d'exploitation puissants. Avant d'aborder les systèmes intermédiaires tels que les micros Apple, étudions le système à base de disque.

Un système d'exploitation résidant uniquement sur disque souple pour ce qui est de sa partie logiciel comportera habituellement peu de programmes en ROM, mis à part le chargeur d'amorce de programme et quelques routines d'intendance. Lors de l'initialisation d'un ordinateur de ce type, le chargeur d'amorce informera l'UC de l'accès au lecteur, et chargera en mémoire centrale le système d'exploitation.

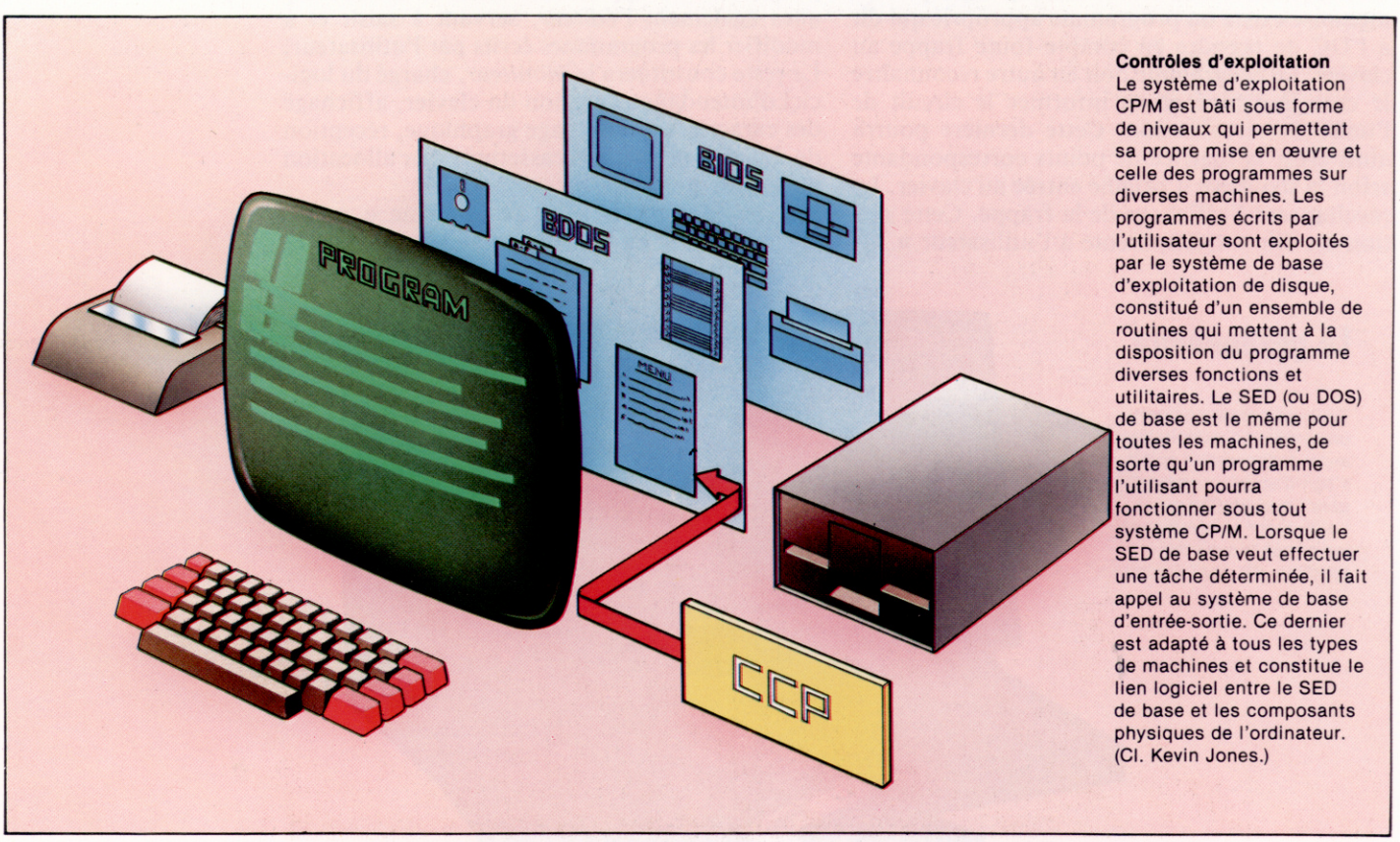
Un système d'exploitation résidant en RAM se doit d'être plus performant qu'un système en ROM. Il devient ce que l'on appelle un système d'exploitation de disque. Ce dernier représente une extension par rapport au système d'exploitation traditionnel; il ajoute aux fonctions générales d'intendance des commandes agissant directement sur des fichiers disque. Ces commandes comprennent le listage des noms de fichiers, l'effacement d'un fichier, le changement d'appellation d'un fichier, la copie de fichiers depuis le disque sur la mémoire centrale ou sur d'autres disques.

Les systèmes d'exploitation plus sommaires qui résident en ROM ne comportent généralement pas de telles commandes de maniement de fichiers. En fait de gestion de fichiers, ils n'ont rien d'autre qu'une simple commande de chargement d'un fichier répertorié depuis une cassette, ou de sauvegarde sur cassette d'un fichier connu. Le SED (système d'exploitation de disque), pour sa part, connaîtra l'emplacement exact de l'adresse voulue sur le disque ou sur la cassette, pour chaque fichier. Les systèmes moins sophistiqués devront quant à eux passer en revue tous les fichiers présents sur le support (cassette ou disque), jusqu'à ce qu'ils trouvent le fichier demandé et le chargent. Ces systèmes rudimentaires devront, pour sauvegarder un fichier, l'écrire sur la cassette à l'emplacement se présentant lors de l'appel de la commande.

Le BBC micro illustre bien les systèmes intermédiaires, avec ses commandes de gestion de fichiers sur cassette ou sur disque en grande partie égales aux systèmes les plus perfectionnés. Le système d'exploitation réside sur un composant ROM, physiquement distinct, donc, de la ROM. On peut le considérer comme un système d'exploitation chargé en mémoire par l'intermédiaire d'un périphérique externe à la mémoire.

Ces systèmes intermédiaires comprennent des commandes de gestion de fichiers nettement plus évoluées que les systèmes résidant uniquement en ROM tels que le Spectrum.

En résumé, le système d'exploitation d'un ordinateur peut être considéré comme un pro-



Contrôles d'exploitation
Le système d'exploitation CP/M est bâti sous forme de niveaux qui permettent sa propre mise en œuvre et celle des programmes sur diverses machines. Les programmes écrits par l'utilisateur sont exploités par le système de base d'exploitation de disque, constitué d'un ensemble de routines qui mettent à la disposition du programme diverses fonctions et utilitaires. Le SED (ou DOS) de base est le même pour toutes les machines, de sorte qu'un programme l'utilisant pourra fonctionner sous tout système CP/M. Lorsque le SED de base veut effectuer une tâche déterminée, il fait appel au système de base d'entrée-sortie. Ce dernier est adapté à tous les types de machines et constitue le lien logiciel entre le SED de base et les composants physiques de l'ordinateur. (Cl. Kevin Jones.)

gramme intermédiaire entre l'utilisateur et la machine dont l'UC, le logiciel système (comme les langages de programmation) et le logiciel d'application.

Portabilité

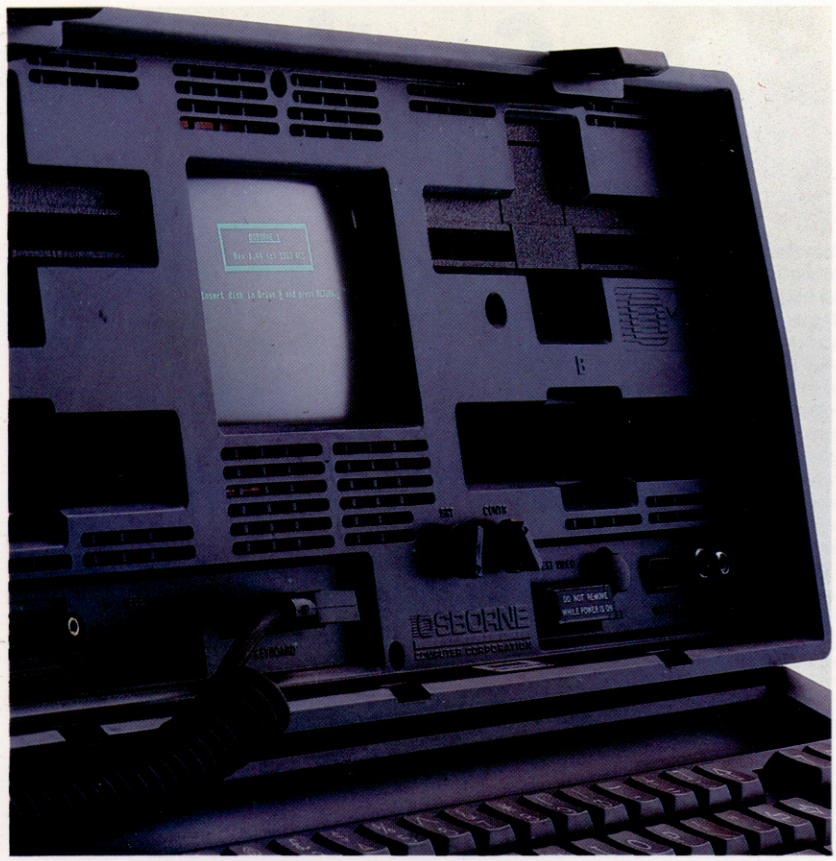
La possibilité d'utiliser un logiciel sur plusieurs machines s'appelle « portabilité ». Ce problème recouvre deux aspects : le premier est que des processeurs différents nécessitent des jeux d'instructions différents pour effectuer des opérations semblables. Ainsi des instructions code machine, destinées à additionner le contenu de deux positions mémoire, prendront des formes différentes selon qu'elles seront écrites pour le microprocesseur 6502 (utilisé par Apple), ou pour le Z-80 (sur le Spectrum). Le problème de la conversion de code évolué en un code machine approprié relève cependant de l'interpréteur ou du compilateur utilisé. Aussi à chaque UC devront correspondre un interpréteur et un compilateur spécifiques.

La portabilité d'un logiciel concerne également des aspects indépendants de l'UC : ainsi, même avec une UC identique d'un système à l'autre, comme dans le cas du BBC et de l'Apple, des complications d'un autre ordre surgissent : la mémoire de l'unité de visualisation utilise des emplacements mémoire différents ; le déplacement du curseur utilise aussi d'autres codes ; les périphériques d'entrée-sortie ne sont pas les mêmes, etc.

Pour pallier ce problème, des systèmes d'exploitation génériques ont été conçus pour permettre, par exemple, aux logiciels écrits pour un ordinateur dont le système d'exploitation Z-80 réside sur disquette, d'être portables sur d'autres ordinateurs utilisant également le Z-80. Le plus connu de ces SED est le CP/M (Control Program/Microcomputers).

Les SED tels que le CP/M sont essentiellement un développement des moniteurs et des systèmes d'exploitation plus spécifiques à la machine. Ils représentent un progrès considérable en termes de portabilité logicielle. De la sorte, tout programme écrit pour fonctionner sous un système d'exploitation générique tel que le CP/M ou MS-DOS fonctionnera sur tout ordinateur utilisant le même système. La seule restriction sera que les programmes ne devront pas utiliser les caractéristiques de développement propres à la machine d'origine (les effets sonores, par exemple). Le logiciel du SED est livré sous une forme standard par ses auteurs. Les fabricants d'ordinateurs doivent, quant à eux, réécrire une partie du programme qui fait intervenir la machine d'implémentation (partie matérielle).

Les SED varient considérablement en complexité et en performances, mais les plus simples, tels que le CP/M et le MS-DOS, comprennent essentiellement trois parties : le processeur de commandes, le SED de base et le système de base d'entrée-sortie. Seul le SED de base influe sur la portabilité des logiciels. Il s'agit en effet d'une partie distincte du programme qui comporte les routines de gestion de tous les périphériques, y compris de l'écran et du clavier. Il doit donc être configuré spécialement pour chaque ordinateur. Tout programme utilisant le même SED utilisera le système de base d'exploitation de disque comme interface avec l'ordinateur. Ce dernier se chargera donc de reconnaître la frappe au clavier, d'émettre des caractères vers l'écran ou vers une imprimante, d'adresser des disques, etc.



La clef du succès
L'Osborne 1 doit beaucoup de son succès à son système d'exploitation, le CP/M. Il ouvre l'accès à tous les programmes développés sous CP/M, dont WordStar (traitement de texte), SuperCalc (tableur), et MBASIC (forme de BASIC).
(Cl. Ian McKinnell.)

Le système de base d'entrée-sortie concerne les éléments qui ne dépendent pas d'un périphérique dans le système d'exploitation. Aussi, cette partie du programme n'a pas besoin d'être modifiée d'un ordinateur à l'autre. Le SED de base et le système de base entrée-sortie correspondent à peu près au moniteur ou au système d'exploitation résidant en ROM.

