ABC INFORMATIQUE

N° 33

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Quels microprocesseurs ?

Table à dessiner

Gestion de stocks

Les portes logiques

EDITIONS ATLAS

M 6062-33-12 F 85 FB - 3,80 FS - $ 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° 33.

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 : Rudman-Réa.

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 : août 1984. 31848. 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

Compétition

Des millions de microprocesseurs sont utilisés dans le monde pour effectuer diverses tâches. Malgré la quantité et la diversité des applications, le marché est dominé par le Z80 et le 6502.

L'utilisation des puces dans les ordinateurs commença accidentellement. En 1972, le fabricant de composants électroniques Intel fut chargé par Datapoint de mettre au point une puce destinée à remplacer les TTL (logique transistor-transistor) que l'on trouvait dans les terminaux de l'époque. Le produit créé fut nommé le 8008. Il était en mesure de traiter des données de huit bits à la fois. Il aurait représenté une solution idéale de remplacement dans les terminaux s'il avait été plus rapide. Bien que Datapoint décidât de ne pas l'utiliser, le potentiel du 8008 comme unité centrale d'un ordinateur d'usage général fut bientôt reconnu par des ingénieurs et par des amateurs. C'est ainsi qu'allait naître l'ordinateur individuel.

Cependant, les restrictions du 8008 sur le plan de la vitesse et de la puissance devinrent rapidement évidentes et Intel entreprit la conception d'une autre puce plus performante. La puce créée alors, le 8080, s'imposa rapidement sur le marché.

Quand Intel introduisit le 8080, le concurrent, Motorola, lança sur le marché un microprocesseur à huit bits nommé le 6800. Les philosophies de conception du 8080 et du 6800 diffèrent considérablement, mais ils sont tous deux très puissants et constituent l'un comme l'autre une excellente base dans la conception d'un ordinateur.

Bien que le 8080 et le 6800 fussent d'efficacité équivalente, un accident de parcours eut pour conséquence qu'une troisième puce, le Z80, remporta un succès phénoménal. En 1974, Gary Kildall, actuellement président de Digital Research, créa pour Intel un système d'exploitation de disquettes nommé CP/M. Cela permit aux ordinateurs construits autour d'un microprocesseur 8080 d'utiliser les lecteurs de disquettes Shugart qui venaient d'être introduits. Le système d'exploitation de Kildall fut refusé par Intel, qui pensait que le logiciel déjà existant satisfaisait les besoins des systèmes informatiques du moment.

Cependant, les petits ordinateurs devinrent de plus en plus populaires et le CP/M facilita grandement la gestion des fichiers sur ces systèmes. C'est ainsi que le 8080 s'imposa sur le marché pendant plusieurs années et que le Motorola 6800 fut relativement négligé. Diverses tentatives furent entreprises pour concevoir des systèmes d'exploitation de disquettes pour le 6800, mais le 8080 attirait davantage les programmeurs que le 6800.

Un choix vital
La plupart des micros se partagent entre une unité centrale 6502 ou Z80. D'autres, moins nombreux, utilisent le 6809 par exemple.
(Cl. Thomson/ SIMIV/Matra.)

Une idée de génie permit la conception d'une nouvelle puce — le Z80. Zilog, une équipe d'ingénieurs qui avait déjà travaillé sur le 8080 pour Intel, constata que le jeu d'instructions pouvait être étendu. En d'autres termes, certaines combinaisons possibles de zéro et de un pouvant être reconnues comme instructions par le 8080 n'avaient pas été exploitées. En utilisant des combinaisons binaires non employées par la puce Intel, Zilog fut en mesure de concevoir un microprocesseur fonctionnant de façon identique au 8080 en lui donnant le jeu d'instructions du 8080, et pouvant offrir une amélioration considérable au niveau des performances. Ils purent ainsi créer une puce qui utilisait les logiciels écrits pour le 8080.

En plus de cette innovation, Zilog offrit un autre avantage commercial important. Alors que la puce Intel exigeait la présence d'un générateur d'horloge spécial et d'un contrôleur de système, l'équipe Zilog réussit à combiner sur

Tableau d'évolution des microprocesseurs
L'évolution technologique des microprocesseurs s'est opérée à partir de deux sources principales : ceux dérivés des microprocesseurs Intel et ceux provenant de la puce rivale Motorola 6800. Ce tableau illustre comment les puces ont été développées et énumère certaines des machines où elles ont été utilisées. De nombreuses puces moins connues apparaissent dans des micros moins populaires. L'Apple III est peut-être la seule machine de gestion qui utilise un processeur 6502. L'Olivetti M20 est le seul micro à usage général à avoir un Z8000. Dans les deux cas, ce choix inusité de microprocesseur et par conséquent le manque de logiciels ont interdit le succès de la machine.

'71
4004
8008
4004 et 8008 : conçu pour remplacer un grand nombre de circuits intégrés TTL, le 4004 était une puce très simple qui pouvait gérer les données par groupes de 4 bits. Intel est passé rapidement à un traitement à 8 bits avec le 8008. Les ingénieurs et les amateurs en reconnurent rapidement le potentiel et commencèrent à construire leurs propres ordinateurs individuels.

'72
'73
'74
6800
6800 : le rival du 8080, avec un potentiel similaire mais issu d'une philosophie de conception complètement différente. Il existe donc deux écoles : ceux qui préfèrent l'approche de l'Intel 8080 et ceux qui préfèrent le mode de travail du 6800 de Motorola.
8080

'75
6502
8085
Z80
8080 : avec l'arrivée du système d'exploitation CP/M, elle devint la première puce à être utilisée pour construire des ordinateurs domestiques et de gestion. Intel réalisa une version améliorée du 8080 et la nomma 8085 ; cette unité était compatible avec le 8080.
Z80 : une équipe de concepteurs quitta Intel pour former Zilog et pour produire cette version améliorée et compatible du 8080.

'76


'77
6809
6809 : la version améliorée du 6800 par Motorola est le 6809, peut-être la plus performante des puces 8 bits. Cependant, son arrivée fut trop tardive pour avoir un impact réel.
6502 : MOS Technology a conçu sa propre puce 8 bits, qui bien que non compatible avec le 6800 en est pourtant très proche. Son faible coût l'a rendue très populaire auprès des amateurs et des concepteurs ; elle fut utilisée dans la première génération d'ordinateurs individuels comme le PET et l'Apple. Elle demeure un choix très populaire pour les micros domestiques.

'78


'79


'80


'81


68000
8086
Z8000
8088
68000 : le succès de cette puce 16 bits de haute qualité a été plutôt limité en raison du peu de logiciels bon marché et de la forte présence du 8088. Sinclair a choisi la version simplifiée 68008 pour son QL.
8088 et 8086 : le 8088 est une version simplifiée qui peut utiliser d'anciennes puces et qui pendant un certain temps fut la puce 16 bits la plus populaire avec une utilisation dans l'IBM PC et dans le Sirius. Le 8086, avec de meilleures performances et une compatibilité complète, est maintenant utilisée dans la plupart des machines.
Z8000 : la première puce 16 bits de Zilog n'a pas réussi à s'imposer sur le marché des ordinateurs d'utilisation générale. Une mauvaise synchronisation et l'adoption du 8088 par IBM expliquent en partie cet échec.

'82
'83
'84
68008



une seule puce toute la logique requise par un ordinateur. Même si la production de cette puce était relativement coûteuse, le fait qu'elle pouvait remplacer plusieurs autres puces la rendit très populaire auprès des fabricants.

Même si le 6800 eut beaucoup moins de succès que le 8080, il fut néanmoins choisi par certains concepteurs et programmeurs. Motorola conçut plus tard un microprocesseur à huit bits très sophistiqué, le 6809, qui améliora le 6800. Malheureusement, avant que le 6809 attaque le marché, une société rivale, MOS Technology introduisait une autre version améliorée du 6800, nommée le 6502. Il s'agit du microprocesseur le plus populaire de l'importante série 6500. Toutes les unités de cette série utilisent le même jeu d'instructions, mais elles diffèrent au niveau de leur puissance et de leurs possibilités.

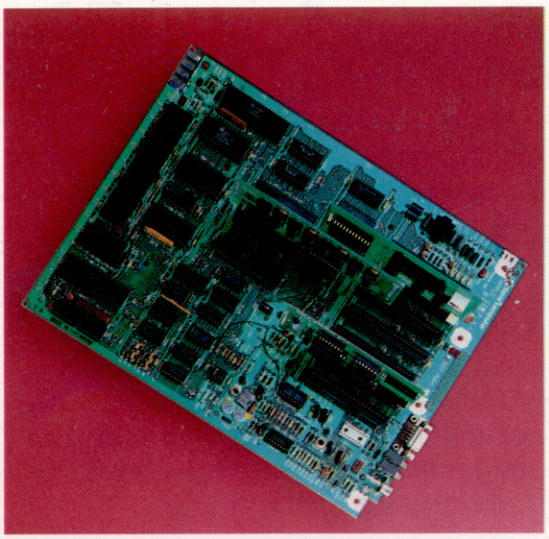
Le 6502 de MOS Technology obéit à une philosophie de conception très proche de celle du 6800 de Motorola, mais n'est pas compatible avec le 6800, au niveau autant du matériel que du logiciel. Le Z80, par contre, intègre le jeu complet des instructions du 8080, et peut le remplacer dans un système informatique. Certaines modifications doivent cependant être apportées au niveau de la conception du matériel.

Le 6502 offre un jeu complet d'instructions bien connu par tout programmeur de 6800, avec des possibilités accrues et des exigences d'interface légèrement plus faciles à satisfaire. Mais il ne permet aucune compatibilité logiciel ni la possibilité d'effectuer un remplacement puce pour puce. Il était donc difficile de prévoir le succès qu'aurait ce composant, mais il eut une chance unique : le 6502 fut utilisé dans l'ordinateur Apple, l'une des plus belles réussites commerciales de l'industrie de la micro-informatique.

Lors de l'introduction de l'Apple, les ordinateurs individuels incorporaient pour la plupart des systèmes de bus S-100. Ceux-ci nécessitaient la présence d'une « carte mère » pour transmettre l'alimentation et les signaux vers une carte distincte pour chaque fonction. Un système S-100 minimal devait donc comporter une alimentation, une carte mère, une carte UC, une carte mémoire, une carte vidéo et probablement une carte imprimante et une carte de lecteur de disquette. Il est par conséquent facile de voir à quel point un système S-100 est plus coûteux qu'un système monocarte comme l'Apple.

En plus du coût de revient assez bas, les principaux atouts de Steve Wozniak et de l'équipe d'Apple résidaient dans un programme d'application nommé VisiCalc. Ce programme remporta beaucoup de succès auprès des hommes d'affaires qui virent en lui un outil puissant pour créer des analyses financières beaucoup plus facilement qu'avec une calculatrice, un crayon et du papier. VisiCalc eut un tel succès qu'il entraîna des ventes massives d'ordinateurs Apple; c'est ainsi que le 6502 devint l'un des microprocesseurs les plus utilisés. Commodore retint le 6502 dans le PET et ses successeurs.

Pendant que le 6502 confirmait sa prépondérance sur le marché des ordinateurs à huit bits, les ordinateurs à seize bits firent leur apparition. Intel présenta le 8088 et le 8086 pour ces derniers types d'ordinateurs, alors que Motorola produisit le 68000 et Zilog conçut le Z8000. Ces trois conceptions à seize bits offrent chacune leurs avantages, mais aucune n'est compatible avec leurs prédécesseurs à huit bits. Heureusement pour Intel, Digital Research et Microsoft ne tardèrent pas à créer des systèmes d'exploitation pour le 8086/8088 (CP/M-86 et MS-DOS respectivement), alors que Zilog et Motorola ne bénéficièrent pas du même enthousiasme de la part des créateurs de logiciels. L'adoption par IBM du 8088 dans son ordinateur individuel confirma le succès de la puce Intel.