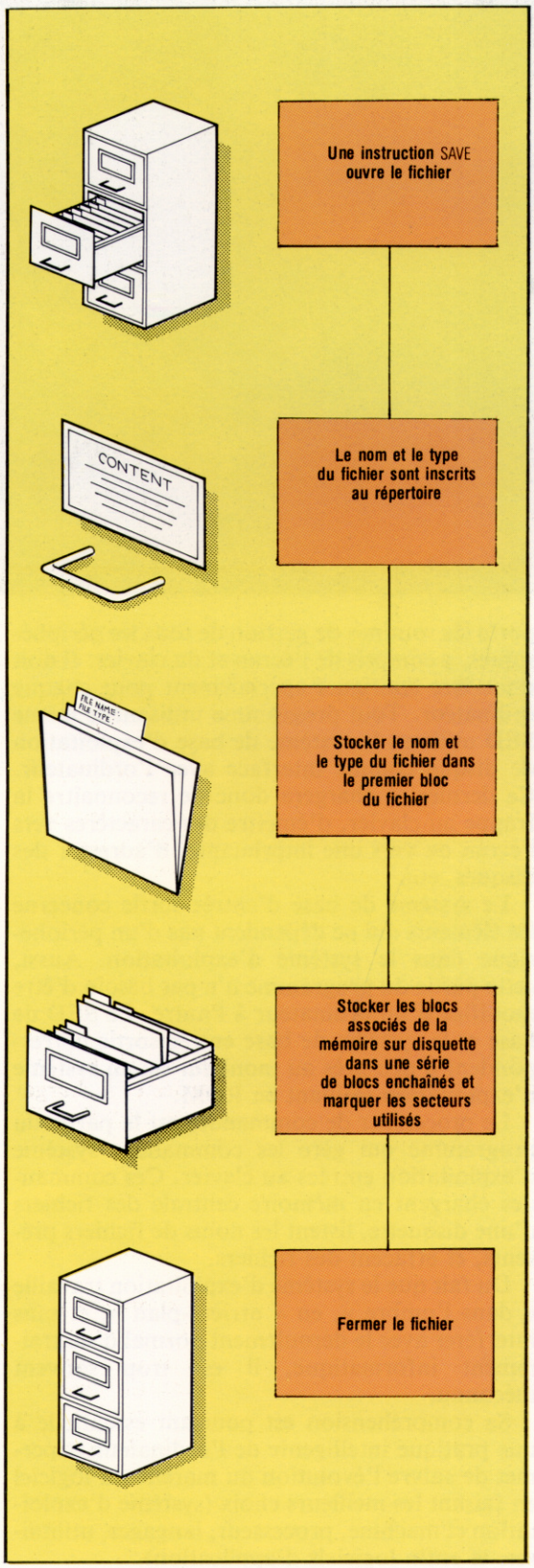
Le processeur de commandes est la partie du programme qui gère les commandes système d'exploitation entrées au clavier. Ces commandes chargent en mémoire centrale des fichiers d'une disquette, listent les noms de fichiers présents, et effacent des fichiers.

Du fait que le système d'exploitation travaille « dans l'ombre », en « arrière-plan », et sans interférer avec le déroulement normal d'un traitement informatique, il est trop souvent méconnu.

Sa compréhension est pourtant essentielle à une pratique intelligente de l'ordinateur et permet de suivre l'évolution du marché du logiciel en faisant les meilleurs choix (système d'exploitation et machine, processeur, langages, utilitaires, et enfin logiciels d'application).

Gestion de mémoire

Sauvegarde d'un fichier binaire
Un fichier binaire est simplement une copie d'une portion de la mémoire. Il peut s'agir d'un programme en mémoire, d'une image écran. Le processus de classement est très simple; après qu'une entrée pour le fichier a été faite dans le répertoire, les données sont écrites sous la forme d'une série de secteurs enchaînés. Le DOS conserve également une liste des secteurs utilisés afin qu'il ne soient pas écrasés lors de la création d'un autre fichier.



Nous commençons à examiner ici les diverses méthodes de gestion de fichiers utilisées par les systèmes de stockage sur disquette : fichiers binaires, séquentiels ou à accès direct. Nous avons déjà examiné les avantages des systèmes à disque par rapport aux bandes.

Afin de « gérer » efficacement nos activités quotidiennes, il est nécessaire d'enregistrer de façon précise nos diverses interventions, transactions monétaires, rendez-vous et autres. La plupart des gens utilisent un agenda et tiennent leur comptabilité bancaire en remplissant les talons de leur chéquier. Ce besoin de stocker des informations et de se les rappeler est amplifié lorsqu'on doit en manipuler une quantité importante, et qui change constamment. Dans ce cas, la gestion de l'information devient de plus en plus complexe. La plupart des problèmes qui surgissent alors peuvent être attribués à une mauvaise gestion et à une mauvaise interprétation de l'information. Une méthode simple et efficace de stockage, d'indexation et d'extraction de l'information est donc la base d'une bonne gestion. Les administrateurs compétents connaissent l'importance du choix des techniques de classement. Ils définissent leurs méthodes de stockage en fonction du type, du volume et du taux de modification de l'information dont ils sont responsables.

Ces principes demeurent inchangés lorsqu'ils sont appliqués à des systèmes informatiques devant manipuler et stocker de façon précise de vastes quantités d'informations à une vitesse incroyablement élevée. Au centre d'un tel système, on retrouve l'« administrateur » de l'ordinateur — le système d'exploitation de disquettes (DOS) — qui fonctionnera parfaitement s'il a reçu les instructions appropriées et si le « gérant » (c'est-à-dire votre programme) lui pose les bonnes questions. Ainsi, l'efficacité des systèmes de stockage informatique sera tributaire de la qualité de la structure des données (ou du système de classement) adoptée par le DOS et de la manière dont le DOS est utilisé.

Les méthodes standard de gestion de fichiers utilisés dans les systèmes micro-informatiques furent initialement développées pour des ordinateurs gros ou mini. Elles peuvent être divisées en trois systèmes : les fichiers binaires, séquentiels et à accès direct. Nous examinerons chacune de ces méthodes séparément.

Répertoire de disquette

L'écran affiche un répertoire de disquette produit par l'utilitaire STAT CP/M qui donne beaucoup plus d'informations que la plupart des répertoires.

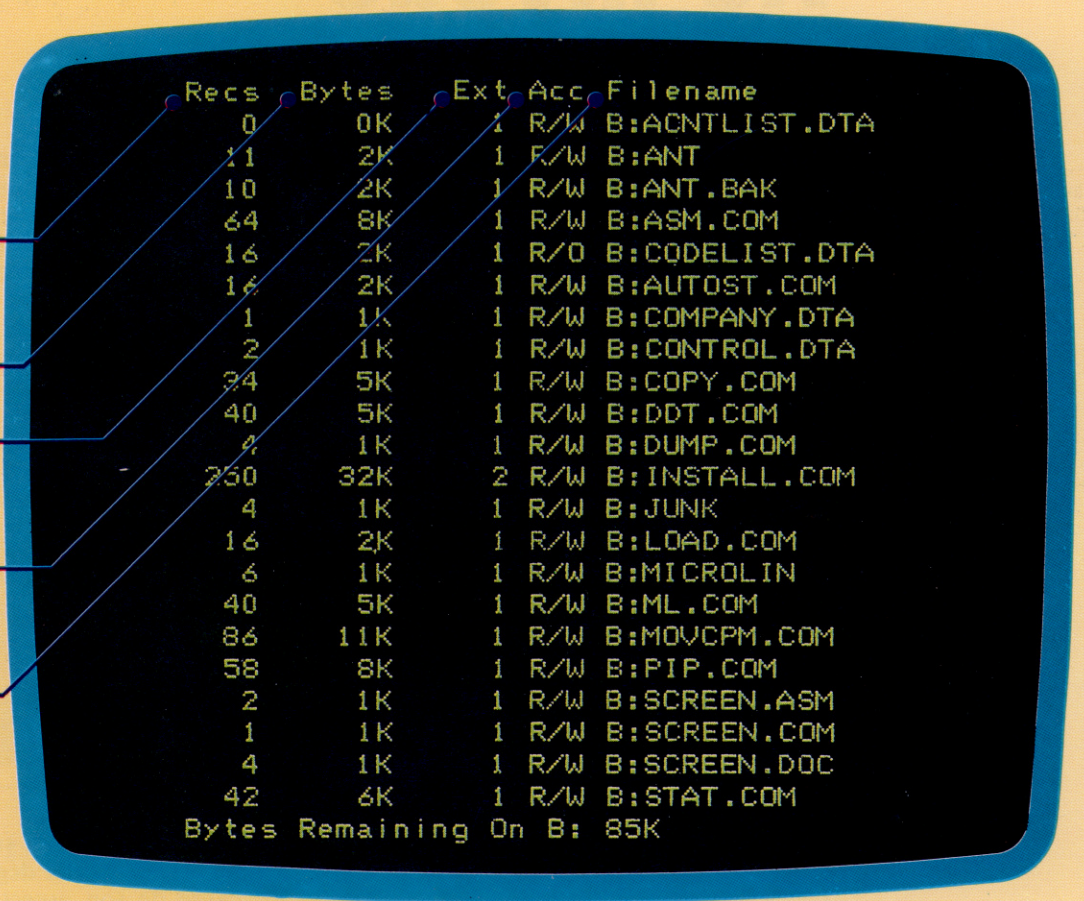
Recs
Le nombre d'enregistrements dans le fichier peut varier; ici les enregistrements ont une longueur de 128 octets.

Bytes
La longueur du fichier en octets.

Ext
Une autre unité de mesure pour indiquer l'espace occupé par le fichier sur la disquette.

Acc
Un fichier peut être inscrit pour la lecture et pour l'écriture (R/W), ou uniquement pour la lecture (R/O).

File Name
Le nom complet du fichier commence par un nom de lecteur (A: ou B:), suivi du nom du fichier (AUTOST, par exemple), suivi par l'extension du nom de fichier (.COM, par exemple), qui peut donner une certaine information au sujet du contenu du fichier.



```
Recs   Bytes   Ext  Acc  Filename
   0      0K     1  R/W  B:ACNTLIST.DTA
  11      2K     1  R/W  B:ANT
  10      2K     1  R/W  B:ANT.BAK
  64      8K     1  R/W  B:ASM.COM
  16      2K     1  R/O  B:CODELIST.DTA
  16      2K     1  R/W  B:AUTOST.COM
   1      1K     1  R/W  B:COMPANY.DTA
   2      1K     1  R/W  B:CONTROL.DTA
  24      5K     1  R/W  B:COPY.COM
  40      5K     1  R/W  B:DDT.COM
   4      1K     1  R/W  B:DUMP.COM
 250     32K     2  R/W  B:INSTALL.COM
   4      1K     1  R/W  B:JUNK
  16      2K     1  R/W  B:LOAD.COM
   6      1K     1  R/W  B:MICROLIN
  40      5K     1  R/W  B:ML.COM
  86     11K     1  R/W  B:MOVCPM.COM
  58      8K     1  R/W  B:PIP.COM
   2      1K     1  R/W  B:SCREEN.ASM
   1      1K     1  R/W  B:SCREEN.COM
   4      1K     1  R/W  B:SCREEN.DOC
  42      6K     1  R/W  B:STAT.COM
Bytes Remaining On B: 85K
```

Fichiers binaires

Un fichier binaire est une simple copie d'une portion de la mémoire utilisateur; un programme sauvegardé est un bon exemple. Comparez la zone de la RAM utilisateur à un bloc-notes. Si vous désirez conserver des notes ou des dessins importants, vous devez arracher les pages concernées et les ranger dans un endroit pratique. Les fichiers binaires fonctionnent de la même manière. Lorsqu'une commande SAVE est utilisée, le DOS stocke le NOM DE FICHIER sur la disquette, l'identifiant d'une manière ou d'une autre comme étant un fichier binaire et il copie alors octet par octet sur la disquette la zone associée de la mémoire. Le programme est stocké dans des blocs enchaînés (avec des marqueurs à la fin de chaque bloc qui indiquent où se trouve le début du prochain bloc) jusqu'à la fin du programme ou des données. Le dernier bloc se termine par un marqueur de fin de fichier. En utilisant notre analogie, nous pourrions dire que nous avons nommé et stocké une page de notre bloc-notes dans un tiroir d'armoire de classement, et ajouté le nom du fichier à la liste du contenu du tiroir.

Lorsque la touche RETURN ou ENTER est pressée, une ligne de programme BASIC est traduite sous une forme compressée dans laquelle les mots clés BASIC sont codés par l'interpréteur BASIC en nombres d'un octet. Ceux-ci peuvent être manipulés plus facilement et décodés ultérieurement pour redonner un listage intelligible. Comme un fichier binaire est une image de la RAM, il peut également stocker des codes ASCII et des données binaires. Cette simplicité de stockage des fichiers ASCII est utile pour stocker le contenu de la mémoire écran; par exemple, les affichages écran peuvent être sauvegardés et rechargés facilement dans la même zone de mémoire. De plus, certains systèmes d'exploitation de disquettes et certains BASIC permettent de stocker un programme BASIC sous une forme ASCII. Cela permet d'éditer le programme non compressé comme un fichier texte avec des éditeurs sophistiqués.

Les fichiers binaires sont très simples à utiliser et à gérer, mais ils sont limités par deux facteurs. D'abord, il n'est possible de sauvegarder l'information associée que dans une section continue de données. Cela a pour conséquences que l'information doit être extraite de la même manière et que les fichiers binaires doivent être chargés en mémoire dans leur intégralité. La taille maximale d'un fichier est limitée par la quantité de mémoire RAM disponible.

Au niveau

Nous étudions ici l'ensemble du processus d'élaboration d'un circuit, depuis ses spécifications initiales jusqu'à son diagramme final.

Avant d'aborder la conception de ces deux applications évoluées, nous étudierons une autre porte logique importante : OU eXclusif, XOU. Nous en avons déjà donné la définition sans en avoir donné la représentation symbolique en algèbre de Boole ou en tant que diagramme de circuit :

Table de vérité			Représentation du circuit
A	B	C	
0	0	0	A, B → XOU → C
0	1	1	
1	0	1	Symbole booléen : ⊕
1	1	0	

Nous constatons à partir de la table de vérité que le résultat C s'exprime de deux manières :

a) $C = A \oplus B = \overline{A} \cdot B \oplus A \cdot \overline{B}$

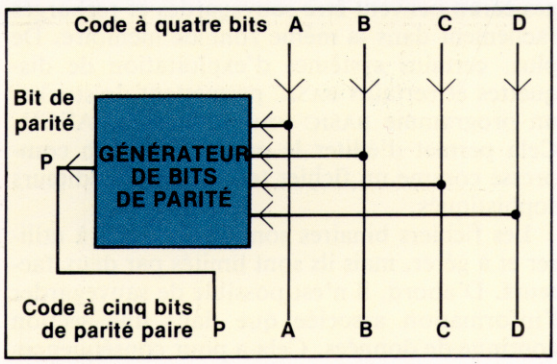
b) $\overline{C} = \overline{A \oplus B} = \overline{A} \cdot \overline{B} \oplus A \cdot B$

La deuxième expression est obtenue en prenant en considération les cas où C n'est pas égal à 1 (soit zéro). Cette porte nous sera particulièrement utile pour notre première application.

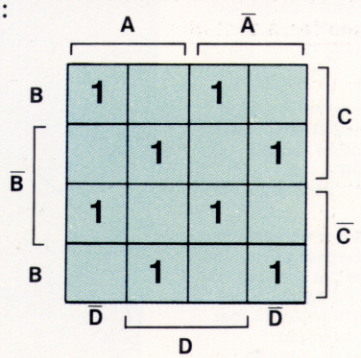
A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table de vérité du GBP

Générateur de bits de parité



La parité est un concept important pour la création de systèmes de transmission de données. Le bit de parité (voir le diagramme précédent) d'un code binaire est ajouté au reste du code afin d'obtenir un nombre pair de uns pour tous les codes transmis. Une autre convention veut qu'il y ait un nombre impair de uns. Le bit de parité agit comme un système de vérification de la bonne transmission des données. Le circuit que nous allons créer reçoit un code à quatre bits et génère un bit de parité approprié. Avec quelques modifications, le circuit peut également tenir lieu de contrôle de parité pour les données à venir. La table de vérité pour ce circuit figure dans la marge. Les valeurs de celle-ci peuvent être représentées sur un tableau de Karnaugh :



Le schéma obtenu sur le tableau de Karnaugh ne permet pas de dégager des simplifications, à cause de son aspect symétrique. L'expression résultante pour P (parité) devient :

$$P = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot C \cdot D + A \cdot B \cdot \overline{C} \cdot D + A \cdot B \cdot C \cdot \overline{D}$$

En regroupant respectivement les termes en rouge et les termes en bleu, l'expression peut se simplifier :

$$P = (\overline{A} \cdot \overline{B} + A \cdot B) \cdot (\overline{C} \cdot D + C \cdot \overline{D}) + (A \cdot \overline{B} + \overline{A} \cdot B) \cdot (\overline{C} \cdot \overline{D} + C \cdot D)$$

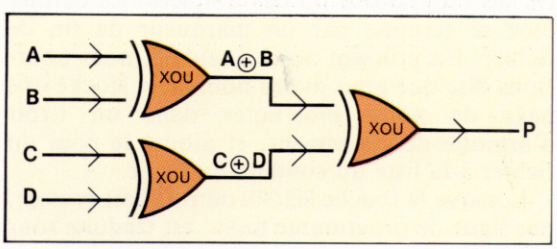
En faisant intervenir les expressions d'une porte logique XOU que nous avons vues précédemment, l'expression pourra s'écrire :

$$P = \overline{(A \oplus B)} \cdot (C \oplus D) + (A \oplus B) \cdot \overline{(C \oplus D)}$$

En considérant chaque terme entre parenthèses comme une entrée à une porte XOU, la même expression deviendra plus simplement :

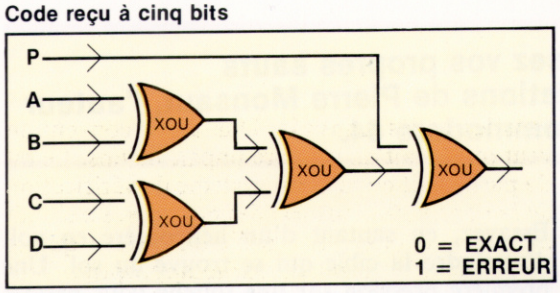
$$P = (A \oplus B) \oplus (C \oplus D)$$

et le circuit formé sera une « cascade » de portes XOU :



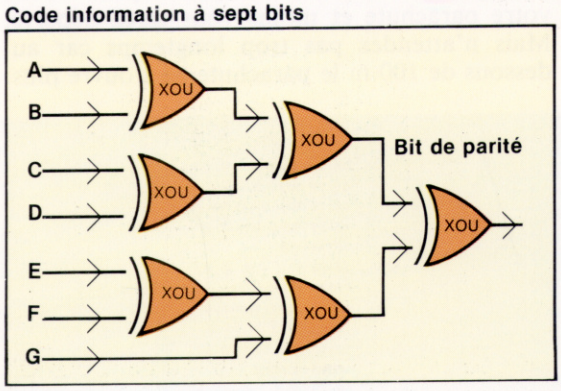
Ce circuit peut être modifié en ajoutant une autre porte logique XOU destinée à comparer le bit de parité reçu à celui généré par le circuit en fin de réception.

Code reçu à cinq bits



La plupart des ordinateurs utilisent les codes standard ASCII pour la transmission des données. Il s'agit d'un code à huit bits dont sept d'information et le dernier de parité. Créons un générateur de bit de parité pour code ASCII :

Code information à sept bits



Encodeur de priorité

L'interruption dans le fonctionnement de l'UC sera suivie d'un numéro de priorité afin de permettre à l'ordinateur de s'occuper des périphériques les plus importants en premier. L'encodeur de priorité que nous allons créer devra relier quatre périphériques en un seul circuit qui sera capable d'identifier le périphérique responsable d'un signal, et appliquera un système de priorité lorsque plusieurs signaux simultanés seront détectés en provenance de deux ou plusieurs périphériques.

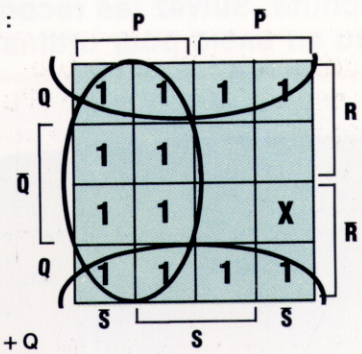
Afin de générer l'information qui spécifie un périphérique parmi quatre, deux lignes de sortie sont nécessaires. Une troisième servira à édicter une interruption. Les quatre périphériques s'appelleront P, Q, R et S : P ayant la priorité la plus élevée et S la priorité la plus faible. Les lignes de sortie s'appelleront A et B; elles identifieront chacune un périphérique. Z signalera qu'il faut avoir recours à une interruption. La table de vérité pour l'encodeur utilisera X pour les cas où l'encodeur n'intervient pas.

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Puisque le périphérique P a la plus forte priorité, nous ne nous soucions pas de savoir si d'autres périphériques se signalent également (étant tous d'un ordre de priorité inférieur).

Les trois lignes de sortie du circuit doivent être étudiées séparément. En prenant la ligne A, le tableau de Karnaugh sera :

Pour A :



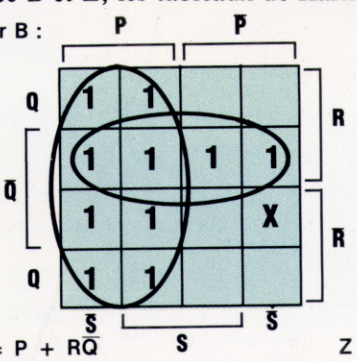
A = P + Q

Les cas où l'encodeur n'intervient pas parmi les résultats pour la ligne A sont représentés par X; mais les cas où l'encodeur n'intervient pas pour les entrées au tableau de Karnaugh sont traités séparément. Prenons le cas où P est à 1, et Q, R et S sont indifférents. Nous devons alors étudier tous les cas de la table pour lesquels P est égal à 1 — il y en a huit en tout. Nous obtenons l'expression simplifiée suivante :

A = P + Q

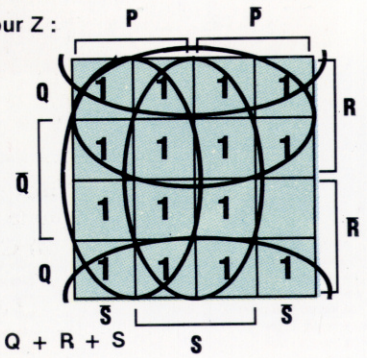
Avec B et Z, les tableaux de Karnaugh sont :

Pour B :



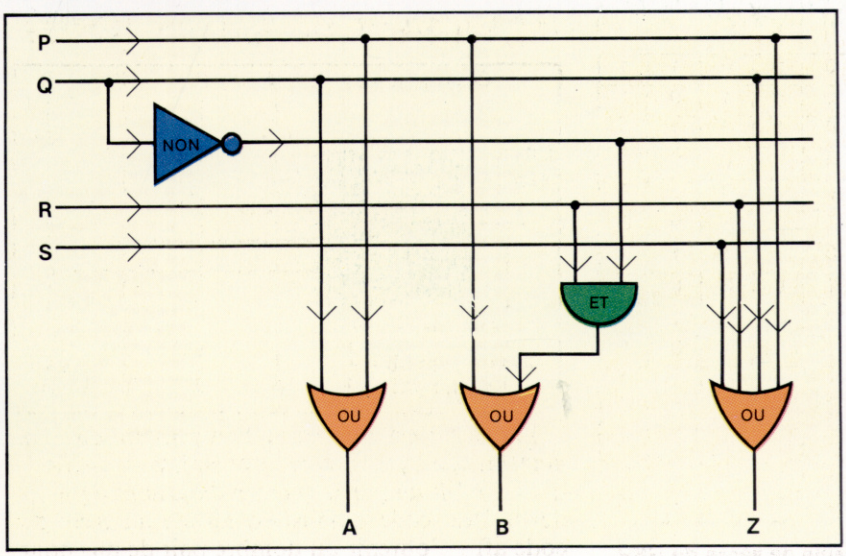
B = P + RQ̄

Pour Z :



Z = P + Q + R + S

En utilisant ces trois expressions, nous aboutissons au circuit suivant :



Parachute

Pour avoir des émotions fortes, réalisez vos propres sauts en parachute. Suivez les recommandations de Pierre Monsaut, l'auteur de ce jeu en BASIC pour ordinateur Commodore 64.

Essayez, en sautant d'un hélicoptère en vol, d'atteindre la cible qui se trouve au sol. Une première pression sur une touche vous permet de descendre verticalement en chute libre. Une seconde pression entraîne l'ouverture du parachute. La descente continue alors plus lentement et avec un angle de 45 degrés car le vent vous pousse. Plus vous attendrez pour ouvrir votre parachute et moins vous serez déporté. Mais n'attendez pas trop longtemps car au-dessous de 100 m le parachute ne s'ouvre plus.

```
5 REM  **************
10 REM * PARACHUTE *
15 REM **************
20 PRINT CHR$(147)
25 GOSUB 10000
30 POKE V+5,50
40 POKE V+1,100
50 POKE V+3,95
70 AV=320
80 N2=320
85 N1=0
90 MS=6
100 POKE 2040,14
110 POKE 2041,14
115 POKE 2042,13
120 POKE 2043,15
125 POKE 2044,16
130 POKE 2045,17
135 POKE V+23,3
140 POKE V+29,3
150 POKE V+39,1
160 POKE V+40,14
170 POKE V+41,5
200 POKE V+42,10
210 POKE V+43,10
215 POKE V+44,5
220 VV=4
230 NN=0
240 SA=0
450 POKE V,N1
460 POKE V+2,63
470 POKE V+4,63
480 POKE V+16,MS
490 POKE V+21,39
500 AV=AV-2
510 IF AV<1 THEN AV=320:MS=MS+4
520 IF AV=254 THEN MS=MS-4
540 N1=N1+1
550 IF N1>320 THEN N1=0:MS=MS+1
560 IF N1=256 THEN MS=MS+1
580 N2=N2-1
590 IF N2<1 THEN N2=320:MS=MS+2
600 IF N2=255 THEN MS=MS-2
610 H=H+VV
620 IF H>255 THEN H=255
630 IF H=230 AND SA>0 THEN 5000
1000 POKE V+16,MS
1010 IF N1<256 THEN POKE V,N1:GOTO 1030
1020 POKE 2043,15
1025 POKE 2044,16
1030 IF N2<256 THEN POKE V+2,N2:GOTO 105
0
1040 POKE V+2,N2-255
1050 IF AV<256 THEN POKE V+4,AV:GOTO 107
0
1060 POKE V+4,AV-255
1070 GET X$
1080 IF X$="" OR AV>255 THEN 1500
1090 ON SA GOTO 1300,1500
1100 SA=1
1110 P=AV
1120 POKE V+6,P
1130 POKE V+21,47
1140 H=58
1150 GOTO 1500
1300 IF H>150 THEN 1500
1305 POKE V+21,55
1310 VV=1
1320 NN=1
1340 SA=2
1500 P=P-NN
1510 IF P<1 AND SA>0 THEN 6000
2000 POKE V+7,H
2010 POKE V+9,H
2020 POKE V+6,P
2030 POKE V+8,P
2040 GOTO 500
5000 IF ABS(L-P)>4 OR SA=1 THEN 6000
5010 SC=SC+10
5020 FOR I=1 TO 1000
5030 NEXT I
5040 RESTORE
5050 GOTO 20
6000 POKE V+21,0
6010 PRINT CHR$(147)
6020 FOR I=1 TO 10
6030 PRINT
6040 NEXT I
6050 PRINT TAB(13)"SCORE :";SC
6060 PRINT
6070 PRINT
6080 PRINT TAB(13)"UNE AUTRE ?"
6090 GET X$
6100 IF X$="" THEN 6090
6110 IF X$<>"N" THEN RUN
6120 END
10000 GOSUB 30000
10010 FOR I=0 TO 29
10020 READ Q
10030 POKE 832+I,Q
10040 NEXT I
10050 FOR I=30 TO 62
10060 POKE 832+I,0
10070 NEXT I
10080 FOR I=0 TO 32
10090 READ Q
10100 POKE 896+I,Q
10110 NEXT I
10120 FOR I=33 TO 62
10130 POKE 896+I,0
10140 NEXT I
10150 FOR I=0 TO 62
10160 READ Q
10170 POKE 960+I,Q
10180 NEXT I
10190 FOR I=0 TO 62
10200 READ Q
10210 POKE 1024+I,Q
10220 NEXT I
10230 FOR I=0 TO 53
10240 POKE 1088+I,0
10250 NEXT I
10260 FOR I=18 TO 20
10270 POKE 1088+I*3,255
10280 POKE 1088+I*3+1,248
10290 POKE 1088+I*3+2,0
10300 NEXT I
10310 RETURN
20000 DATA 0,0,0,127,255,0,0,128,0
20005 DATA 1,192,7,3,240,7,31,252,15
20010 DATA 127,255,255,255,255,255
20020 DATA 255,255,255,127,255,254
20030 REM
20040 DATA 0,112,0,7,255,0,31,255,224
20050 DATA 127,255,248,255,255,254
20060 DATA 255,255,255,255,255,255
20070 DATA 127,255,254,31,255,248
20080 DATA 7,255,192,1,252,0
20090 REM
20100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20110 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20120 DATA 13,128,0,5,0,0,5,0,0,7,0,0
20130 DATA 7,0,0,7,0,0,31,192,0,18,64,0
20140 DATA 23,64,0,7,0,0,0,0,0,0,0,0,
20160 REM
20170 DATA 15,128,0,63,224,0,127,240,0
20180 DATA 127,240,0,255,248,0,255,248,0
20190 DATA 64,16,0,64,16,0,32,32,0
20200 DATA 32,32,0,23,64,0,23,64,0
20210 DATA 18,64,0,31,192,0,7,0,0,7,0,0
20220 DATA 7,0,0,5,0,0,5,0,0,13,128,0
30000 V=53248
30005 POKE V+21,0
30010 POKE V+11,230
30020 L=INT(RND(1)*100)+51
30030 POKE V+10,L
30040 RETURN
```

Atari : derniers cris

Atari est depuis longtemps sur le marché de l'ordinateur domestique avec deux modèles, le 400 et le 800. Ses deux nouveaux modèles « XL » sont vendus à meilleur prix et comportent des raffinements.