Cherchez la puce... L'intégration de plus en plus grande des circuits électroniques réduit le nombre de puces sur une carte. Quand Apple développa son Apple II, la nouvelle version IIe possédait moitié moins de puces.

La bataille pour la conquête du marché des puces seize bits risque d'être une répétition de ce qui s'était produit dans le cas des puces huit bits. Le 8086 d'Intel (et la version simplifiée, le 8088) sont devenus des standards comme l'étaient le Z80 et le 6502. Ce succès s'explique principalement par la compatibilité logiciel offerte par les systèmes d'exploitation MS-DOS et CP/M-86 et par leur sélection dans les micros les mieux vendus, notamment l'IBM et le Sirius. La puce Z8000 de Zilog n'a été utilisée que dans un seul ordinateur d'usage général — l'Olivetti M20. Olivetti s'est efforcé de créer des logiciels pour cette machine, et a finalement introduit une carte enfichable munie d'un 8086 pour permettre l'exécution des logiciels MS-DOS et CP/M-86. Depuis, Zilog a entrepris la conception d'une nouvelle puce, le Z800, qui en plus d'être un seize-bits peut également exécuter les programmes fondés sur le processeur Z80.

Malgré la popularité des seize-bits, la plupart des ordinateurs vendus sont toujours conçus à partir du Z80 ou du 6502. Les ordinateurs seize bits offrent indiscutablement des avantages au niveau de la vitesse et de la puissance d'exécution sur leurs prédécesseurs, mais les machines huit bits risquent d'occuper le marché pendant encore un bon moment en raison du vaste choix de logiciels dont elles disposent.

Demi-mesure

Des circuits intégrés simples se substituent aux transistors dans les micro-ordinateurs d'aujourd'hui. Après avoir construit des portes ET, OU et NON à l'aide de transistors, nous nous servons de circuits intégrés pour mettre au point un demi-additionneur.

Les portes logiques que nous avons déjà réalisées forment la base de circuits numériques bien plus complexes. L'un d'eux est le demi-additionneur, dont nous avons déjà parlé dans notre cours d'informatique. Il sert à additionner deux bits isolés. Il accepte deux entrées (les bits) et produit deux sorties : la somme et le bit de retenue. La table de vérité qui permet de représenter l'opération a l'aspect suivant :

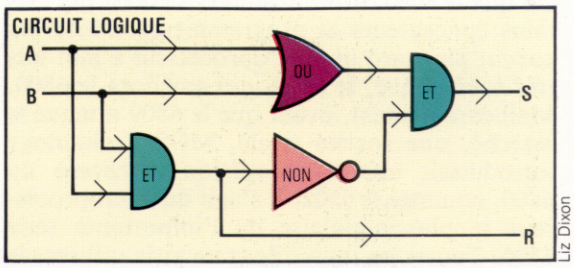
A	B	S	R
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

A la sortie, la somme est constituée par l'addition des deux bits d'entrée. Lorsque tous deux sont égaux à 1, elle est donc de 10 en binaire. C'est un résultat qui ne peut être figuré par un seul bit de sortie; il se produit un dépassement de capacité, qui est transmis à un second bit, qu'on appelle bit de retenue.

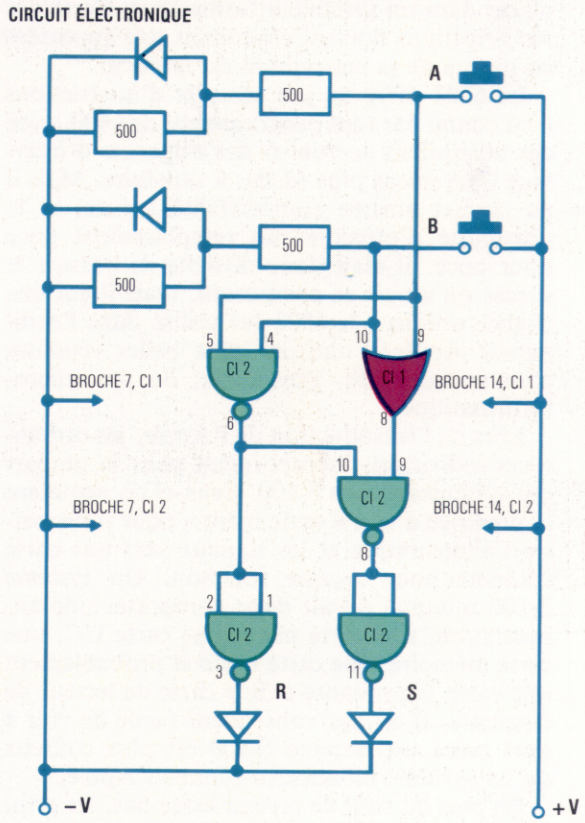
Un demi-additionneur n'a qu'une utilité limitée au sein d'un ordinateur à huit bits, qui a besoin d'un circuit capable d'additionner deux mots de huit bits. Un tel circuit peut être construit à l'aide de seize demi-additionneurs. Les deux premiers bits sont additionnés à l'aide du premier de ces seize dispositifs; leur somme formera le premier bit du résultat. Le bit de retenue est ajouté au résultat de la deuxième somme, dont le bit de retenue est joint à la troisième, et ainsi de suite; c'est une opération « en chaîne ».

Un simple demi-additionneur nécessiterait l'emploi de dix transistors dans les portes que nous avons construites. Heureusement ET, NON-ET, OU, NON-OU et bien d'autres sont disponibles sous forme de circuits intégrés bon marché, par groupes de quatre. Il suffit d'en rassembler quelques-uns pour réaliser sans difficultés notre demi-additionneur.

Le circuit logique correspondant est présenté ci-contre. C'est là la forme la plus simple de l'ensemble, qui n'a recours qu'à trois portes logiques : OU, ET et NON. Comme les circuits dont nous serons appelés à nous servir ne



contiennent qu'un seul type de porte, le circuit logique lui-même a été schématisé afin de ne nécessiter qu'un assortiment restreint : celui que nous construirons comprend quatre portes NON-ET et une seule porte OU. Le nombre de circuits intégrés est ainsi réduit à deux. Mais le dispositif est quand même plus complexe que



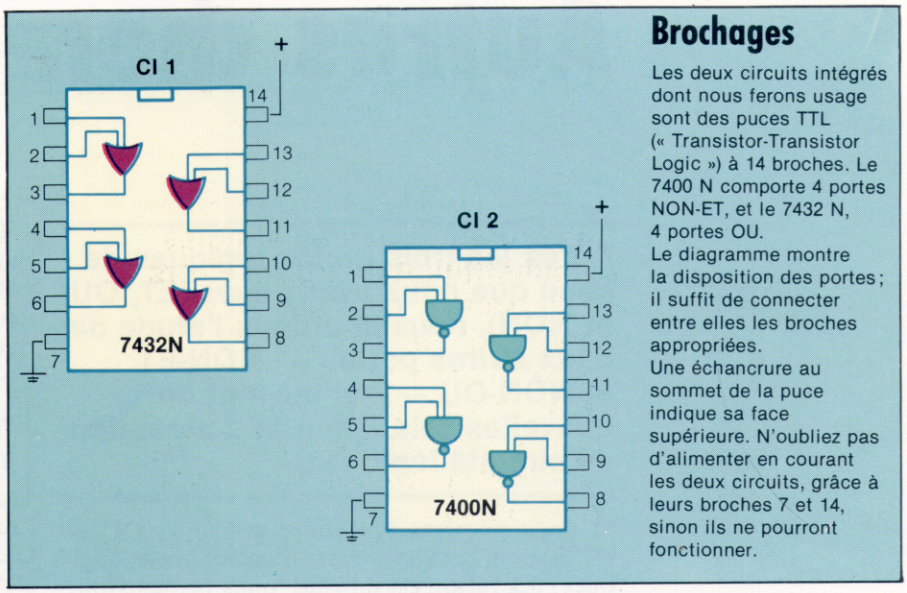
ceux que nous avons déjà abordés, et il faudra s'assurer que tous les composants sont bien à leur place sur la plaquette d'essai.

Une fois que vous en serez venu à bout, vous penserez peut-être qu'il vous a fallu bien des efforts pour parvenir à des résultats très minces. La tâche est certes plus aisée qu'avec des composants discrets tels que les transistors; mais il est difficile d'imaginer qu'un ordinateur puisse être entièrement réalisé de cette façon.

De fait, les puces capables par exemple d'additionner deux nombres de quatre bits.

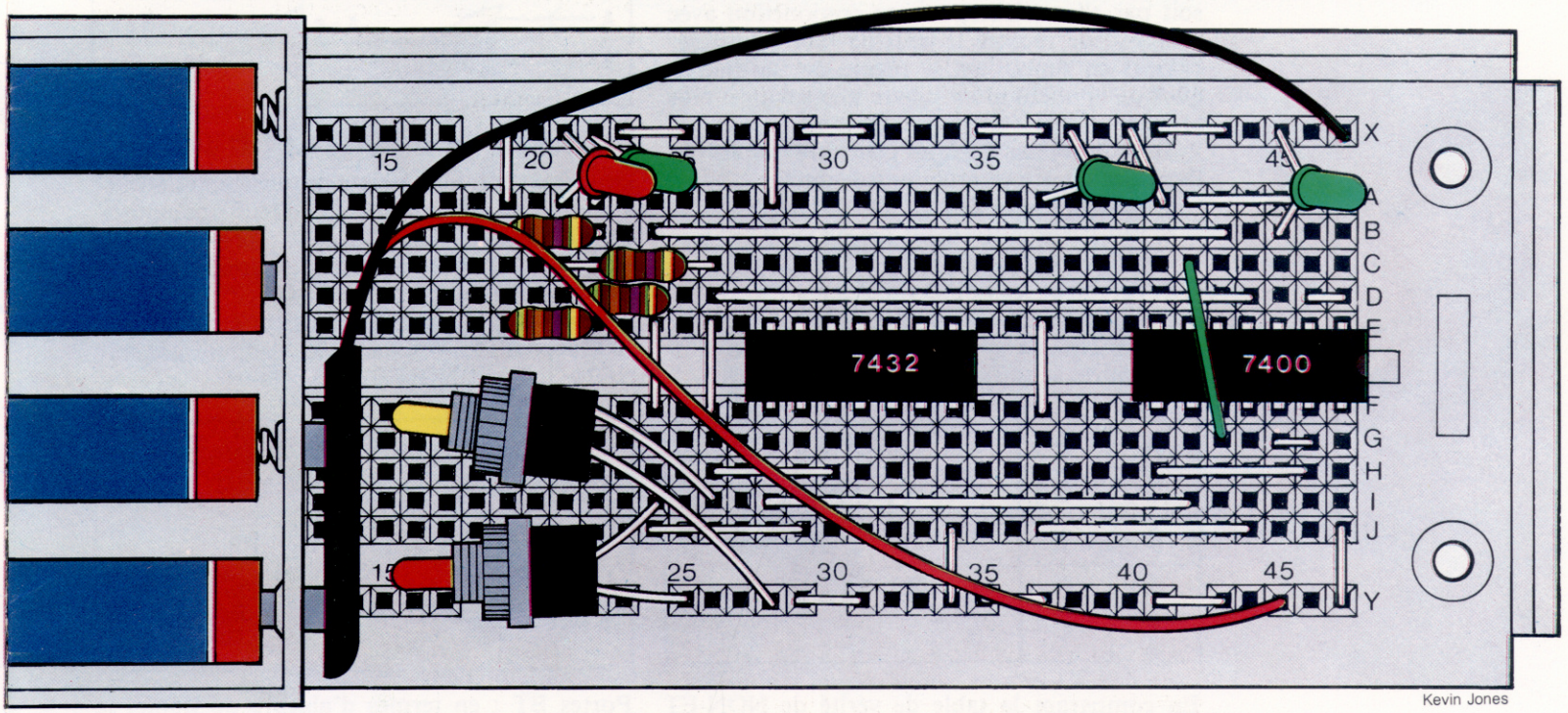
Ce dont vous aurez besoin

4 résistances de 500 ohms, un quart de watt
4 DEL
2 commutateurs à bouton-poussoir
1 circuit intégré 7400 N
1 circuit intégré 7432 N
4 piles HP7 ou équivalent
1 boîtier batterie
1 prise batterie
1 plaquette d'essai (Experimentator 300 ou similaire)
Un peu de fil électrique