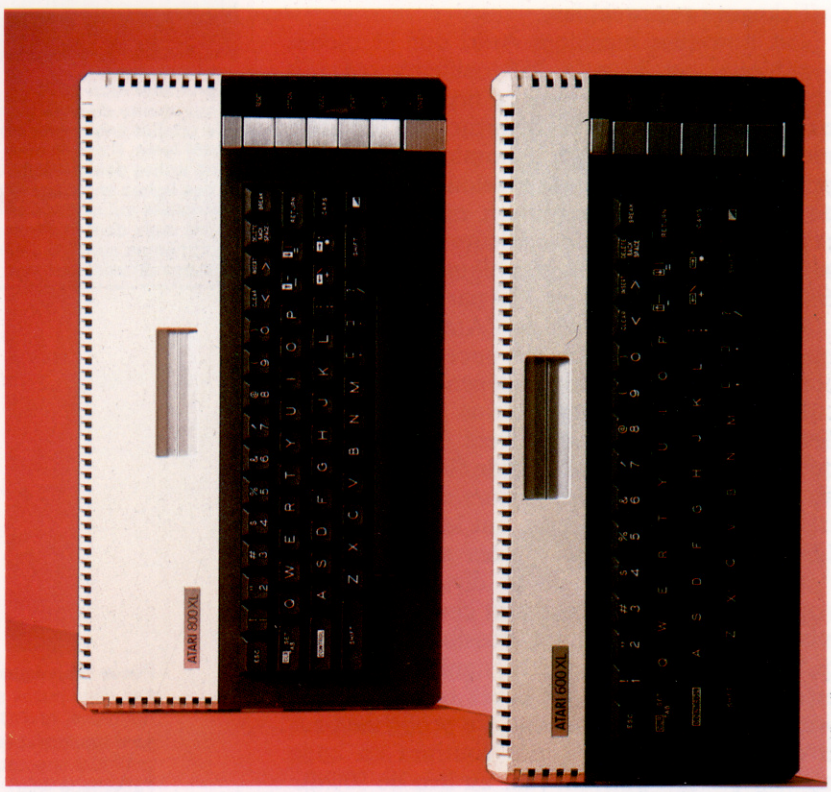
Confronté à une compétition de plus en plus forte, Atari a reconçu ses ordinateurs et a introduit le 600-XL et le 800-XL. Ces nouvelles machines peuvent utiliser les logiciels écrits pour le 400 et le 800, d'où l'accès immédiat à une vaste gamme de programmes, allant des jeux très populaires aux programmes de gestion.

Heureusement, les deux machines ont adopté le clavier du 800. L'Atari 400 a un clavier à membrane. Une frappe rapide est pratiquement impossible en raison de la petite dimension des touches et de la pression devant être exercée pour effectuer un contact. Le 800 dispose donc d'un clavier de type machine à écrire qui est l'un des meilleurs sur le marché.

Les claviers des nouvelles machines sont identiques et ont 62 touches au total : quatre touches de fonction (START, SELECT, OPTION et RESET) qui sont placées du côté droit du clavier ; une touche HELP qui avec certains nouveaux programmes permet de solliciter des écrans d'aide ; avec 29 touches graphiques, les Atari disposent d'un jeu complet de caractères ASCII intégré. Le clavier souffre d'une bizarrerie héritée du 800 : le mode d'utilisation des touches de commande du curseur. Les flèches sont en fait un troisième caractère imprimé sur quatre touches distinctes. Pour déplacer le curseur, vous devez maintenir la touche CONTROL enfoncée tout en pressant la touche fléchée appropriée. Malgré cela, les claviers sont bien conçus et agréables à utiliser.

L'Atari 400 dispose de 16 K de mémoire utilisateur, alors que le 800 offre 48 K. Les deux ordinateurs ont des connecteurs d'extension sur la carte système que l'on peut atteindre en retirant le couvercle de la machine. Les modèles XL ne peuvent être ouverts ; les ports d'extension sont intégrés et placés à l'extérieur du boîtier. La seule différence entre ces deux machines se situe au niveau de la capacité mémoire de la version de base. Le 600-XL offre 16 K, ce qui peut être porté à 64 K en enfichant un module d'extension. Le 800-XL est livré avec 64 K.

Les deux machines XL ont une interface intégrée nommée « Expander ». Il s'agit d'un port d'extension de type bus qui peut être connecté à une variété de périphériques. Il y a aussi un connecteur pour cartouche ROM juste derrière le clavier pour des jeux en cartouche et pour d'autres programmes. L'Atari 800 d'origine a deux connecteurs de cartouche, mais de très rares programmes utilisent le second connecteur. Il a donc été supprimé dans les ordinateurs XL. Le 400 et le 800 ont quatre ports de manche à balai ; mais ici encore très peu de programmes les utilisent. Ils ont donc été ramenés à deux sur les machines XL, et ces ports ne sont plus placés à l'avant du boîtier mais sur le côté.



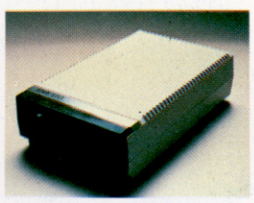
Ian McKinnell.

Les jumeaux
Les ordinateurs 600-XL et 800-XL d'Atari sont très semblables, mais le premier possède 16 K de mémoire contre 64 K pour le second. Comme ce sont des modèles améliorés, ils disposent d'une logithèque importante.

Son et lumière

Tous les ordinateurs Atari peuvent être connectés directement à un téléviseur, et (à l'exception du 400) également à un moniteur d'ordinateur ; mais l'affichage Atari est très bon, même sur un téléviseur. Le jeu de caractères est généralement facile à lire et le contraste entre le texte et l'arrière-plan est assez bon. Il y a de nombreuses options de couleurs, mais l'affichage normal du texte est composé de lettres blanches sur un arrière-plan bleu foncé. Les ordinateurs Atari affichent un maximum de 40 colonnes par 24 lignes de texte. Il existe quatre modes texte avec des affichages différents.

Les Atari furent parmi les premiers ordinateurs à offrir des graphiques à plan objet. Leur fonction plan objet se nomme Player-Missile Graphics, et est contrôlée par une puce interne



Lecteur de disque
Le lecteur de disque est une option intéressante pour le 800-XL, qui lui offre ainsi un surcroît de mémoire de 127 K. Le 600-XL standard ne possède pas une mémoire suffisante pour gérer le lecteur de disque.

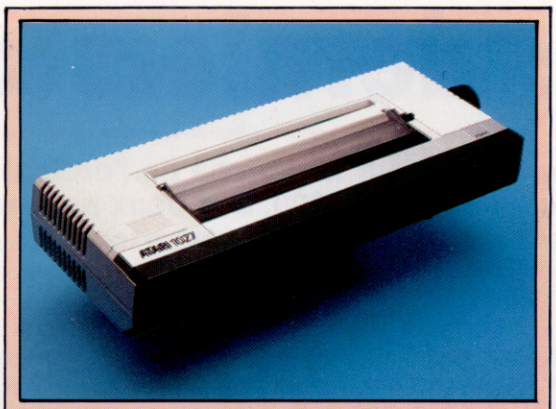
spéciale nommée GTIA. Les figures sont créées à l'aide de points. Dès que la forme d'une figure a été déterminée, les valeurs de chaque point sont écrites (POKE) dans une zone de la mémoire nommée « table de dessin ». Vous avez la possibilité de créer jusqu'à quatre figures qui, chacune, peuvent être associées à un élément missile. Ces figures peuvent être unies ou multicolores; les couleurs sont manipulées sur l'écran en changeant les valeurs dans la table de dessin. Bien que les graphiques Player-Missile ne soient pas faciles à utiliser, ils permettent de créer des affichages remarquables.

Les nouvelles machines Atari ont 11 modes graphiques, et jusqu'à 256 couleurs (en fait 16 couleurs ayant chacune 16 nuances); cependant, en raison de la quantité de mémoire requise pour l'affichage, le nombre de couleurs pouvant être utilisées varie selon la résolution de l'écran. Plus la résolution est élevée, plus petit est le nombre de couleurs pouvant être affichées. La résolution graphique maximale du 600-XL et du 800-XL est 320 par 192 points.

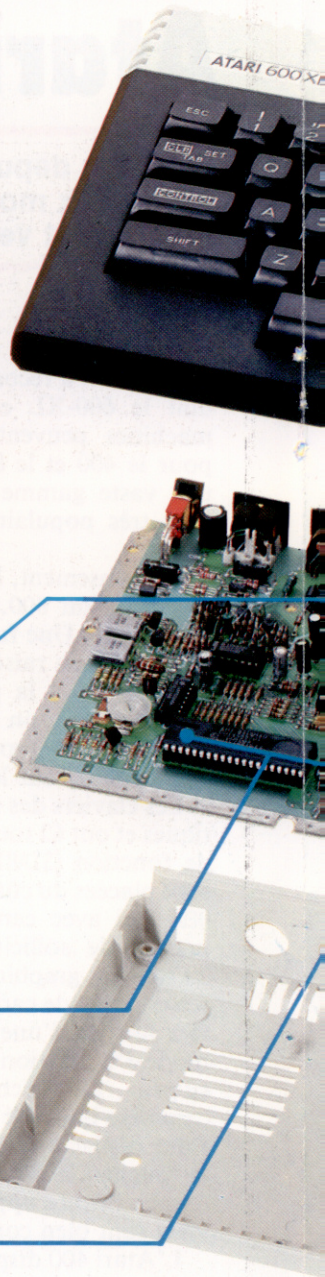
Les fonctions sonores de l'Atari sont également commandées par une puce spécialement conçue. Il y a quatre voix indépendantes et chacune d'entre elles dispose d'une plage de trois demi-octaves. Les voix peuvent être commandées au moyen de la commande SOUND de BASIC, ou en écrivant (POKE) les valeurs dans les registres de mémoire qui produisent les divers sons. Les sons peuvent être ajustés au niveau de l'oscillation, du ton, de la distorsion et du volume. Lorsque les voix sont commandées au moyen de SOUND, il n'est possible d'amorcer qu'une voix à la fois. Cela signifie que pour créer une harmonie vous devez mettre chaque voix en fonction séparément. Ce problème peut être résolu en utilisant des routines en code machine, ou en utilisant POKE au lieu de SOUND.

Les Atari 400 et 800 n'ont aucun langage intégré; vous devez utiliser une cartouche séparée. Les 600-XL et 800-XL ont un BASIC Atari intégré. Il ne s'agit pas du meilleur BASIC. Par exemple, il n'y a pas de commande CIRCLE, aucune fonction PRINT ou PRINT USING, pas de numérotation ou de renumérotation automatique, et pas de variables de nombres entiers. Cependant il est possible de se procurer un BASIC étendu et un BASIC Microsoft sous forme de cartouche ROM.

Pour accompagner ses nouveaux ordinateurs, Atari a reconçu les périphériques existants et a augmenté le nombre d'options offertes en extension. Le périphérique probablement le plus utile est le boîtier d'extension, qui s'enfiche dans l'Expander. Il offre huit connecteurs d'extension qui peuvent recevoir des cartes d'interface pour plusieurs périphériques, deux ports série RS232 et un bus parallèle. Atari offre aussi un module CP/M avec un microprocesseur Z80, le système d'exploitation CP/M 2.2 et un affichage 40/80 colonnes commutable.



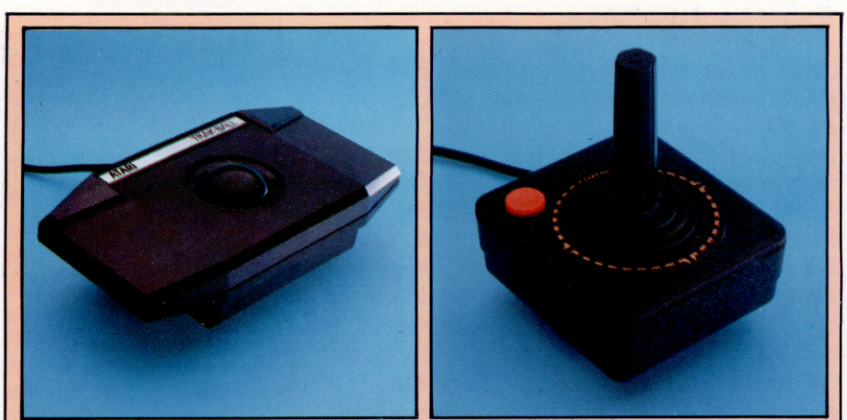
Imprimante Atari 1027
L'imprimante dispose d'une tête d'impression à boule. Elle produit une impression aussi bonne mais elle est plutôt lente. Il existe deux autres imprimantes Atari; l'une utilise des stylos-bille pour dessiner des lettres et des lignes en quatre couleurs, et une imprimante matricielle qui donne une qualité d'impression moins bonne mais dont la vitesse est plus grande. Ces imprimantes sont les seules qui puissent être utilisées directement avec les machines XL.



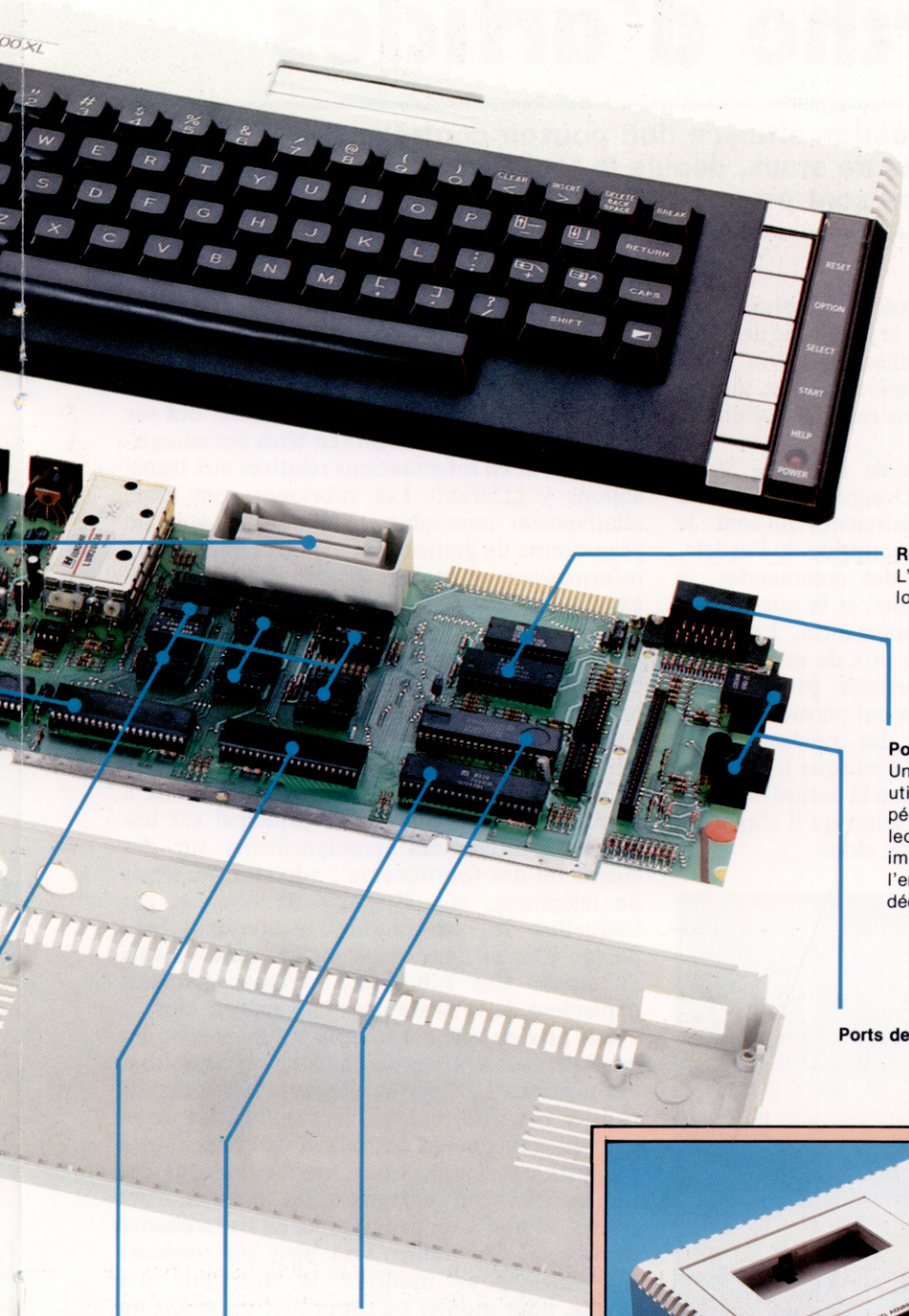
Port cartouche
La gamme XL a un seul connecteur pour cartouches ROM.

Puces graphiques
Les deux puces spéciales intégrées, ANTIC et GTIA, sont responsables du potentiel spectaculaire de l'affichage Atari.

RAM
Les 16 K de RAM sont contenus dans des puces.



Boule et bâton
Atari offre deux systèmes pour diriger les jeux. La méthode conventionnelle part de la manette Atari (à droite), un modèle du genre. La commande à boule (à gauche) permet de combiner les deux opérations du système précédent.



ROM
L'intercepteur BASIC est logé dans ces deux puces.

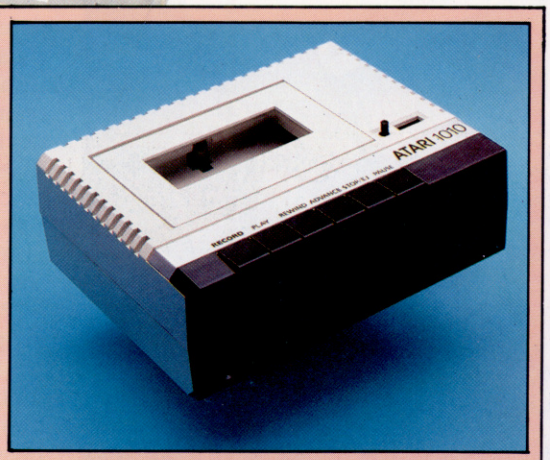
Port périphérique
Un port à 13 broches est utilisé pour connecter les périphériques, dont les lecteurs de disquettes, les imprimantes et l'enregistreur à cassette dédié.

Ports de manches à balai

Puce sonore
Une puce fabriquée sur commande nommée POKEY est responsable de la génération sonore.

Puce E/S
Un 6520 gère les ports d'entrée et de sortie.

UC
L'Atari est basé sur une puce 6502 rapide.



Unité à cassette Atari
L'Atari ne peut utiliser que sa propre unité à cassette. Cela accroît le prix du système mais offre aussi des avantages. D'abord, l'unité étant construite par Atari, elle est plus fiable. Ensuite, l'enregistreur utilise deux pistes. L'une des pistes sert à sauvegarder les programmes de façon normale, l'autre enregistre des sons. Cela permet d'avoir des programmes d'apprentissage linguistique parlants.

ATARI 600 XL 800 XL

PRIX
600-XL : *
800-XL :**

DIMENSIONS
600-XL : 380 × 170 × 40 mm.
800-XL : 380 × 220 × 40 mm.

UCT
6502, 2Mhz.

MÉMOIRE
RAM de 16-64 K,
ROM de 24 K.

AFFICHAGE
Jusqu'à 24 lignes de 40 colonnes de texte; graphiques jusqu'à 320 × 192 points avec des plans objets et 16 couleurs dans 16 niveaux de brillance.

INTERFACES
Manches à balai (2), port périphérique, port d'extension, port cartouche.

LANGAGE DISPONIBLE
BASIC, FORTH, LOGO, PILOT, langage d'assemblage du 6502.

CLAVIER
De type machine à écrire avec 62 touches, dont des touches de commande de curseur et des touches de fonction dédiées comme SELECT et START pour la commande des programmes.

DOCUMENTATION
Les manuels n'ont jamais été le point fort d'Atari. Ce constructeur a tendance à considérer ses ordinateurs comme des machines à caractère ludique et il a négligé les détails techniques. Il existe cependant une vaste gamme d'excellents manuels et de magazines indépendants.

FORCES
Les Atari continuent à offrir les meilleurs jeux, avec de superbes graphiques et sons, et un choix immense.

FAIBLESSES
Les Atari peuvent être coûteux — vous avez besoin d'une unité à cassette dédiée et le prix des logiciels est anormalement élevé. La programmation graphique et sonore est aussi plus difficile que sur de nombreuses machines.

Recherche d'articles

Un programme de gestion efficace doit pouvoir contrôler tous les mouvements de stock, depuis le fournisseur jusqu'aux rayons où les articles sont présentés.

Nous avons déjà vu comment les articles sont identifiés et répertoriés sur le fichier de données du stock. Cependant l'utilisateur devra attribuer, en plus de chaque numéro de code, un certain nombre d'informations relatives à l'article concerné.

Le progiciel de gestion de stocks « Stock Recording System » du Dragon 64 comporte 6 zones pour chaque enregistrement qui sont : le numéro de l'article, une description de l'article, l'état du renouvellement des commandes, le prix coûtant, le prix de vente, et la quantité.

Les zones sont très importantes. La différence entre prix coûtant et prix de vente donne le bénéfice brut. Les articles peuvent être regroupés par catégorie, ce qui permet d'analyser les ventes et d'en tirer des conclusions. La zone quantité est essentielle puisque la désignation de la quantité varie selon la nature de l'article : à l'unité ou par lots selon qu'il s'agit par exemple de vêtements ou de clous.

Réponses sur le stock
« Stock Recording System » est un programme d'application destiné à gérer les stocks sur un micro-ordinateur Dragon 64 et devant fonctionner sous le système d'exploitation OS9. Il s'agit d'un système d'exploitation multiprogramme et multitâche développé par Dragon à partir du système d'exploitation Unix.
(Cl. Ian McKinnell.)

Le plus intéressant avec un logiciel de gestion de stocks est sans doute l'aspect dynamique des données, plutôt que statique, comme c'est le cas par exemple avec un programme de comptabilité. Ce dernier consiste en des informations sur un compte ou sur un client (en-têtes des enregistrements) et en informations relatives aux transactions y afférant. Les rubriques sont alors relativement immuables. Par contre, avec un programme de gestion de stocks, la limite entre informations constantes et informations changeantes est beaucoup moins nette.

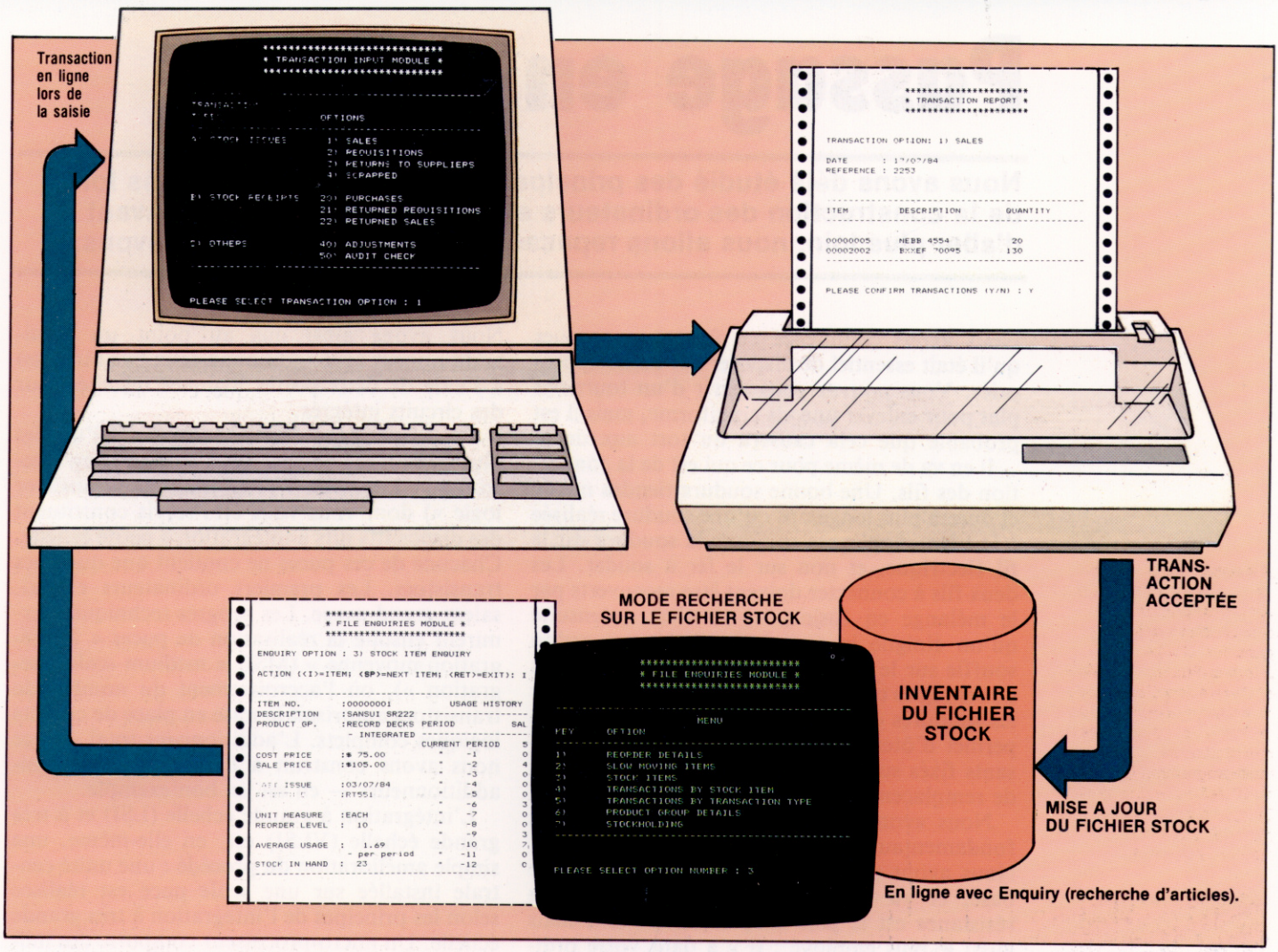
La désignation de l'article, les descriptions du groupe d'articles auquel il appartient et son code constituent la partie fixe de l'enregistrement. Cependant, ces rubriques sont continuellement modifiées à partir d'informations découlant des transactions sur le stock.

Le programme comportera un aspect mise à jour de rubriques du fichier principal sur lesquelles figurent des renseignements sur des clients ou des fournisseurs : adresses, numéros de téléphone, etc. Il s'agit de données sur lesquelles le programme n'intervient pas puisqu'elles ne dépendent pas des mouvements de stock. Ces rubriques, variant rarement, dépendent de la saisie pour leur mise à jour.

Les prix de vente et les prix coûtants sont des rubriques susceptibles de changer à chaque nouveau stock. La solution logique est alors de confier ces informations (accompagnées de la date du changement des prix et de l'importance du stock concerné), à une routine de saisie des transactions qui enregistre les marchandises reçues, et non à un programme de maintenance de fichier. Ce dernier sera bien sûr toujours nécessaire pour modifier le descriptif des stocks, pour ajouter ou supprimer un article du fichier stock, mais une masse considérable de données proviendra des informations de base sur les transactions.