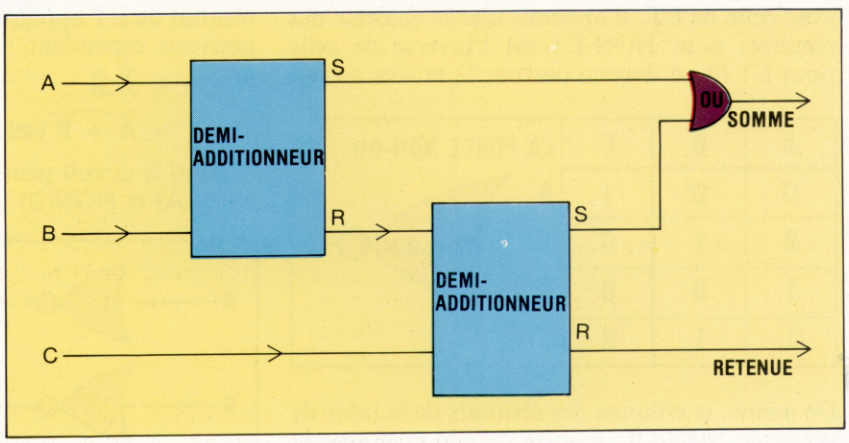
Brochages

Les deux circuits intégrés dont nous ferons usage sont des puces TTL (« Transistor-Transistor Logic ») à 14 broches. Le 7400 N comporte 4 portes NON-ET, et le 7432 N, 4 portes OU.
Le diagramme montre la disposition des portes ; il suffit de connecter entre elles les broches appropriées.
Une échancrure au sommet de la puce indique sa face supérieure. N'oubliez pas d'alimenter en courant les deux circuits, grâce à leurs broches 7 et 14, sinon ils ne pourront fonctionner.



Kevin Jones

Le circuit
Une fois le circuit électronique conçu, il faut disposer l'ensemble des composants sur une plaquette d'essai. On peut soit faire l'acquisition de papier millimétré, soit utiliser simplement la photocopie d'une plaquette d'essai vide ! Il faut également s'efforcer de réaliser un circuit aussi proche que possible du plan originel ; plus celui-ci sera clair, plus il sera facile à construire. Reproduisez fidèlement le schéma qui vous est donné : tous les composants sont à leur place exacte.



Additionneur complet
A titre d'exercice, vous voudrez peut-être faire de votre demi-additionneur un additionneur complet. Ce dernier ne se borne pas à additionner deux bits ensemble, mais il y ajoute une retenue provenant de n'importe quelle position de bit antérieure. Plusieurs additionneurs complets peuvent additionner des mots binaires entiers. La façon la plus simple d'en construire consiste à jumeler deux demi-additionneurs, comme cela vous est montré ci-contre.

Autres chemins

Après les trois portes logiques de base que nous avons vues (ET, OU et NON), nous abordons l'étude de deux autres portes — NON-ET et NON-OU — qui ouvrent de nouvelles voies pour la conception de circuits logiques.

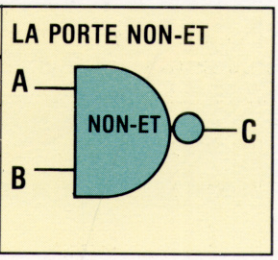
Si nous pouvons tout faire avec ET, OU et NON, pourquoi apprendre d'autres portes logiques? La raison est qu'elles nous permettront, soit par elles-mêmes, soit en conjonction avec d'autres, de réduire le coût de fabrication des circuits en simplifiant le réseau de fils électriques ou en mettant en œuvre une solution plus efficace.

Tout problème logique peut être résolu par l'une des quatre techniques suivantes :

a) utiliser ensemble ET, OU et NON;
b) utiliser exclusivement des NON-ET;
c) utiliser uniquement des NON-OU;
d) combiner les points précédents.

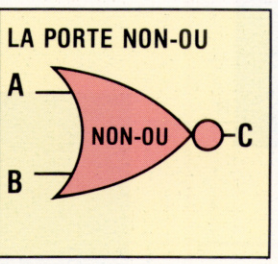
Étudions ces nouvelles portes logiques. Comme pour les circuits et leurs éléments, la fonction d'une porte se traduit par sa table de vérité.

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0



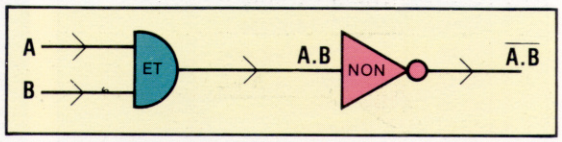
En comparant la table de vérité de NON-ET avec celle de ET, il apparaît que la colonne des résultats pour NON-ET est l'inverse de celle pour ET (0 est devenu partout 1, et vice versa).

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

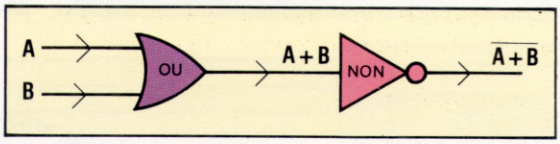


De même, la colonne des résultats de la table de vérité de NON-OU montre qu'elle est l'inverse de celle pour OU (il suffit aussi de remarquer que les zéros et les uns sont également intervertis dans le tableau).

Il n'y a pas de symboles booléens particuliers pour les opérations NON-ET et NON-OU, mais nous pouvons représenter chaque fonction par des symboles ET, OU et NON. Une porte logique NON-ET revient à ce simple circuit :



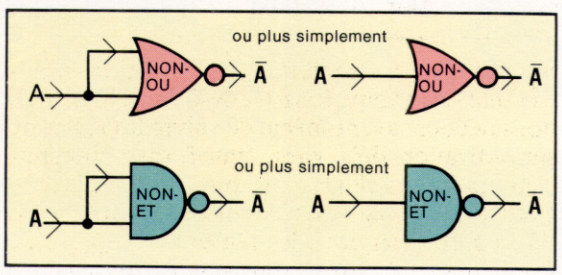
La porte logique NON-OU équivaut à une porte OU suivie d'une porte NON :



Utilisation de NON-ET et NON-OU

De même qu'il est possible de représenter NON-ET et NON-OU en tant que circuits composés de ET, OU et NON, on peut représenter ET, OU et NON sous la forme de circuits faisant intervenir NON-ET ou NON-OU.

Portes NON : la négation peut être obtenue en réunissant les deux entrées et en utilisant soit NON-OU, soit NON-ET.

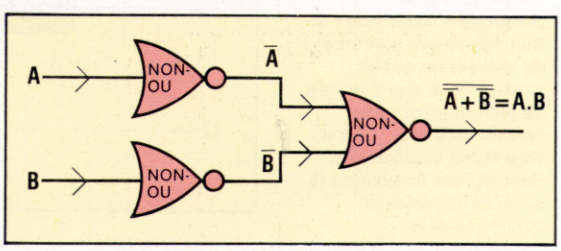


Portes ET : en termes d'algèbre de Boole, le résultat de ET appliqué à A et B est A.B. Nous pouvons cependant tourner cette expression :

$A.B = \overline{\overline{A}.\overline{B}}$ (puisque $A = \overline{\overline{A}}$)

$= \overline{\overline{A} + \overline{B}}$ (selon la loi de Morgan)

Ainsi le circuit peut être établi en « passant » NON(A) et NON(B) par la porte NON-OU :

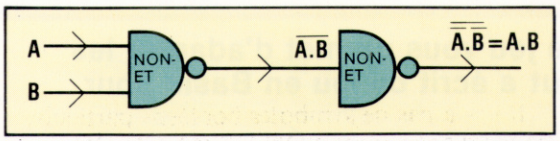


Il est également possible de créer une porte ET en utilisant NON-ET. Le résultat sera $\overline{A.B}$.

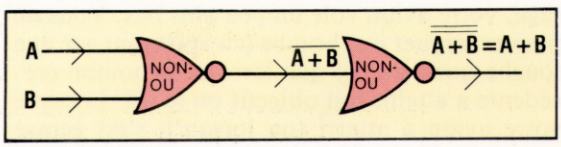
En appliquant une négation, nous obtenons :

$$\overline{\overline{A.B}} = A.B$$

Le circuit sera donc :



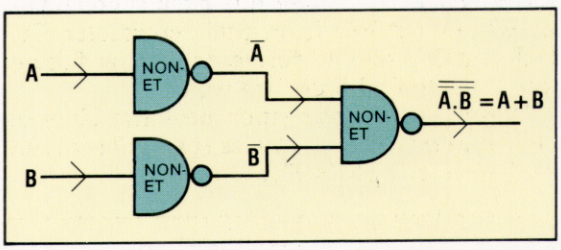
Portes OU : de la même manière qu'en chaînant deux portes NON-ET on obtient une porte ET, deux portes NON-OU qui se suivent donneront un circuit équivalent à une porte OU :



Le résultat attendu d'une porte OU est $A+B$. Cela peut être exprimé en algèbre de Boole sous une forme faisant intervenir NON-ET :

$$A + B = \overline{\overline{A}+\overline{B}}$$
$$= \overline{\overline{A}.\overline{B}}$$

Le circuit correspondant utilisant des portes NON-ET sera :



Si nous désirons construire un circuit qui n'utilise que des expressions NON-ET et NON-OU, nous devons, avant même de suivre les règles de simplification déjà vues, transformer l'expression booléenne finale sous une forme utilisable. Pour les circuits comprenant des portes NON-ET, nous appliquons les règles de l'algèbre de Boole pour créer une expression regroupant des fonctions ET reliées par des fonctions OU.

Nous appliquons ensuite plusieurs fois le théorème de Morgan pour aboutir à une expression entièrement sous la forme d'opérateurs NON-ET. Pour des circuits sous la forme de portes NON-OU, nous suivrons la même démarche, ainsi que le montre l'exemple qui suit. Pour illustrer la manière dont ces règles sont appliquées, voyons à nouveau la porte OU *exclusif* (XOR). Le résultat provenant de cette porte se définit : $C = \overline{A.B}.(A+B)$.

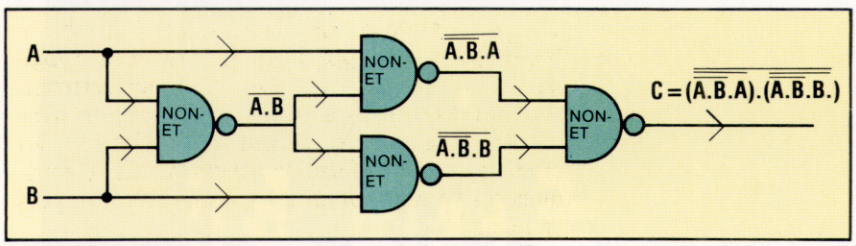
Prenons cette expression et transformons-la de sorte qu'un circuit pour la porte XOR soit uniquement fait de portes NON-ET.

$$C = \overline{A.B}.(A+B)$$
$$= (\overline{A.B}.A)+(\overline{A.B}.B)$$

(distributivité des parenthèses)

$$= \overline{\overline{(\overline{A.B}.A)}.\overline{(\overline{A.B}.B)}}$$ (loi de Morgan)

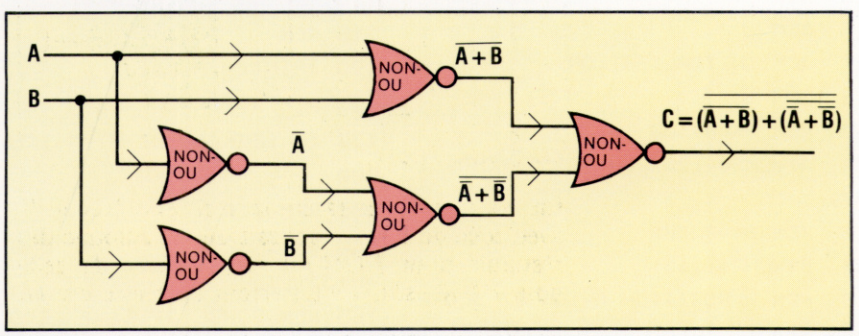
Lorsqu'on crée le circuit d'une expression aussi complexe que celle ci-dessus, il vaut mieux partir du résultat et remonter jusqu'aux données.