Pour comprendre le fonctionnement d'un programme de gestion de stocks, il faut examiner les routines de saisie ainsi que les possibilités d'études sur les ventes qu'elles permettent, qui aboutissent à des rapports. Nous avons choisi le programme du Dragon comme exemple et avons représenté ses divers fichiers sur notre diagramme. Les trois éléments à retenir sont les routines d'entrée de transactions, le détail des transactions et le fichier de recherche d'un article du stock.

La présentation des états d'entrée de transactions est toujours la même sur le système Dra-



gon. Les types de transactions sont plus ou moins complexes. Nous nous en tiendrons pour l'instant aux ventes (c'est-à-dire aux mouvements de sortie du registre du stock). Remarquons en premier lieu que les programmes de gestion sont conçus pour être accessibles et progressifs pour l'utilisateur. Ils assisteront ce dernier par des messages-guides opérateur offrant toutes les informations dont ils ont besoin. Cette démarche suppose qu'il faut concilier deux contraintes de programmation pour réaliser un progiciel efficace et à la portée de tous.

D'une part le programme doit pouvoir reconnaître certaines zones de données et effectuer des opérations arithmétiques ou autres sur les données ; d'autre part, il doit guider l'utilisateur et détecter les données saisies à mauvais escient. Il doit donc vérifier les données.

Une fois les données d'appel de saisie tapées ainsi que le numéro de référence servant à identifier l'accès au type de données que l'utilisateur désire entrer dans le programme (le même numéro figurera sur les factures permettant la saisie) et le numéro de l'article, l'ordinateur ira lire le fichier du stock afin de savoir si l'article en question existe. S'il trouve ce numéro, il affichera automatiquement l'état de cet article. L'utilisateur peut ainsi vérifier qu'il a bien demandé l'article correspondant aux données et entrer, ensuite, le volume des ventes approprié. A partir de cette information, l'ordinateur pourra travailler et établir des rapports d'analyse. Ainsi, l'option FILE ENQUIRIES (analyse du fichier), du menu principal, comporte un sous-menu de sept options couvrant les articles du stock, les transactions et les groupes de produits. Ces options sont : STOCK DETAILS (détail du stock), SLOW MOVING ITEMS (articles peu demandés), RE-ORDER DETAILS (détail du renouvellement des commandes), TRANSACTIONS (BY STOCK ITEMS) (transactions par article), TRANSACTIONS (BY TRANSACTION TYPE) (transactions par type), PRODUCT GROUP DETAILS (détail du groupe de produits), et STOCKHOLDING (tenue du stock).

Les transactions sur les ventes influent sur tous ces types de rapports d'analyse. S'il faut un rapport sur le renouvellement des commandes, par exemple, le programme vérifiera que le stock vendu correspond bien au stock précédemment rentré pour les mêmes articles.

Chaque détail d'un état d'affichage d'une transaction de vente a son importance et est susceptible d'être utilisé d'une manière ou d'une autre dans un rapport d'analyse.

L'état du programme Dragon de recherche d'un article fournit une bonne illustration des ressources de traitement qu'offrent les transactions de vente.

Saisie validée
Le programme de commande de l'état du stock reçoit les informations sur les transactions. Il vérifie la validité de l'accès à la saisie, ainsi que la validité des transactions de données tapées. Le fichier du stock, qui comporte l'état individuel du stock par article, est alors en ligne pour modification. Le programme comporte également un module base de données qui permet à l'utilisateur d'analyser le contenu du fichier de commande d'état du stock, et de pouvoir ainsi créer des rapports d'analyse des divers aspects de son inventaire.
(Cl. Collins/Duncan Smith.)

Passage en revue

Nous avons déjà étudié des principes et des techniques utilisés lors de la construction des ordinateurs et de leurs périphériques. Avant d'aller plus loin, nous allons repasser en revue ce que nous savons.

Nous avons commencé en faisant remarquer qu'il était essentiel de disposer des outils appropriés. Vous *pouvez* vous servir d'un tournevis plat pour enlever une vis cruciforme, mais il est probable que cela causera du tort aux deux.

Il en va de même pour ce qui est de la connection des fils. Une bonne soudure tiendra mieux et durera plus iongemps qu'une soudure réalisée à la hâte. Appliquez toujours la soudure sur le fil électrique, et non sur le fer à souder. Les deux fils à connecter doivent être recouverts par le mélange en fusion, et ce n'est qu'ensuite qu'on éloigne la source de chaleur et qu'on souffle sur le tout pour le refroidir. Le respect de ces règles très simples vous permettra d'éviter les soudures « sèches ». La « désoudure » vous permet d'ôter sans problèmes certains composants fixés sur des circuits imprimés, à des fins de remplacement ou de réparation.

Nous avons également abordé les principes fondamentaux de l'électronique numérique. Nous avons passé en revue les composants de base, et vu comment ils fonctionnaient. La résistance est le plus simple d'entre eux, mais aussi le plus employé : il y a dans votre ordinateur bien plus de résistances que de circuits intégrés.

Les capacités sont elles aussi d'emploi fréquent. Dans les ordinateurs, elles ont pour fonction essentielle de filtrer le bruit indésirable qui s'attache à tout signal, à chaque fois que celui-ci est modifié, amplifié ou déplacé. Il est donc indispensable de lui redonner sa valeur primitive. Le moyen le plus simple consiste à se servir d'une capacité.

Le transistor reste pourtant le plus important de tous ces composants. Il est essentiel, lorsqu'il s'agit de modifier et de manipuler les signaux qui parcourent l'ordinateur. Il peut les amplifier et il sert également de commutateur, en permettant ou en empêchant leur passage. Plus intéressant encore : un signal peut être traité de cette façon en concordance avec un autre signal, ou plusieurs d'entre eux. C'est par exemple ce qui se passe à l'intérieur des portes logiques.

Nous en avons construit trois parmi les plus simples (NON, OU et ET), à l'aide de quelques composants. Il est vrai que dans les circuits intégrés, elles sont fréquemment bâties à l'aide de transistors différents, plus complexes.

En elles-mêmes, les portes logiques n'ont qu'une utilité restreinte. Mais en les combinant entre elles, on obtient des circuits logiques capables d'effectuer des opérations sur des données. Nous avons ainsi mis au point un demi-additionneur (qui additionne deux bits binaires) à l'aide de deux portes logiques installées sur des circuits intégrés.

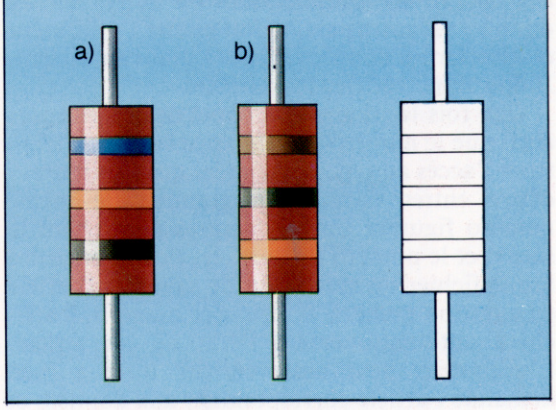
Ce dernier type de composants est l'un des plus complexes de tous ceux employés en électronique. Les puces TTL (« transistor-transistor logic ») dont vous vous êtes servis constituent des dispositifs dits à intégration à petite échelle. Chacune de ces puces ne contient que quelques transistors. Les premiers ordinateurs en faisaient grand usage. Les progrès techniques permirent ensuite la réalisation de puces « à intégration moyenne » (MSI, « medium scale integration »), où l'accroissement du nombre de transistors autorisait la mise en place de circuits logiques complets. L'additionneur total — que nous avons construit à l'aide de deux demi-additionneurs — en est un bon exemple.

L'intégration à grande échelle (LSI) et à très grande échelle (VLSI) est, en elle-même, une simple amélioration du procédé : une unité centrale installée sur une seule puce est réalisée selon les principes de l'intégration à très grande échelle. Un micro-processeur comporte des milliers de transistors, et il est très difficile d'imaginer quel peut bien être le circuit logique correspondant; mais cette complexité même donne à la puce une très grande puissance, et facilite son insertion dans un circuit imprimé.

La connaissance de ces principes fondamentaux vous permettra d'aller plus loin et de réaliser certains ajouts utiles à votre ordinateur.

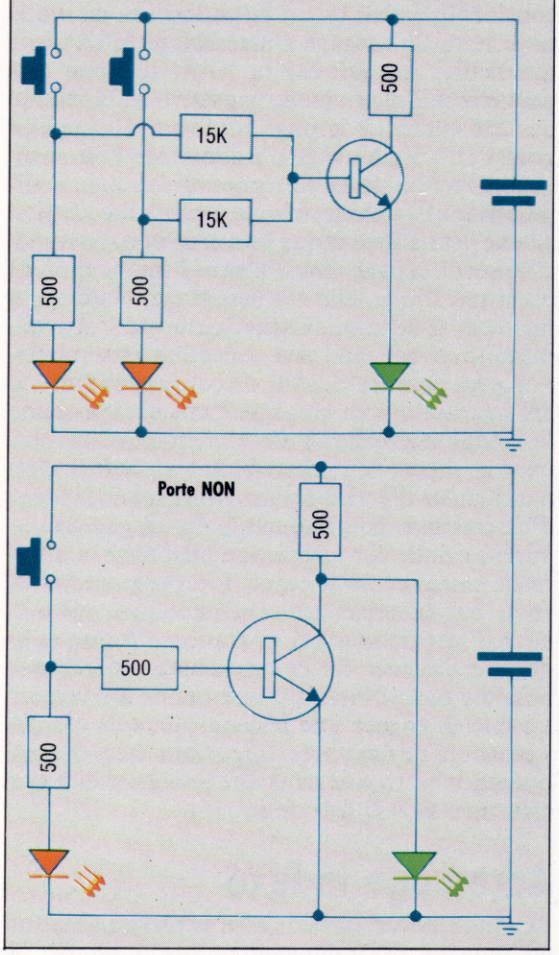
1. Résistances

Les résistances sont entourées de bandes colorées permettant de déterminer leur valeur. Quelle est la résistance de celles qui sont présentées ci-dessous ? Quelles sont les bandes de couleur qui permettent d'identifier une résistance de 150 Ω (ohms) ?



2. Porte NON-OU

Nous avons déjà réalisé les portes NON, OU et ET, à l'aide de quelques transistors. Nous allons maintenant nous efforcer de construire une porte NON-OU, en partant des mêmes principes. Pour vous aider, vous trouverez ci-dessous les circuits OU et NON. Le problème peut être résolu de deux façons. Vous pouvez simplement combiner une porte OU et une porte NON. Mais une méthode plus rapide peut être découverte en étudiant attentivement le circuit NON.



3. Convertisseur décimal/binaire

Construisez un circuit permettant de passer du système décimal au binaire. Pour des raisons de simplicité, nous nous bornerons aux nombres binaires à deux bits, c'est-à-dire aux nombres décimaux compris entre zéro et trois. Votre circuit devra comporter quatre commutateurs d'entrée intitulés respectivement zéro, un, deux et trois. Quand vous actionnez l'un d'entre eux, les deux bits binaires correspondants doivent être signalés par une paire de diodes. Une table de vérité (partielle) destinée au convertisseur a cette allure :

BOUTON ZÉRO	BOUTON UN	BOUTON DEUX	BOUTON TROIS	HI-BIT	LO-BIT
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

4. DCB or not DCB?

Nous avons étudié le système de numérotation binaire codé décimal (DCB). Il se situe à mi-chemin entre le décimal et le binaire. Chaque chiffre d'un nombre décimal est converti en son équivalent binaire, et les deux groupes de quatre bits ainsi obtenus sont rassemblés pour donner le résultat en DCB. 53 devient par exemple 01010011 ; 5 est représenté par 0101 et 3 par 0011. Cela signifie que pour être exact tout chiffre entre 0000 et 1001 a un correspondant de 0 à 9 en décimal. Les groupes allant de 1010 à 1111 sont illégaux en DCB.
Construisez un circuit permettant de déterminer si un groupe quelconque de quatre bits a ou non valeur légale. Il devra comporter quatre commutateurs (B0, B1, B2, B3), représentant le nombre à tester. Les sorties seront au nombre de deux : une diode verte si le nombre est légal, une diode rouge s'il ne l'est pas. La table de vérité a l'aspect suivant :

Décimal équivalent	B3	B2	B1	B0	DCB légal	DCB illégal
0	0	0	0	0	1	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	0
4	0	1	0	0	1	0
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	0	1

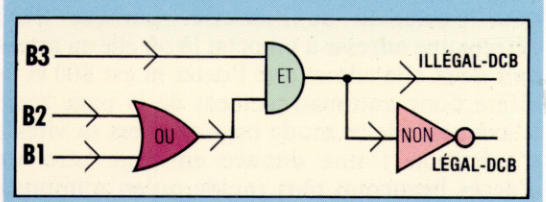
Nous pouvons en conclure que le signal DCB nous est donné par $\overline{B3} + \overline{B2} \cdot \overline{B1}$. Tout signal illégal en est l'équivalent NON :

$$\overline{\overline{B3} + \overline{B2} \cdot \overline{B1}}$$

Ce qu'on peut simplifier en :

$$\overline{\overline{B3}} \cdot \overline{\overline{B2} \cdot \overline{B1}}$$
$$B3 \cdot (B2 + B1)$$

Le circuit qu'il nous faut construire est donc très simple et ne nécessite que trois des entrées. Nous vous suggérons un diagramme logique de ce genre.



Modes d'adresses

La force et la souplesse des instructions du langage d'assemblage sont rehaussées par les possibilités d'adressage mémoire. Il peut être direct, indirect, indexé, ou bien une combinaison de ces trois modes.

Toute instruction en langage d'assemblage se réfère au contenu de mémoire, et puisqu'un octet ne se distingue d'un autre que par son adresse, toute instruction en langage d'assemblage doit donc se référer à une adresse au moins. La manière dont on s'y réfère peut être directe et évidente, comme dans LDA $E349, ce qui signifie « charger l'accumulateur avec le contenu de l'adresse $E349 ». Dans ce cas, à la fois l'accumulateur et l'adresse $E349 sont mentionnés sans ambiguïté.