Pour NON-OU, nous devons à nouveau commencer par l'expression originelle simplifiée de XOR, et la transformer en groupe de fonctions OU reliées par des opérateurs ET. Cela s'obtient par la loi de Morgan appliquée à la partie gauche de l'expression :

$$C = \overline{A.B}.(A+B)$$
$$= (\overline{A}+\overline{B}).(A+B)$$
$$= \overline{\overline{(\overline{A}+\overline{B})}+\overline{(\overline{A}+\overline{B})}}$$

A nouveau, le circuit dérivé de cette expression s'obtiendra en partant de la fin de l'expression :



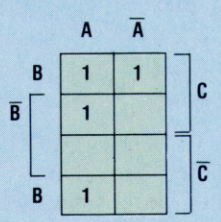
Réponses à l'exercice 6

1a)

Données			Résultat
A	B	C	P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

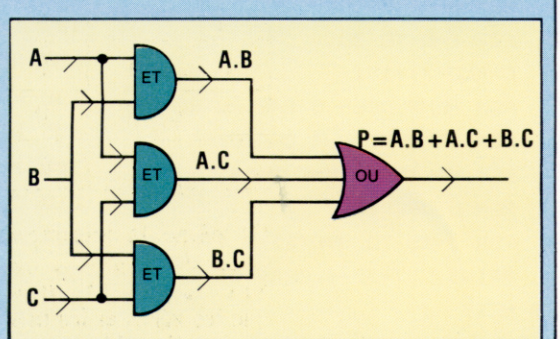
b) L'expression booléenne de P est : $P = A.B.C + A.\overline{B}.C + A.B.\overline{C} + A.\overline{B}.\overline{C}$

... que l'on simplifie ainsi avec la table de Karnaugh :



Soit $P = A.B + A.C + B.C$

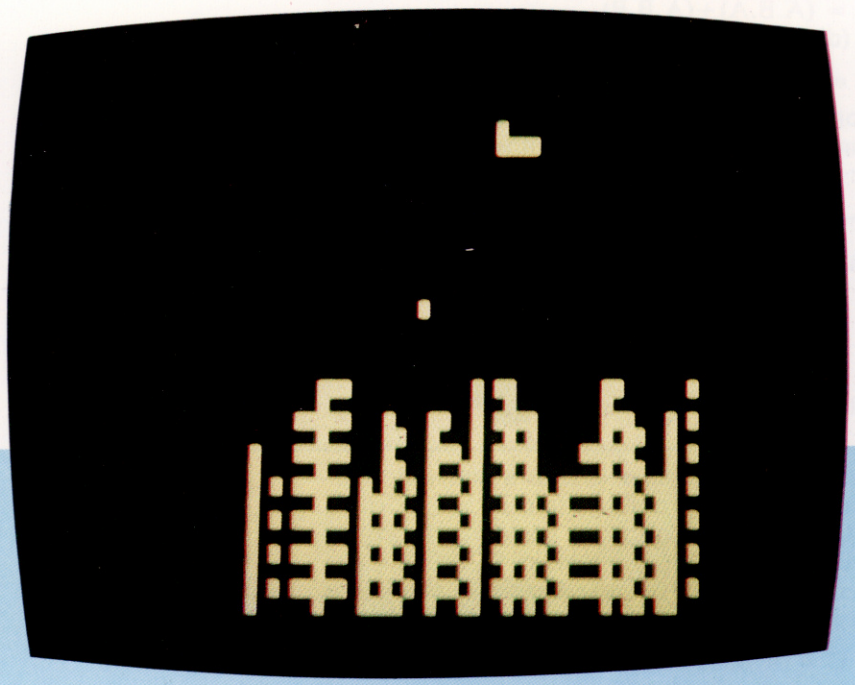
c) Le circuit est le suivant :



Liz Dixon

Blitz

Combat, adresse, le programme de ce jeu vous permet d'adapter les difficultés à vos désirs. Pierre Monsaut a écrit ce jeu en Basic pour l'ordinateur ALICE de Matra.



Votre mission est de détruire la ville que vous survolez afin de pouvoir atterrir. A chaque passage, votre avion vole un peu plus bas. Vous ne pouvez larguer une bombe (en appuyant sur une touche quelconque) que lorsque la bombe précédente a atteint son objectif ou le sol. Lorsque votre avion a atterri (ou lorsqu'il s'est écrasé contre un immeuble), le score est affiché ainsi que le record du jour. Si ce jeu vous paraît trop difficile, vous pouvez changer les limites de la ville (6 et 26, ligne 60) et la hauteur maximale des immeubles (en remplaçant 8, ligne 80 par une valeur supérieure, par exemple 10).

```
5  REM  *********
10  REM * BLITZ *
15  REM *********
16  REM
17  REM INITIALISATION
18  REM
19  REM A$=AVION
20  A$=CHR$(128)+CHR$(155)+CHR$(1
47)
25  REM B$=BOMBE
30  B$=CHR$(145)
35  REM H=POSITION AVION ET SCORE
40  H=0
50  CLS 0
55  REM AFFICHAGE VILLE
60  FOR I=6 TO 26
70  C=RND(7)+151
75  REM REMPLACER 8 PAR AUTRE
76  REM VALEUR POUR CHANGER
77  REM HAUTEUR IMMEUBLES
78  REM LIGNE 80
80  FOR J=15 TO RND(4)+8 STEP-1
90  PRINT@ J*32+I,CHR$(C);
100 NEXT  J
110 NEXT  I
114 REM
115 REM BOUCLE PRINCIPALE
116 REM
120 PRINT@ H,A$;
125 REM AVION ECRASE?
130 IF PEEK(16387+H) <> 128 THEN 1
000
155 REM TIR
160 IF INKEY$ <> "" AND B=0 THEN B
=H+33
165 REM BOMBE ATTEINT LE SOL?
170 IF B<> 0 THEN GOSUB 5000
175 REM PAS DE BOMBE LACHEE?
180 IF B=0 THEN GOSUB 6000
185 REM AVANCE AVION
190 H=H+1
195 REM AVION POSE?
200 IF  H=507 THEN 1000
210 GOTO 120
994 REM
995 REM AVION POSE
996 REM
999 REM RECORD BATTU?
1000 IF H>R THEN R=H
1010 PRINT@ 3,"SCORE :";H,
1020 PRINT "RECORD :";R;
1030 FOR I=1 TO 100
1040 NEXT I
1050 R$=INKEY$
1060 PRINT@ 73,"UNE AUTRE ?";
1070 R$=INKEY$
1080 IF R$="" THEN 1070
1090 IF R$=<>"N" THEN 20
1100 CLS
1110 END
4994 REM
4995 REM BOMBE LACHEE
4999 REM SOL ATTEINT
5000 IF B>=510 THEN B=0:GOTO 502
0
5005 REM IMMEUBLE ATTEINT?
5010 IF PEEK(16384+B)<>128 THEN
PRINT@ B,CHR$(128);
5015 REM AFFICHAGE BOMBE
5020 PRINT@ B1,CHR$(128);
5030 IF B<>0 THEN PRINT B,B$;:B
1=B:B=B+32
5040 RETURN
5994 REM
5995 REM DELAI
5996 REM
6000 FOR I=1 TO 20
6010 NEXT I
6020 RETURN
```

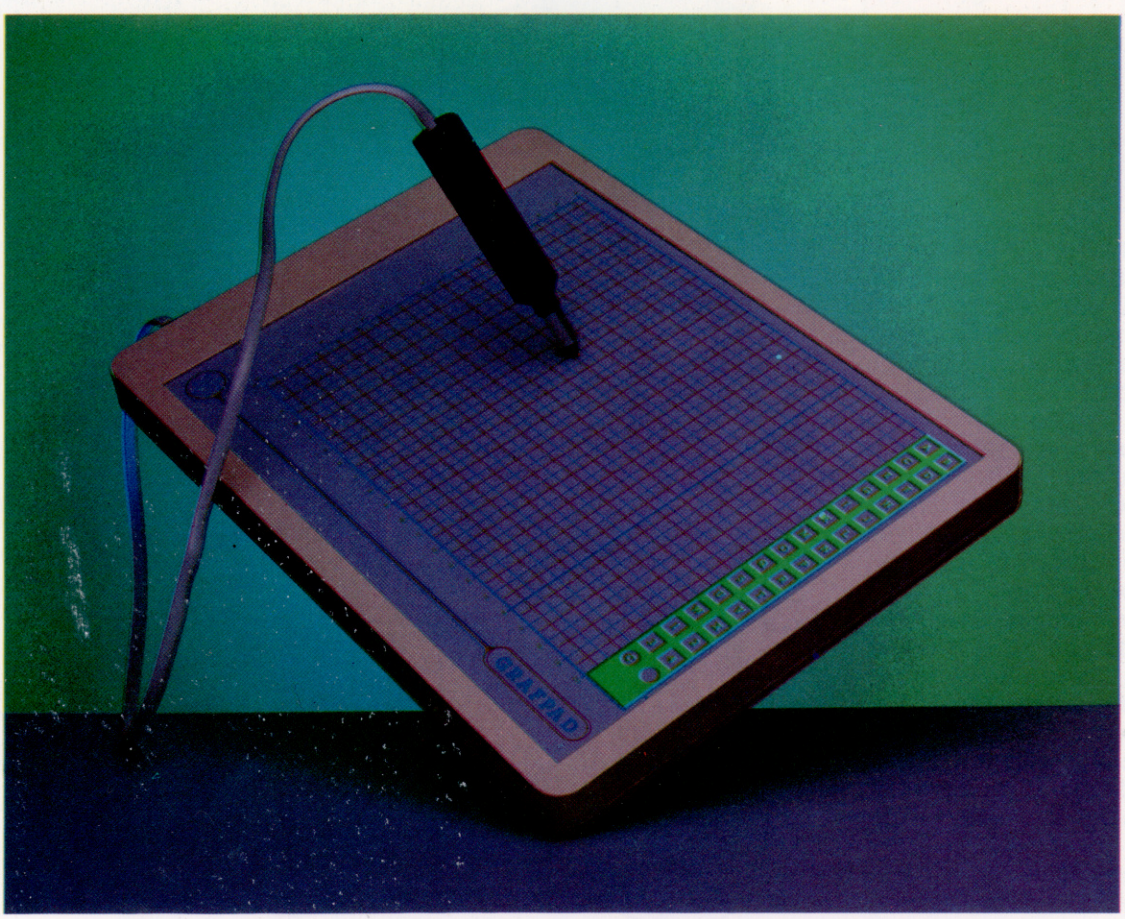
Table à gribouiller

Grafpad est une table qui trace des schémas détaillés et des relevés graphiques à partir des données numériques fournies par ordinateur. Son prix est attractif pour les utilisateurs non professionnels.

Ces tables constituent un des périphériques les plus utiles et polyvalents pour un micro. Les utilisations les plus évidentes sont les dessins et les graphiques-aides à la conception assistée par ordinateur. Les applications vont du dessin artistique à main levée jusqu'à la conception de circuits électroniques et le tracé de cartes topographiques. Mais en dehors de ces applications de dessin immédiat et interactif, les tables numériques constituent un moyen précieux de saisie des données. Ainsi la disposition d'une carte électronique à créer peut prendre sur la table numérique la forme d'un programme soit avec des mots, soit avec des dessins. Vous devez alors effleurer avec le photostyle ou crayon lumineux une des commandes représentées sur la table : le logiciel exécutera la fonction que vous aurez ainsi choisie.

Grafpad est constitué de trois éléments : la table elle-même, un photostyle relié par un câble souple et le logiciel de commande. La table est reliée au micro *via* le port utilisateur et le photostyle se branche sur le côté. La table représente une grille de 16 × 20 cases. Sur un bord sont les commandes sous la forme de lettres formant un bloc. Le bloc des commandes permet d'utiliser les commandes sans passer par le clavier du micro-ordinateur. La table est recouverte d'une pellicule protectrice de Plexiglas. Vous avez la possibilité de créer vos propres grilles ainsi que vos propres commandes.

Sous le plateau de la table à numériser, existe un fin réseau, sous forme de grille, de 320 × 256 fils électriques, distants d'environ 1,2 mm les uns des autres. La pointe du photostyle avec lequel vous dessinez est en réalité un interrupteur minuscule. Lorsque vous appuyez le photostyle sur la pellicule de Plexiglas qui recouvre la table, un composant dit « réseau de portes programmables » (ULA) transmet des impul-

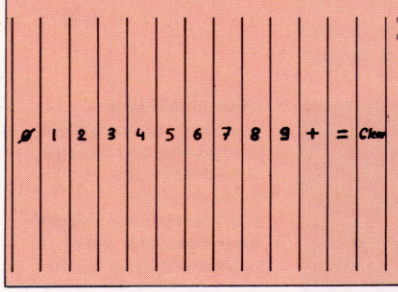


Idées graphiques
Grafpad peut être utilisé avec son propre logiciel afin de créer des schémas et des graphiques. Vous pouvez aussi l'associer à vos propres programmes en tant que périphérique de saisie.

Machine à additionner

Nous avons écrit sur le BBC Micro un programme simple pour utiliser Grafpad en saisie d'un programme d'addition. Une grille comportant toutes les clefs nombres figure sous la pellicule de Plexiglas. L'effleurement de la touche appropriée par le photostyle assure l'exécution du logiciel.

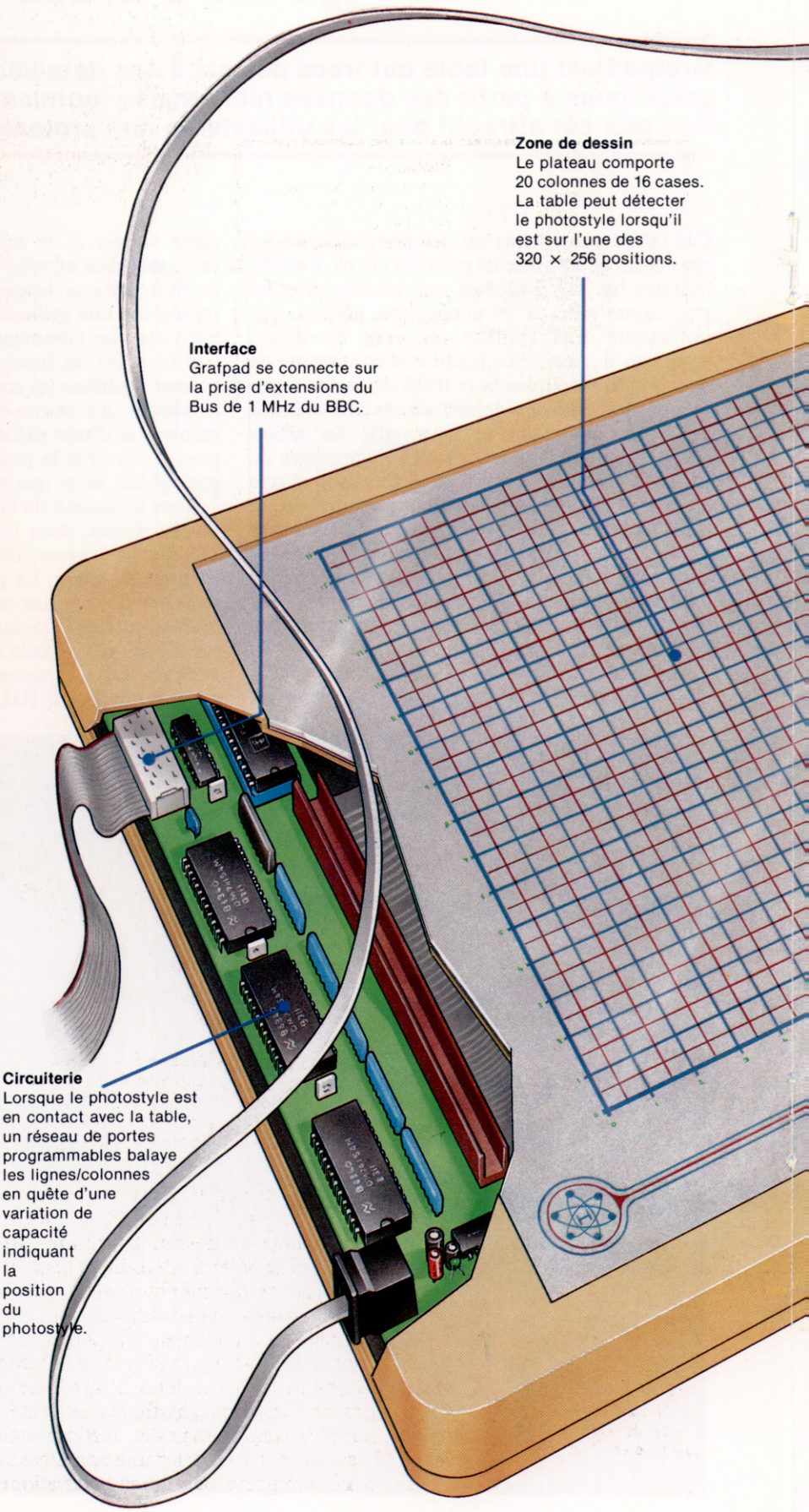
```
10  REM DEMO GRAFPAD
20  REM
30  REM MACHINE A ADDITIONNER UTILISANT GRAFPAD
40  REM
50  MODE 1
60  PRINT "MACHINE A ADDITIONNER":PRINT
70  HINEM = &29FF
80  XADJUST = 7
90  REM
100 REM ETABLIR LA TABLE DES COORDONNEES
110 REM
120 DIM B(15) :FOR I = 1 TO 15:B(I) = I*25:NEXT I
130 R1 = 0:R = 0
140 REM
150 REM CHARGEMENT DU PROGRAMME DE COMMANDE DE GRAFPAD
160 REM
170 *LOAD "LECTURE TABLE" 2A00
180 PEN% = &2A00
190 REM
200 REM DEBUT DU PROGRAMME PRINCIPAL
210 REM
220 REM ATTENTE DU CONTACT DU PHOTOSTYLE
230 CALL PEN%
240 !X% = !X% - XADJUST:IF !X% < 0 THEN !X% = 0
250 IF ?U% > 0 THEN 230
260 REM BIP AU CONTACT DU PHOTOSTYLE / REGISTRE
270 SOUND 1,-15,120,1
280 X = !X%
290 REM TRANSCRIT LA POSITION EN 0-12
300 I = 0
310 IF X > B(I) THEN I = I + 1:GOTO 310
320 REM INTERPRETER CODES 0-12
330 IF I < 10 THEN R = R * 10 + I
340 IF I = 10 THEN PRINT R1" +":R1 = R1 + R:R = 0
350 IF I = 11 THEN PRINT R1" =":PRINT "  ":R1
    = R1 + R:PRINT R1:R1 = 0:R = 0:PRINT:PRINT
360 IF I = 12 THEN PRINT:PRINT"CLEAR":PRINT:R = 0:R1 = 0
370 REM ATTENDRE FIN DU CONTACT PHOTOSTYLE
380 CALL PEN%
390 IF ?U% = 0 THEN 380
400 REM BOUCLER POUR PROCHAIN CONTACT
410 GOTO 230
```




```
MACHINE
A ADDITIONNER

  34 +          4 +
  96 =          8 =
 -----         ----
 130           12

   3 +       19876 +
   9 +        8876 +
   6 +        8864 +
  28 +        1243 =
 126 =       -------
 -----       38859
 172
```

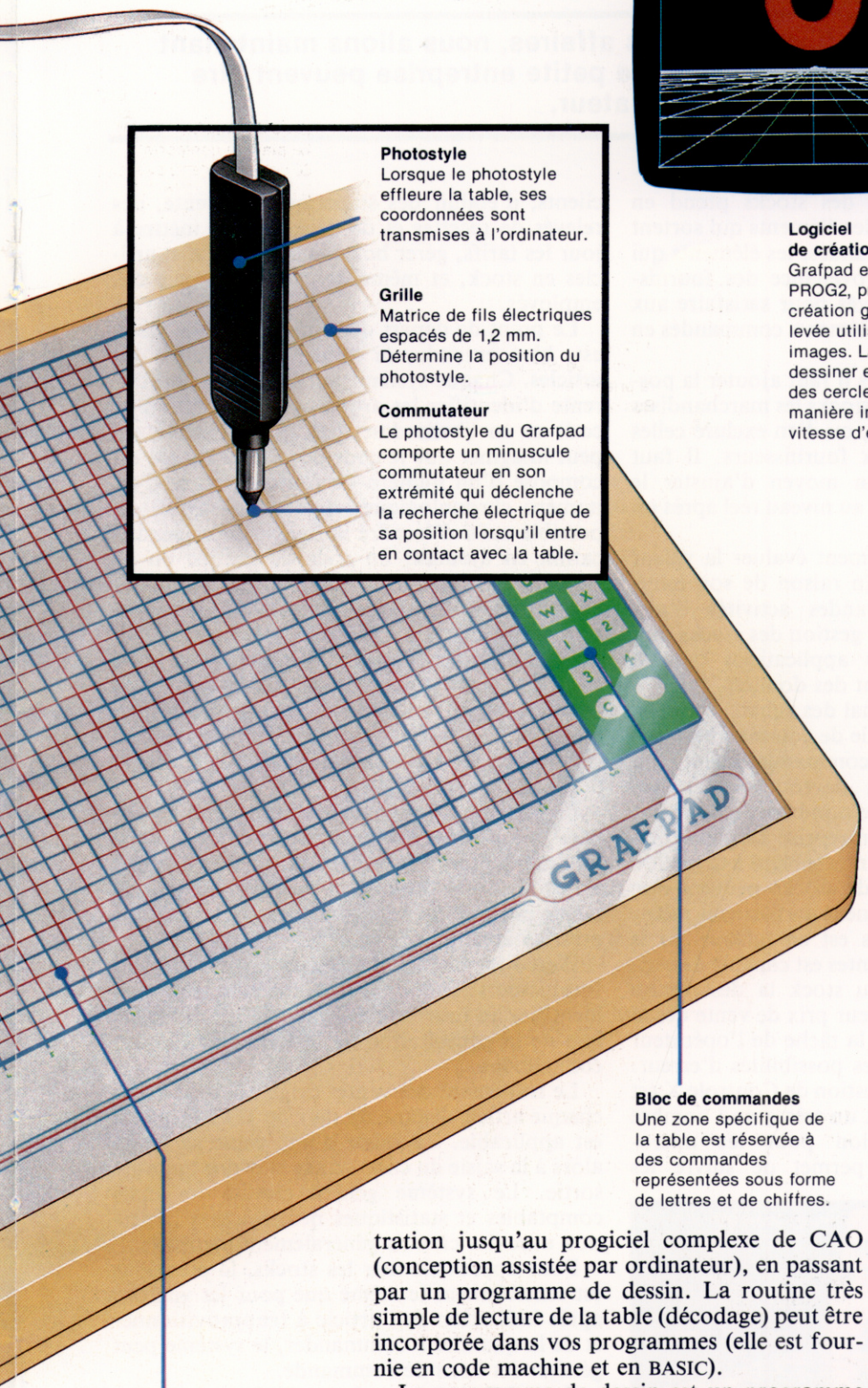


sions aux fils concernés et détecte la position du photostyle par ces variations de capacité. Ce balayage a lieu 2 000 fois par seconde, ce qui permet une localisation extrêmement rapide du photostyle. Le photostyle se tient par sa partie terminale, près de la pointe.