Par ailleurs, la référence à une adresse peut être moins évidente : RET, qui signifie « retour de sous-programme », est un bon exemple. Il n'y a aucune référence évidente à une adresse, à moins de développer l'instruction par « l'adresse de la prochaine instruction à effectuer est à l'emplacement où a été effectué le dernier appel à un sous-programme ». Ici, on ne mentionne pas l'adresse dont il faut changer le contenu (c'est-à-dire le compteur de programme, registre portant l'adresse de la prochaine instruction à effectuer), ni l'adresse où doit se trouver son nouveau contenu. Ces deux instructions constituent des exemples opposés de *modes d'adressage*.

Jusqu'à présent, nous avons vu des instructions dans deux types de modes d'adressage : *mode immédiat,* comme dans LD A,$45 ou ADC #$31, et *mode direct absolu,* comme dans STA $58A7 ou LD ($696C),A. On peut penser qu'il s'agit là des modes d'adressage « naturels », couvrant tous les cas possibles à l'exception de modes implicites tels que RTS ou RET, mais il y a encore d'autres possibilités.

Adressage de la page zéro

L'*adressage de la page zéro* (aussi connu sous le nom d'*adressage court*) est utilisé chaque fois qu'une instruction se réfère à une adresse comprise entre $0000 et $00FF. Toutes ces adresses ont un octet hi égal à $00 et se trouvent donc en page zéro de la mémoire. Ces instructions n'ont besoin que de deux octets — un pour l'opc et un pour l'octet lo de l'adresse. Lorsque l'UC détecte une adresse à un octet là où elle en attendait deux elle admet que l'octet hi est $00 et se réfère donc automatiquement de la page zéro. L'avantage de ce mode page zéro est sa vitesse d'exécution : une donnée en page zéro est d'accès beaucoup plus rapide qu'en n'importe quelle autre page, car elle ne requiert qu'une adresse d'un seul octet.

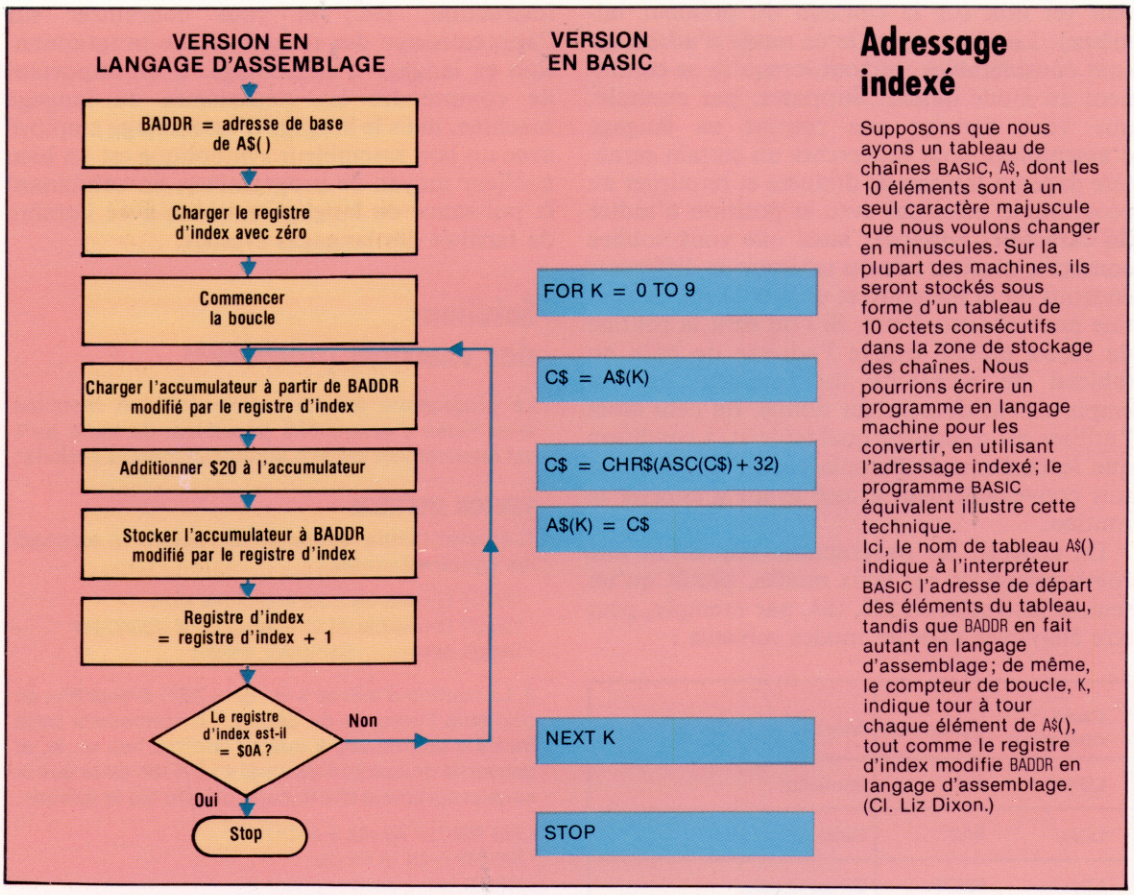
Les microprocesseurs Z-80 et 6502 diffèrent considérablement quant à l'utilisation du mode page zéro. En langage d'assemblage 6502, toute instruction qui adresse la RAM (comme LDA) peut être utilisée en mode page zéro, et la totalité des 256 octets de la page zéro sont disponibles pour l'UC. En langage d'assemblage Z80, seule une intruction — RST (« recommencer » ou « réinitialiser ») — caractérise le mode page zéro, et elle ne peut adresser que huit emplacements spécifiques de la page zéro. Puisque l'instruction RST est si spécifique, elle n'a besoin que d'un opc à un octet (l'adresse faisant partie de l'opc lui-même), ce qui rend son exécution très rapide.

Il peut paraître ridicule de comparer la vitesse des instructions en langage d'assemblage quand le temps d'exécution de l'instruction la plus lente se mesure en microsecondes; mais il n'est pas difficile d'écrire des programmes en langage d'assemblage dans lesquels le fait de gagner une microseconde par instruction peut faire la différence entre succès et échec. Les programmes de jeux, par exemple, peuvent impliquer des millions d'instructions par opération d'écran, et ils doivent exécuter ces commandes aussi vite que possible dans l'intérêt d'une bonne animation. Le fait de rogner une microseconde de chaque opération devient très important quand cette opération se trouve dans une boucle et doit être effectuée 64 000 fois de suite.

Adressage indexé

Le *mode indexé* est vital pour la programmation en langage d'assemblage puisqu'il permet la construction de structure de données de type tableau. Sans cela, les programmes en sont réduits à adresser des emplacements mémoire individuellement : c'est ce que nous avons fait dans tous les programmes jusqu'à présent. Cependant, si l'indexation est possible, on peut traiter bien plus de données et le programmeur dispose de plus de puissance.

Les éléments essentiels du mode indexé sont une *adresse de base* et un *indice*. Supposons que vous vouliez tenir un tableau de données — les codes de caractères ASCII, par exemple — dans des octets consécutifs. L'adresse de base du tableau est l'adresse du premier octet. Cela dit, nous pouvons nous référer à tous les octets suivants du tableau par leur position relative à l'adresse de base, de sorte que le premier octet est à la position zéro, le deuxième à la position un, le troisième à la position deux, et ainsi de



Adressage indexé

Supposons que nous ayons un tableau de chaînes BASIC, A$, dont les 10 éléments sont à un seul caractère majuscule que nous voulons changer en minuscules. Sur la plupart des machines, ils seront stockés sous forme d'un tableau de 10 octets consécutifs dans la zone de stockage des chaînes. Nous pourrions écrire un programme en langage machine pour les convertir, en utilisant l'adressage indexé; le programme BASIC équivalent illustre cette technique.
Ici, le nom de tableau A$() indique à l'interpréteur BASIC l'adresse de départ des éléments du tableau, tandis que BADDR en fait autant en langage d'assemblage; de même, le compteur de boucle, K, indique tour à tour chaque élément de A$(), tout comme le registre d'index modifie BADDR en langage d'assemblage. (Cl. Liz Dixon.)

suite. Une position d'octet relative à l'adresse de base de la table est appelée son indice, et l'adresse absolue d'un octet du tableau est calculée en faisant la somme de l'adresse de base et de l'indice de l'octet. Si nous pouvons construire une boucle de programme en langage d'assemblage et utiliser le compteur de boucle comme un indice d'adresse de base du tableau, alors nous pouvons adresser chaque octet du tableau en séquence, exactement comme nous pourrions accéder aux éléments d'un tableau BASIC utilisant une boucle FOR... NEXT.

Là encore, les langages d'assemblage Z-80 et 6502 manient différemment l'adressage indexé. La puce 6502 contient deux registres à un seul octet, appelés X et Y, dont chacun peut porter un indice qui modifie une adresse de base. Cela limite la longueur d'un tableau à 256 octets (le plus grand nombre possible à un octet). La puce Z-80 contient deux registres à deux octets, IX et IY, qui peuvent porter l'adresse de base elle-même, puis sont incrémentés ou décrémentés pour indiquer les octets successifs du tableau. Puisque ce sont des registres à deux octets, IX et IY peuvent adresser n'importe quel octet adressable par l'UC elle-même. Leur contenu peut aussi être modifié par un indice à un octet.

Adressage indirect

L'adressage indirect implique l'utilisation d'adresses de pointeur, concept que nous avons déjà introduit, en relation avec les limites flottantes en mémoire. Imaginez qu'un groupe de personnes créent un ciné-club et qu'elles se rencontrent chaque semaine pour voir un film choisi par le président du club. Le film peut être présenté dans une douzaine de cinémas différents; aussi, lorsqu'il a choisi le film pour la semaine, le président écrit-il les détails concernant l'heure et le lieu sur la vitrine d'un magasin du centre ville. Les membres du club ne savent pas d'une semaine à l'autre où passera le film, mais ils savent où se trouve le magasin, et ce dernier leur « indique » le bon cinéma. L'adresse du magasin est, indirectement, l'adresse du cinéma.

En mode d'adressage indirect, il est possible d'écrire des instructions qui contiennent l'adresse d'un pointeur, et qui opèrent sur le contenu de l'emplacement qu'indique le poin-

Renée Panier

Pointeur indicateur
Des exemples d'adressage indirect ne se présentent pas tous les jours. Toutefois, sur cette photo, le panneau indicateur de trains contient les données demandées par le voyageur, de sorte que le signe lui indiquant où trouver ce panneau le renvoie indirectement à cette donnée. L'adressage indirect en langage d'assemblage signifie que l'adresse fournie dans l'opérande contient l'adresse de l'octet où est stockée la donnée; l'adresse de l'opérande est un pointeur.

teur (et non sur le contenu du pointeur lui-même). Les avantages de ce mode d'adressage sont considérables, surtout lorsqu'ils se combinent au mode indexé. Supposez, par exemple, que vous écriviez une routine en langage d'assemblage pour rechercher un certain caractère dans un tableau de données et retourner au programme principal avec la position d'indice du caractère. Supposez aussi que vous vouliez conserver plusieurs de ces tableaux en différents endroits de la mémoire et utiliser la même routine pour chacun d'eux. Si l'on écrit la routine de façon qu'elle trouve l'adresse de base du tableau de recherche indirectement via un emplacement de pointeur donné, on peut alors l'utiliser sur n'importe quel tableau, à condition que le contenu de l'emplacement du pointeur soit convenablement ajusté avant d'appeler la routine.

En général, les programmes requièrent une combinaison de ces deux modes, plutôt qu'un seul. L'instruction 6502, LDA, par exemple, peut être employée dans les modes suivants :

Code opération	Opérande	Mode
LDA	$34	Immédiat
LDA	$A2	Direct page zéro
LDA	$967F	Direct absolu
LDA	$A2,X	Page zéro, indexé par X
LDA	$967F,X	Absolu, indexé par X
LDA	$967F,Y	Absolu, indexé par Y
LDA	($A2),X	Indirect, pré-indexé par X
LDA	($A2),Y	Indirect, post-indexé par Y

Il est commode d'utiliser une intruction 6502 dans cet exemple parce que cela montre très clairement les combinaisons de modes d'adressage. Notez que les deux variantes indirectes de l'instruction sont en page zéro ainsi qu'en modes indirect et indexé. Un tableau tel que celui-là peut paraître peu évident à première vue, mais l'utilisation des différents modes rend bientôt claire leur signification.

Le tableau ne répond pas à une question : comment savoir le mode d'adressage d'une instruction lorsque le mnémonique est le même dans tous les cas? On peut voir que le format de l'opérande est différent pour chaque mode, et que la seule ambiguïté possible consiste à savoir si une instruction telle que LDA SYMB1 est en mode page zéro ou absolu. Un programme assembleur résoudra cela à votre place, mais si vous assemblez le programme à la main, vous devrez déterminer si SYMB1 a été défini comme une quantité à un ou deux octets.

En général, une fois que vous commencez à utiliser un assembleur, vous pouvez oublier les opc (codes opérations), le nombre d'octets par instruction, etc., et vous concentrer sur l'apprentissage des techniques de programmation en langage d'assemblage. Il est important de comprendre les mécanismes du langage machine, mais le langage d'assemblage employé avec un bon assembleur symbolique est un bien meilleur moyen de programmer, en combinant la puissance du langage machine avec nombre de facilités des langages évolués.