Lorsque le photostyle appuie sur la table, l'ordinateur reçoit le message « photostyle en position » et ses coordonnées. L'effet produit dépend du logiciel. Il se peut qu'un curseur sous forme d'une croix apparaisse à l'écran sur l'emplacement correspondant, ou bien qu'une commande déterminée soit exécutée.

Grafpad représente une économie de moyens considérable. En effet, le photostyle ne peut être détecté que sur une grille de 320 × 256 positions, ce qui rend le tracé de détails difficile. La table est également petite, au format A4 d'une feuille de papier : cette dimension réduite de la surface de travail de Grafpad est en fait très commode et correspond aux besoins courants.

Grafpad est accompagné de trois progiciels. Ils vont depuis une simple routine de démons-



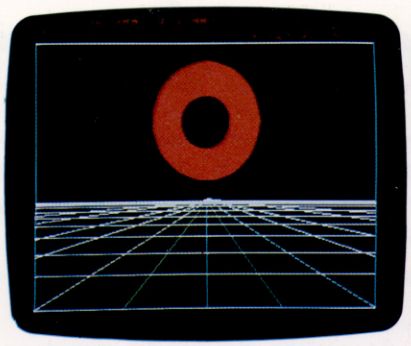
Photostyle
Lorsque le photostyle effleure la table, ses coordonnées sont transmises à l'ordinateur.

Grille
Matrice de fils électriques espacés de 1,2 mm. Détermine la position du photostyle.

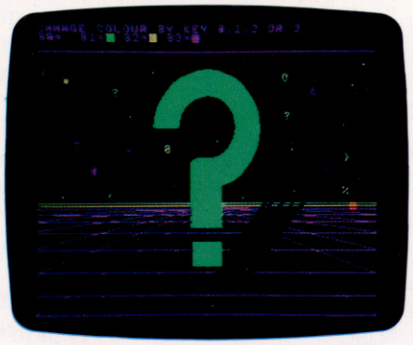
Commutateur
Le photostyle du Grafpad comporte un minuscule commutateur en son extrémité qui déclenche la recherche électrique de sa position lorsqu'il entre en contact avec la table.

Pellicule de Plexiglas
Une pellicule de Plexiglas protège la surface de la table. Des grilles de saisie peuvent venir s'y insérer.

Bloc de commandes
Une zone spécifique de la table est réservée à des commandes représentées sous forme de lettres et de chiffres.



Logiciel de création graphique
Grafpad est livré avec PROG2, progiciel de création graphique à main levée utilisé pour ces images. La capacité de dessiner et de colorier des cercles accroît de manière importante la vitesse d'exécution.



tration jusqu'au progiciel complexe de CAO (conception assistée par ordinateur), en passant par un programme de dessin. La routine très simple de lecture de la table (décodage) peut être incorporée dans vos programmes (elle est fournie en code machine et en BASIC).

Le programme de dessin est un programme électronique comparable à la plupart des progiciels de ce type avec ou sans table à numériser. Il présente toutes les caractéristiques de base : lignes, cases, cercles, triangles et dessin à main levée. Il peut en outre colorier une zone. Néanmoins, il ne comporte pas de caractéristiques très sophistiquées : ainsi il ne peut pas copier ni déplacer les zones d'un dessin. En outre, il n'a rien qu'un logiciel de dessin à partir du clavier ne puisse faire, bien qu'il permette de tracer des schémas. La version BBC affiche seulement quatre couleurs à la fois et a un temps de réponse relativement lent.

Le programme de CAO constitue une démonstration des caractéristiques du système. Vous créez d'abord un jeu de caractères à utiliser dans vos créations ultérieures. Dans le domaine de l'électronique, ces caractères graphiques ou formes peuvent être des composants du genre transistors, résistances, etc. Vous pouvez aussi créer des portes logiques, ou dans d'autres domaines, pourquoi pas des meubles, des tuiles ou des pots de fleurs! Une fois ces objets créés, vous pouvez passer à l'action sur la table elle-même, véritable tableau noir. Vous pouvez librement répéter ces formes et les agencer, les relier par des lignes.

Cela ressemble beaucoup aux vrais progiciels de CAO. Le Grafpad n'est pas pour autant destiné à un usage professionnel. Ainsi il vous manquera des caractéristiques telles que nom des diagrammes, rotation et échelle des dessins, zoom sur une zone de l'écran, positionnement très précis de détails, etc.

Grafpad par lui-même est un périphérique aux multiples utilisations et très économique. Il est bien sûr limité en surface, en résolution graphique et en fiabilité; mais c'est la contrepartie de son prix. Le logiciel accompagnant Grafpad est, lui, plutôt décevant. Aussi la table est-elle surtout intéressante pour ceux qui écrivent leurs propres programmes. Moyennant quoi, ce genre de table numérique peut permettre l'exploration de nouveaux domaines et inciter au développement de nouveaux micros plus performants sur le plan graphique.

Tenue de stock

Parmi les logiciels utilisés dans les affaires, nous allons maintenant examiner comment les stocks d'une petite entreprise peuvent être mieux contrôlés par un micro-ordinateur.

Un système de gestion des stocks prend en compte quatre aspects : les éléments qui sortent pour satisfaire les commandes, les éléments qui sont réceptionnés en provenance des fournisseurs, les éléments réservés pour satisfaire aux commandes à venir, les éléments commandés en réapprovisionnement.

A ces quatre catégories il faut ajouter la possibilité de réintégrer aux stocks les marchandises renvoyées par les clients, ou d'en exclure celles qui sont retournées aux fournisseurs. Il faut également disposer d'un moyen d'ajuster le niveau des stocks calculé au niveau réel après un inventaire physique.

Le système doit également évaluer la valeur monétaire des stocks. En raison de son interconnexion avec les grandes activités d'une entreprise, le système de gestion des stocks doit être intégré à d'autres applications qui lui apportent ou en extraient des données.

Par exemple, un journal des achats, un journal des ventes, un module de commandes fournisseurs, un module de commandes clients, un module de facturation devront être tenus. L'intégration présente de nombreux avantages. Si, par exemple, une entreprise dispose d'un module de gestion des ventes intégré à la gestion des stocks, les fichiers de stocks peuvent être mis à jour automatiquement au moment même où la commande clients est enregistrée. Si le module de gestion des ventes est capable de vérifier dans les fichiers du stock la description complète des articles et leur prix de vente dès la saisie d'un code article, la tâche de l'opérateur sera limitée, ainsi que les possibilités d'erreur.

Le logiciel CX Multigestion de Controle X est un programme intégrant un très grand nombre d'applications de gestion pour les micro-ordinateurs Apple. Il permet de suivre les clients, d'établir des statistiques de vente, des relevés de recettes et de dépenses, de mettre à jour les tarifs, gérer bons de commande et articles en stock, et même de calculer la paie des employés.

Avoir des rayons bien garnis
Les informations contenues dans les codes à barres sont lues automatiquement aux caisses et envoyées vers l'ordinateur central qui contrôle les stocks. Ce suivi instantané permet aux grandes surfaces de s'assurer que leurs rayons et leurs magasins contiennent toujours les bons produits en quantité suffisante. (Cl. Jean Riby/Éd. Atlas.)

Le point de départ de toute gestion de stock est, bien entendu, le fichier d'inventaire des articles. Chaque système aura sa manière différente d'identifier les articles par un numéro de code et un libellé descriptif. Par exemple, on peut imaginer que le code de chaque article est composé d'un numéro de fournisseur, et d'un numéro chronologique par fournisseur. Il est possible aussi d'étendre la finesse de la classification en ajoutant un code de groupe ou de famille de produits. L'ensemble de ces codes constitue ainsi un numéro unique d'identification qui peut être justifié sur neuf ou dix caractères, à droite ou à gauche. Un numéro de code justifié à gauche permet d'inscrire des codes de différentes longueurs pour plusieurs produits, et peut être utile pour identifier certains types d'articles tels que le modèle, la couleur, la taille, etc.

Le système de codage peut également distinguer des groupes à l'intérieur du stock et ainsi rendre possibles des traitements différents selon les groupes de produits. Parmi ceux-ci, il est même possible de faire une distinction entre les prestations et les produits.

Il est aussi intéressant de disposer d'un fichier fournisseurs. Cela permet de recenser et de bien identifier les fournisseurs, ainsi que de rappeler, lors de la consultation des articles, où ils sont commandés.

Le traitement des ventes exige également que chaque article dispose du taux de T.V.A. qui lui est applicable. La saisie d'une vente se limite alors à la saisie du code article et d'une quantité sortie. Le système génère ensuite les états comptables et statistiques nécessaires à la gestion avec des cumuls généralement mensuels.

Pour approvisionner les stocks, le processus est sensiblement le même que pour les ventes : après indication de l'article à réapprovisionner et de la quantité à commander, le système peut générer les bons de commande.

Quand les commandes sont livrées par les fournisseurs, il faut alors enregistrer les réceptions de façon à incrémenter les stocks. La saisie des réceptions est pratiquement identique à celle des commandes, et la boucle est bouclée.

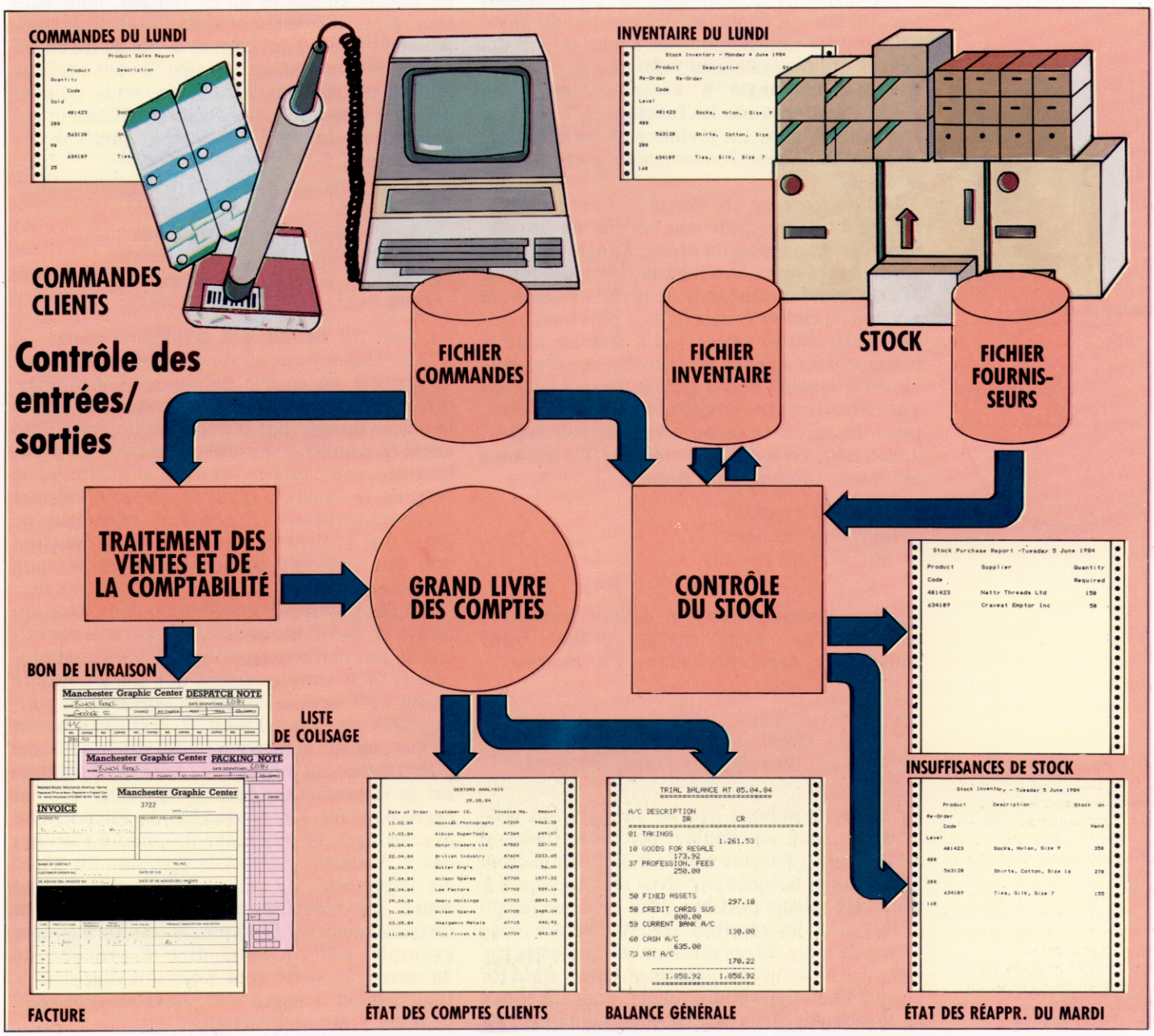
Il existe des logiciels de gestion de stock pour tous les types d'applications, depuis la petite

épicerie jusqu'à la grosse entreprise, et pour tous les systèmes. Citons le logiciel d'un magasin de détail, Gesma Gestion, conçu par APDI. Il s'applique au cas d'un magasin comprenant une vingtaine de vendeurs et jusqu'à 1 000 articles. Il effectue le suivi du stock, l'analyse des ventes, le suivi de la marge, la valorisation de l'inventaire et l'état des commandes en cours. Toutes ses fonctions sont protégées par des mots de passe.

Certains logiciels de gestion de stock s'appliquent à certaines catégories bien précises : il existe par exemple des programmes réalisés tout particulièrement pour la gestion de pharmacies, ou même pour une usine de micro-ordinateurs.

PGS est un programme de gestion de stock fonctionnant sur des micro-ordinateurs compatibles IBM PC. Ses principales fonctions recouvrent la saisie et la mise à jour des fiches articles et fournisseurs, la génération et l'édition des bons de commandes, la réception des produits avec mise à jour des entrées en stock, les ventes aux clients avec sortie des stocks, l'édition d'étiquettes d'articles et de statistiques. Bien sûr, ce logiciel est destiné à des micro-ordinateurs relativement importants (16 bits), mais il en existe aussi pour les ordinateurs familiaux, tel le programme de gestion de stock proposé par la société Loriciels pour l'Oric-1. Ce programme permet de gérer jusqu'à 410 fiches articles et 50 fiches fournisseurs. Étant donné le support utilisé — des cassettes, alors que sur les systèmes plus importants ce sont des disquettes, voire des disques durs — le programme ne peut avoir la rapidité des logiciels professionnels de gestion de stock, l'Oric n'étant pas au départ une machine franchement professionnelle. Mais pour moins de 200 F, ce programme sera bien utile pour gérer un gros congélateur ou une cave...

Contrôle des stocks
Les systèmes intégrés de contrôle des ventes et des stocks des entreprises à fort chiffre d'affaires ne peuvent que bénéficier d'un système automatisé au point de vente, pour tenir à jour l'inventaire et les fichiers de comptes. Les données de ventes peuvent être inscrites sous forme de codes à barres sur le produit. Ceux-ci sont lus par un lecteur optique, au niveau de la caisse enregistreuse, relié soit à un micro-ordinateur, soit à un terminal qui transmet les données à l'ordinateur central. (Cl. Tony Duncan/ Smith/Liz Dixon.)



Degrés de précision

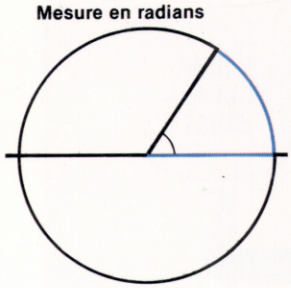
Dans ce deuxième article sur les mathématiques dans la programmation basic, nous étudions l'emploi que l'on peut faire des fonctions sinus et cosinus.

On pourrait penser que calculer la position d'un point sur une ligne après lui avoir fait subir une rotation d'un certain nombre de degrés est une chose facile puisque le BASIC comporte les fonctions COS et SIN. Le COSinus d'un angle θ donne la position selon l'axe des *x* (l'abscisse). Le SINus de l'angle donne la position selon l'axe des *y* (ordonnée).

La difficulté d'utilisation de ces fonctions tient au fait que la plupart des versions du BASIC utilisent des radians pour les angles et non des degrés. Il faut également faire attention au fait que lorsque l'angle θ avoisine 0 ou 1, les valeurs produites par les fonctions COS et SIN ne sont plus fiables. La première chose à voir est donc la distinction fondamentale entre radians et degrés.

Si l'on trace un arc de cercle (une partie du cercle) d'une longueur telle qu'elle soit exactement égale au rayon du cercle, l'angle au centre est dit égal à un radian (voir la figure ci-contre). Si l'on prend comme unité le rayon du cercle, la portion de circonférence définie plus haut prendra également la valeur 1. La formule mathématique pour trouver la circonférence d'un cercle est $2\pi r$, aussi doit-il y avoir 2π radians pour une circonférence complète. Une rotation complète du rayon trace un cercle de 360 degrés. Aussi, 360° est-il égal à 2π radians. Ce qui nous indique le rapport entre degrés :

Mesure en radians



$360° = 2\pi$ radians
$180° = \pi$ radians
$90° = \pi/2$ radians
$1° = \pi/180 = 0{,}0174$ radian

Un programme BASIC qui devrait trouver le cosinus d'un angle devrait préalablement convertir les degrés en radians. Par exemple :

```
10 INPUT «DONNEZ L'ANGLE EN DEGRÉS »;A
20 LET B# = A * 0.0174
30 LET C# = COS(B#)
40 PRINT « LE COSINUS DE »;A;« DEGRÉS EST »; C#
50 END
```

Le signe «#» indique que les variables du programme sont en double précision (nous y reviendrons). Par une simple modification de ce programme, la fonction SINus sera appliquée à toutes les valeurs possibles pour un angle (de 0° à 360°), et les résultats seront disposés sous forme de table. Si ces résultats sont reportés sur l'axe des ordonnées (l'axe des abscisses étant les valeurs en radians pour l'angle), nous obtenons la courbe des sinus familière aux amateurs de hi-fi et aux ingénieurs en électronique. Il s'agit de la projection des positions représentant l'intersection de l'hypoténuse avec le cercle unité sur l'axe des *y* et pour tous les angles de rotation. En d'autres termes, il s'agit d'une manière différente pour aboutir à décrire mathématiquement un cercle.

Certaines versions BASIC autorisent aussi bien les calculs en degrés qu'en radians, mais elles sont rares. Elles utilisent alors une sorte de commutateur logiciel qui permet de passer du calcul en degrés au calcul en radians. Si vous préférez utiliser uniquement les degrés, vous pourrez définir à votre usage une fonction utilisateur qui assurera les conversions automatiquement. En voici une :

```
10 REM FONCTION UTILISATEUR POUR TRAVAILLER
   EN DEGRÉS
20 DEF FN SID (D#) = SIN(D#*0.0174532931)
30 INPUT «ENTREZ L'ANGLE EN DEGRÉS »;D#
40 PRINT «LE SINUS DE »;D#,« DEGRÉS EST »; FN SID(D#)
50 END
```

La ligne 20 définit une fonction appelée SID (SInus/Degrés) qui utilise pour seul paramètre la variable en double précision D#. La partie droite de la définition indique comment obtenir la valeur donnée par la fonction (le sinus d'un angle en degrés). Pour utiliser une fonction utilisateur, vous prenez, comme d'habitude, le nom de la fonction (avec la valeur à calculer entre parenthèses). Il convient cependant de noter que la ligne de définition de la fonction doit être exécutée avant tout appel de cette dernière.

Un des problèmes d'utilisation de la fonction SINus en BASIC est que tous les BASIC ne la calculent pas correctement lorsque l'angle approche de 0. Il semble évident que SIN θ doit être voisin de 0 lorsque l'angle est voisin de 0, puisque SIN θ est égal à 0 pour $\theta = 0$. En d'autres termes, au fur et à mesure que l'angle se ferme pour finalement atteindre 0, l'arc qu'il délimite sur la circonférence décroît vers 0, et le point correspondant à l'intersection de l'hypoténuse avec le cercle se rapproche du point 0 sur l'axe des *y*. Malheureusement la précision du BASIC est imparfaite. Ce qui signifie qu'il ne peut traiter les valeurs grandes et petites que dans certaines limites. Lorsque θ est infime (par exemple 10^{-36}, c'est-à-dire 1 précédé de 36 zéros), le BASIC sera incapable d'assurer le traitement et donnera zéro. Aussi est-il prudent de tester votre BASIC avant.

Sous-estimation du calcul du sinus ou erreurs d'approximation

```
1 REM TEST DE SOUS-ESTIMATION DU SINUS OU ERREURS D'APPROXIMATION
10 LET X = 10/6
20 PRINT «ITÉRATION», «VAL DE X», «VAL DE SIN (X)»
30 FOR I = 1 TO 40
40 LET X = X/10
50 PRINT I,X,SIN(X)
60 NEXT I
70 END
```

ITERATION	VAL DE X	VAL DE SIN(X)
1	.166667	.165896
2	.0166667	.0166659
3	1.66667E-03	1.66667E-03
4	1.66667E-04	1.66667E-04
5	1.66667E-05	1.66667E-05
6	1.66667E-06	1.66667E-06
7	1.66667E-07	1.66667E-07
8	1.66667E-08	1.66667E-08
9	1.66667E-09	1.66667E-09
10	1.66667E-10	1.66667E-10
11	1.66667E-11	1.66667E-11
12	1.66667E-12	1.66667E-12
13	1.66667E-13	1.66667E-13
14	1.66667E-14	1.66667E-14
15	1.66667E-15	1.66667E-15
16	1.66667E-16	1.66667E-16
17	1.66667E-17	1.66667E-17
18	1.66667E-18	1.66667E-18
19	1.66667E-19	1.66667E-19
20	1.66667E-20	1.66667E-20
21	1.66667E-21	1.66667E-21
22	1.66667E-22	1.66667E-22
23	1.66667E-23	1.66667E-23
24	1.66667E-24	1.66667E-24
25	1.66667E-25	1.66667E-25
26	1.66667E-26	1.66667E-26
27	1.66667E-27	1.66667E-27
28	1.66667E-28	1.66667E-28
29	1.66667E-29	1.66667E-29
30	1.66667E-30	1.66667E-30
31	1.66667E-31	1.66667E-31
32	1.66667E-32	1.66667E-32
33	1.66667E-33	1.66667E-33
34	1.66667E-34	1.66667E-34
35	1.66667E-35	1.66667E-35
36	1.66667E-36	1.66667E-36
37	1.66667E-37	1.66667E-37
38	1.66667E-38	1.66667E-38
39	0	0
40	0	0

Une exploitation de ce programme appliqué au BASIC Microsoft est donnée ici. Ce BASIC traite convenablement des petites valeurs d'angle jusqu'à 10^{-38}.

La bonne exécution de ce programme dépend de l'amplitude dynamique du BASIC dans le traitement des opérations arithmétiques en virgule flottante. Il convient de répéter à cette occasion que, préalablement à toutes opérations mathématiques en BASIC, vous devez chercher à connaître les limites de l'ensemble des nombres traités avec succès.