Solutions des exercices précédents

Le programme moniteur était écrit en modules avec l'idée d'extensions possibles, de sorte qu'il est relativement facile d'ajouter un menu d'options :

VERSION SPECTRUM

1. Ajustez l'initialisation en éditant ou en ajoutant les lignes suivantes :

```
1050 LET LT=5:DIM C$(LT):DIM O$(LT,24):DIM X$(16)
1150 LET C$=«ADGQB»:LET C1 =-48:LET C2=10-CODE(CS$(1))
1280 LET O$(5)= «      B-AFFICHAGE BINAIRE»
```

2. La routine d'entrée à la ligne 2000 a déjà mis en évidence l'adresse de départ, l'a normalisée sous forme d'un nombre à quatre chiffres hex en A$, et convertie en nombre décimal en DN, de sorte que le sous-programme d'affichage binaire est le suivant :

```
7000 REM**S-P AFF HEX & BIN**
7020 FOR P=DN TO (DN+15)
7040 LET NM=P:GOSUB 3100:PRINT H$,
7060 LET N=PEEK(P): LET NM=N
7080 GOSUB 3000:PRINT B$;« »;
7100 GOSUB 7300:PRINT B$
7120 IF P=65535 THEN LET P=DN+15
7140 NEXT P
7200 RETURN
7300 REM** S-P OCTET BINAIRE**
7310 LET B$=«»
7320 FOR D=8 TO 1 STEP-1
7330 LET N1=INT(N/2)
7340 LET R=N-2*N1
7350 LET B$=STR$(R)+B$
7360 LET N=N1
7370 NEXT D
7380 RETURN
```

VERSION BBC/COMMODORE

Copiez la version Spectrum, avec les modifications suivantes :

1. Changez l'initialisation de LT et O$() comme dans la version Spectrum et ajoutez C$(5)=«B» à la ligne 1150.

2. La ligne 600 transfère le contrôle à la commande de sous-programme, voici donc comment modifier cela pour le Commodore 64 et le BBC Micro :

```
600 ON CM GOSUB 5000, 5500, 6000, 6500, 7000
```

3. Sur le BBC, remplacez la ligne 7060 par :

```
7060 N=?(P):NM=N
```

4. Sur le Commodore 64, remplacez la ligne 7350 par :

```
7350 B$=MID$(STR$(R),2)+B$
```

Du détail au gros

Tandy est devenu ces dernières années l'une des plus grosses chaînes de vente au détail du monde, et propose aussi bien des micro-ordinateurs que de multiples produits électroniques.

Tandy Corporation possède 392 centres informatiques et plus de 5 500 boutiques de vente au détail, dans 76 pays différents; l'ensemble de ce réseau constitue la chaîne Tandy and American Radio Shack. Par ailleurs, 29 de ses usines produisent des articles mis en vente sous ces deux noms.

Pourtant Tandy n'avait à l'origine rien à voir avec l'électronique. Fondée en 1927 par Norton Hinckley et David Tandy, la Hinckley-Tandy Leather Company avait en effet pour ambition de fournir du cuir aux cordonniers de Beaumont, dans le Texas. Ce n'est qu'en 1963 que Charles Tandy, le fils de David, eut l'idée de prendre des parts dans une petite entreprise de Boston, Radio Shack, qui connaissait alors des temps difficiles. Depuis les années vingt elle proposait aux radio-amateurs et aux passionnés d'électronique toutes sortes de composants électriques. Elle disposait de neuf points de vente dans la région de Boston, et réalisait la plus grande part du chiffre d'affaires grâce aux ventes par correspondance; mais elle était largement déficitaire. En quatre ans Charles Tandy parvint à passer de 4 millions de dollars de pertes à 20 millions de dollars de bénéfice.

L'étape suivante fut l'achat par Tandy d'une chaîne de grands magasins nommée Leonards; la compagnie s'introduisait ainsi dans le marché des composants électriques destinés au simple consommateur. C'est un domaine qui a été suivi au point qu'aujourd'hui le catalogue de Tandy comprend 396 articles relatifs à l'informatique, et 2 625 autres produits, qui vont des résistances aux appareils haute-fidélité, en passant par les synthétiseurs.

Tandy s'installa en Grande-Bretagne en 1973, et y devint rapidement un détaillant apprécié de composants électriques. En 1978, lors du lancement de son premier ordinateur, le TRS-80 Modèle I, il possédait 120 boutiques en Angleterre; ce chiffre était passé à 227 en 1983.

Le Modèle I fit aussitôt de Tandy l'un des plus importants constructeurs d'ordinateurs. C'était une machine équipée d'un microprocesseur Z-80, d'au moins 4 K de RAM et d'un écran noir et blanc pourvu d'un graphisme basse résolution. Elle pouvait être connectée à des lecteurs de disquettes, et ses utilisateurs pouvaient même y faire tourner le fameux système d'exploitation CP/M.

Depuis, Tandy s'est toujours efforcé de ne jamais se laisser distancer, sans pourtant retrouver la position dominante que lui avait valu cet appareil. La firme s'aventura ainsi sur le marché de l'informatique de gestion avec le Modèle II. Aujourd'hui, elle propose le Modèle 12 et le Modèle 16, qui sont tous deux des 16-bits, ainsi que le nouveau Modèle 2000, qui est un compatible IBM. Le Modèle III, version « supérieure » du Modèle I, peut être décrit comme un ordinateur domestique coûteux, ou comme un ordinateur de gestion bon marché. Tandy l'a remplacé par le Modèle 4 (et sa variante portable, le 4P). Ces deux derniers appareils, bien que destinés essentiellement aux milieux d'affaires, restent compatibles avec les modèles I et III — un fait trop rare pour n'être pas signalé! La série des

J. Beaufort, directeur de Tandy France

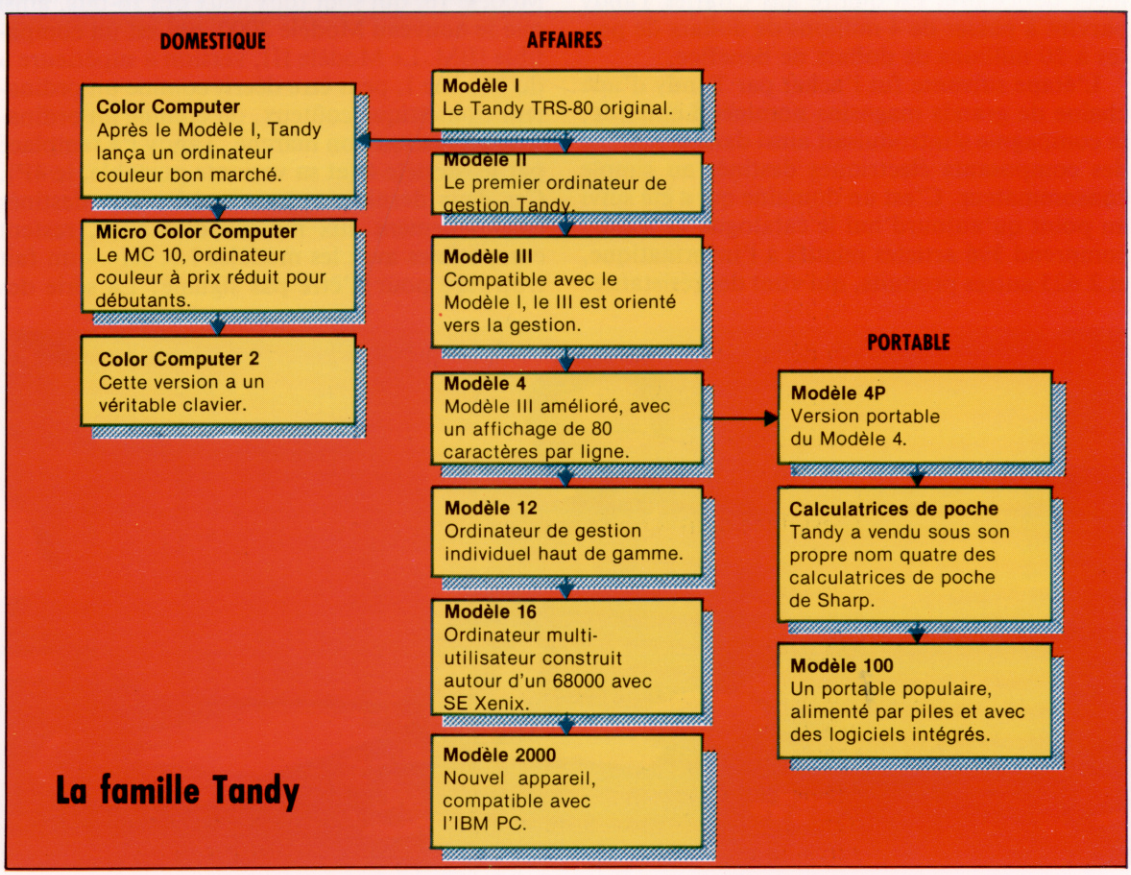
Être partout
Les boutiques Tandy constituent le point fort de la compagnie : un seul détaillant assure l'approvisionnement, les réparations et les conseils dans chaque ville. Aux États-Unis, ce réseau de vente est bien connu sous le nom de Radio Shack. (Cl. Ian McKinnell.)

TRS-80 dispose ainsi d'une très vaste gamme de logiciels.

Tandy a également tenté de revenir en force sur le marché de l'informatique individuelle avec le Color Computer, construit autour du microprocesseur 6809. Il s'est très bien vendu aux États-Unis. Une variante simplifiée et remise à jour, le Micro Color Computer, montre bien que la firme n'a pas renoncé à ses objectifs en ce domaine.

Elle fabrique aussi des ordinateurs portables, qui constituent sans doute la part la plus intéressante de sa production. Elle commença d'abord par vendre des calculatrices de poche basées sur les modèles de la gamme Sharp. Plus récemment, un contrat d'association avec une autre compagnie japonaise, Kyocera, et la célèbre firme Microsoft lui a permis de lancer le Modèle 100, alimenté par batteries. Il a la taille d'un livre, dispose d'un BASIC résident, d'un traitement de texte, d'un agenda et de logiciels de communication.

L'avenir paraît sourire à Tandy, qui s'apprête à sortir une gamme de nouveaux produits mis au point dans son centre de recherches de Fort Worth, au Texas (où elle s'est désormais installée), ainsi que dans sa filiale TC Electronics Corporation, basée à Tokyo. Tandy Corporation, avantagée à la fois par ses vastes capacités de production et son important réseau de détaillants, semble être idéalement placée pour tirer le meilleur parti de tous les progrès à venir de la micro-électronique, et tout indique qu'elle sera, à long terme, parmi les rares élus du marché de la micro-informatique.



PROGRAMME N° 19

LES TABLES

Dans l'écriture d'un programme, il n'est pas recommandé d'écrire toutes les variables qui appartiennent à une même fonction, pour ne pas occuper trop de place en mémoire. Pour cela, il faut faire appel aux tables, qui représentent un bloc de variables portant le « même nom ».

Ainsi, pour mémoriser trois quantités nous pourrions utiliser trois variables, Z1, Z2, Z3, en écrivant :

```
10 INPUT " VALEUR 1? "; Z1, par exemple 14
20 INPUT " VALEUR 2? "; Z2, par exemple  3
30 INPUT " VALEUR 3? "; Z3, par exemple 19
```

Utilisons plutôt une table que nous appellerons VA ( ). Ainsi, nous obtenons les trois variables représentées sous les noms VA (1), VA (2), VA (3).

```
VA (1)      14
VA (2)       3
VA (3)      19
```

Nous pourrions alors écrire les trois lignes du programme de la façon suivante :

```
10 INPUT " VALEUR 1? "; VA (1)
20 INPUT " VALEUR 2? "; VA (2)
30 INPUT " VALEUR 3? "; VA (3)
```

Mais utilisons plutôt une boucle FOR NEXT pour simplifier la saisie.

```
10 FOR I = 1 TO 3
20 INPUT " VALEUR? : "; VA (I)
30 NEXT I
```

Au départ, I est égal à 1, par conséquent :

```
20 INPUT " VALEUR? "; VA (I)
```

est équivalent à

```
20 INPUT " VALEUR? : " VA (1)
```

C'est donc dans VA (1) que la première note est introduite :

```
VA (1) = 14
VA (2) = 0
VA (3) = 0
```

Au second passage dans la boucle FOR NEXT, VA (I) est équivalent à VA (2) puisque I est égal à 2.

Même chose pour toutes les valeurs de I, de 1 à 3. Sans la boucle FOR... NEXT, on ferait de la manière suivante :

```
10 I = 1
20 INPUT " VALEUR : "; VA (I)
30 I = I + 1
40 IF I >3 THEN 60
50 GOTO 20
60 END
```

Éditons maintenant la table VA ( ).

```
5   REM SAISIE DE LA TABLE
10   FOR I = 1 TO 3
20   INPUT "VALEUR :";VA(I)
30   NEXT I
40   REM EDITION DE LA TABLE
50   PRINT : PRINT : PRINT
55   PRINT "LISTE DES VALEURS SAIS
      IES"
57   PRINT : PRINT : PRINT
60   FOR I = 1 TO 3
70   PRINT "VALEUR NO:";I;"
          ";VA(I)
80   NEXT I
110   END

]

]

]RUN
VALEUR :13
VALEUR :16
VALEUR :3


LISTE DES VALEURS SAISIES


VALEUR NO:1          13
VALEUR NO:2          16
VALEUR NO:3          3
```

L'INSTRUCTION « DIM »

L'instruction DIM sert à dimensionner les tables, c'est-à-dire à réserver en mémoire centrale des emplacements pour chaque élément de cette table.
Dès qu'une table dépasse 10 éléments, elle doit être dimensionnée.
Écriture : DIM (nombre d'éléments de la table).

Pour illustrer cela, faisons un petit programme après avoir dimensionner la table VA. En réservant 11 emplacements en mémoire centrale, on lit 11 notes en DATAS.
Ces 11 notes seront imprimées; ensuite, on effectuera un petit traitement : calcul du total et écriture de la moyenne de ces 11 notes.

```
]LIST

10   REM SAISIES DES VALEURS EN D
      ATAS
20   DATA 0,15,17,9,6,14,11,10,1,
      15,17
30   DIM VA(11)
35   REM LECTURE DES DATAS
40   FOR I = 1 TO 11
50   READ VA(I)
60   NEXT I
61   REM IMPRESSION DE LA TABLE
62   PRINT : PRINT : PRINT "LISTE
      DE LA TABLE DES VALEURS"
63   PRINT : PRINT : FOR I = 1 TO
      11
64   PRINT "VALEUR NO :";I,VA(I)
65   NEXT I
70   REM CALCUL DES TOTAL ET MOYE
     NNE
80  TT = 0
90   FOR I = 1 TO 11
95  TT = TT + VA(I)
100   NEXT I
110   PRINT : PRINT : PRINT
120   PRINT "TOTAL :";TT
130   PRINT : PRINT "MOYENNE :";TT
       / 11
140   END

]RUN

LISTE DE LA TABLE DES VALEURS

VALEUR NO :1     0
VALEUR NO :2     15
VALEUR NO :3     17
VALEUR NO :4     9
VALEUR NO :5     6
VALEUR NO :6     14
VALEUR NO :7     11
VALEUR NO :8     10
VALEUR NO :9     1
VALEUR NO :10    15
VALEUR NO :11    17

TOTAL :115

MOYENNE :10.4545455
```

ORGANIGRAMME