Souvenez-vous aussi qu'un nom de variable seul, tel que X ou TREND, sera nécessairement en précision simple (c'est-à-dire ne pouvant stocker que 7 chiffres). Il est également possible d'indiquer cette précision simple pour une variable en la faisant suivre d'un point d'exclamation, comme dans X! ou TREND!. Les variables en double précision (elles peuvent stocker jusqu'à 17 chiffres) sont indiquées en les faisant suivre du signe «#», comme dans X# ou TREND#. Les variables entières (qui ne peuvent contenir que des nombres entiers) sont spécifiées dans de nombreuses versions du BASIC en les faisant suivre du signe «%», comme dans X% ou TREND%.

Nous terminerons cet article par un petit programme destiné à vous permettre de déterminer combien de chiffres vous pouvez mettre dans vos variables par rapport à votre version du BASIC. Vous trouverez également un listage du programme exécuté sous le BASIC Microsoft. Nous donnons en fait deux programmes, l'un pour les petits nombres, et l'autre pour les grands. Le listage pour les petits nombres montre que lorsqu'ils deviennent très faibles (inférieurs à $3{,}3 \times 10^{-38}$), le BASIC les arrondit à 0.

Les petits nombres en BASIC

```
1 REM TESTS SUR LE TRAITEMENT DES PETITS NOMBRES EN BASIC
10 LET X# = 0.000033333333#
20 PRINT «ITÉRATION», «DBL PREC», «», «SPL PREC»
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# / 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I      90 END
```

ITERATION	DBL PREC	SPL PREC
1	.000003333333333	3.33333E-06
2	.0000033333333333	3.33333E-07
3	3.333333333D-08	3.33333E-08
4	3.333333333D-09	3.33333E-09
5	3.333333333D-10	3.33333E-10
6	3.333333333D-11	3.33333E-11
7	3.333333333D-12	3.33333E-12
8	3.333333333D-13	3.33333E-13
9	3.333333333D-14	3.33333E-14
10	3.333333333D-15	3.33333E-15
11	3.333333333D-16	3.33333E-16
12	3.333333333D-17	3.33333E-17
13	3.333333333D-18	3.33333E-18
14	3.333333333D-19	3.33333E-19
15	3.333333333D-20	3.33333E-20
16	3.333333333D-21	3.33333E-21
17	3.333333333D-22	3.33333E-22
18	3.333333333D-23	3.33333E-23
19	3.333333333D-24	3.33333E-24
20	3.333333333D-25	3.33333E-25
21	3.333333333D-26	3.33333E-26
22	3.333333333D-27	3.33333E-27
23	3.333333333D-28	3.33333E-28
24	3.333333333D-29	3.33333E-29
25	3.333333333D-30	3.33333E-30
26	3.333333333D-31	3.33333E-31
27	3.333333333D-32	3.33333E-32
28	3.333333333D-33	3.33333E-33
29	3.333333333D-34	3.33333E-34
30	3.333333333D-35	3.33333E-35
31	3.333333333D-36	3.33333E-36
32	3.333333333D-37	3.33333E-37
33	3.333333333D-38	3.33333E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

Les grands nombres en BASIC

```
1 REM TESTS SUR LE TRAITEMENT DES GRANDS NOMBRES EN BASIC
10 LET X# = 3.3333333333334#
20 PRINT «ITÉRATION», «DBL PREC», «», «SPL PREC»
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# * 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I      90 END
```

ITERATION	DBL PREC	SPL PREC
1	33.33333333333334	33.3333
2	333.3333333333334	333.333
3	3333.333333333334	3333.33
4	33333.33333333334	33333.3
5	333333.3333333334	333333
6	3333333.333333334	3.33333E+06
7	33333333.33333334	3.33333E+07
8	333333333.3333334	3.33333E+08
9	3333333333.333334	3.33333E+09
10	33333333333.33334	3.33333E+10
11	333333333333.3334	3.33333E+11
12	3333333333333.334	3.33333E+12
13	33333333333333.34	3.33333E+13
14	333333333333333.4	3.33333E+14
15	3333333333333334	3.33333E+15
16	3.3333333333333334D+16	3.33333E+16
17	3.3333333333333334D+17	3.33333E+17
18	3.3333333333333334D+18	3.33333E+18
19	3.3333333333333334D+19	3.33333E+19
20	3.3333333333333334D+20	3.33333E+20
21	3.3333333333333334D+21	3.33333E+21
22	3.3333333333333334D+22	3.33333E+22
23	3.3333333333333334D+23	3.33333E+23
24	3.3333333333333334D+24	3.33333E+24
25	3.3333333333333334D+25	3.33333E+25
26	3.3333333333333334D+26	3.33333E+26
27	3.3333333333333334D+27	3.33333E+27
28	3.3333333333333334D+28	3.33333E+28
29	3.3333333333333334D+29	3.33333E+29
30	3.3333333333333334D+30	3.33333E+30
31	3.3333333333333334D+31	3.33333E+31
32	3.3333333333333334D+32	3.33333E+32
33	3.3333333333333334D+33	3.33333E+33
34	3.3333333333333334D+34	3.33333E+34
35	3.3333333333333334D+35	3.33333E+35
36	3.3333333333333334D+36	3.33333E+36
37	3.3333333333333334D+37	3.33333E+37
	OVERFLOW ERROR	IN 50

Mettre un drapeau

Après nous être servis de l'instruction d'addition, nous nous intéressons au registre d'état du processeur et à son rôle dans l'addition — en particulier celui du drapeau de retenue.

L'instruction d'addition dans les langages d'assemblage de Z80 et 6502 est ADC (« additionner avec retenue »), mnémonique d'une grande importance pour la programmation en langage d'assemblage. Le concept de bit de « retenue » est particulièrement significatif. Considérons l'addition de deux nombres hex dans l'accumulateur :

```
   A7      =    10100111
+  3E      =  + 00111110
-----------------------
   E5      =    11100101
```

Puisque l'accumulateur est un registre à huit bits, le nombre à additionner et la somme elle-même doivent être compris entre $00 et $FF (comme c'est le cas ici), sinon ils n'entrent pas dans l'accumulateur. Devons-nous en déduire que nous sommes limités à des additions dont la somme est inférieure à $100? Considérons une autre addition dans l'accumulateur, où cette restriction est violée :

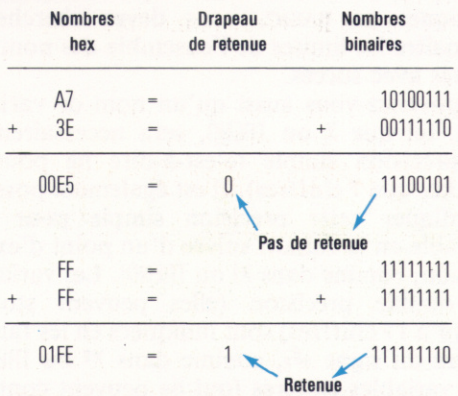
```
   FF      =    11111111
+  FF      =  + 11111111
-----------------------
   1FE     =    111111110
```

C'est l'addition des deux nombres le plus grand possible à un octet, et elle paraît illégale. Elle requiert un accumulateur à neuf bits. La solution de ce dilemme est suggérée dans l'énoncé du problème — nous avons seulement besoin d'un bit supplémentaire dans l'accumulateur pour contenir le plus grand nombre pouvant être généré par l'addition de deux nombres à un octet. Ce bit supplémentaire n'est nécessaire que dans la somme, et non dans les opérandes de l'addition, et uniquement s'il y a une retenue provenant du bit le plus significatif de l'accumulateur.

Le bit supplémentaire est donc appelé « bit de retenue » et il est localisé dans le registre à huit bits associé à l'accumulateur et connu sous le nom de « registre d'état du processeur » ou PSR *(Processor Status Register)*. Cet important registre est connecté à l'accumulateur et à l'ALU de telle sorte que chaque bit du PSR est mis à 1 ou 0 après toute opération de l'accumulateur, selon les résultats de cette opération. Le contenu du PSR peut être considéré comme un simple nombre, mais il est généralement plus intéressant de le traiter comme une série de huit drapeaux binaires, dont les états individuels montrent les effets particuliers de la dernière opération (un drapeau est une variable dont la valeur indique l'état ou la véracité d'une condition, plutôt qu'une valeur absolue. Une variable drapeau n'a généralement que deux états : haut ou bas, ouvert ou fermé, 0 ou 1).

Si l'on effectue sur l'accumulateur une opération qui implique une retenue dépassant le huitième bit, alors le drapeau de retenue du PSR sera automatiquement mis à 1; une opération n'entraînant pas de retenue remettra ce drapeau à 0. Cela ne s'applique qu'aux opérations pouvant légitimement impliquer une retenue. Certaines opérations telles que le chargement ou le stockage dans l'accumulateur n'affectent pas le drapeau de retenue. Chaque fois que nous étudierons une nouvelle instruction de langage d'assemblage, nous nous demanderons quels bits du PSR (ou registre de drapeau) elle affecte. Naturellement, il nous faut en savoir plus sur les autres drapeaux du PSR, mais terminons d'abord sur le drapeau de retenue.

En général, quand on additionne deux nombres à un octet, on ne sait pas d'avance ce qu'ils donneront, c'est pourquoi il faut se préparer à obtenir des sommes dépassant $FF; la plupart du temps, il s'agira de réserver deux octets de RAM pour contenir le résultat d'une addition. Considérons à nouveau les exemples d'addition précédents :



Le résultat de l'addition est représenté dans les deux exemples par un nombre à deux octets. Dans le premier cas, le drapeau de retenue est remis à 0 parce qu'il n'y a pas de retenue provenant du huitième bit de la somme (le résultat à deux octets est $00E5, dont l'octet hi est $00). Dans le second cas, cependant, il y a une retenue du huitième bit, et donc le drapeau de retenue est mis, et l'octet hi du résultat est $01.

Pour être sûr d'obtenir le résultat exact d'une addition, nous devons ainsi stocker le contenu

de l'accumulateur dans l'octet lo de l'emplacement à deux octets, puis stocker le drapeau de retenue dans l'octet hi de cet emplacement. Il n'existe pas d'instruction pour stocker uniquement le drapeau de retenue, mais l'opc (code opération) ADC implique précisément cette opération : en fait, ADC signifie « additionner l'opérande de l'instruction au contenu actuel du drapeau de retenue, puis additionner ce résultat au contenu de l'accumulateur ». L'addition est ainsi un processus en deux étapes : dans la première on utilise l'état actuel du drapeau de retenue et dans la seconde, cet état est mis à jour.

Cela implique qu'avant de commencer une addition, nous devons considérer l'état du drapeau de retenue, puisqu'il devra intervenir dans l'addition; d'où les deux instructions inexpliquées auparavant, CLC et AND A. La première, une instruction 6502, signifie « mettre à zéro le drapeau de retenue » et fait précisément cela. La version Z80, AND A, signifie « effectuer l'opération ET logique de l'accumulateur avec lui-même ». Bien qu'elle ne soit pas destinée uniquement à remettre à zéro le drapeau de retenue, elle a bien cet effet et n'affecte rien d'autre, de sorte qu'on l'utilise comme l'équivalent Z80 de l'instruction 6502 CLC.

Ayant mis le drapeau de retenue à zéro avant de commencer une addition, nous devons donc stocker ensuite son contenu. On fait cela en additionnant la valeur immédiate $00 à l'octet hi du résultat.

Tout ce que nous avons dit conduit à une première méthode pour l'arithmétique à un octet :
1) mettre à zéro le drapeau de retenue;
2) charger l'accumulateur avec un nombre;
3) additionner le second nombre;
4) stocker le contenu de l'accumulateur dans l'octet lo d'un emplacement à deux octets;
5) charger l'accumulateur avec le contenu de l'octet hi;
6) additionner la valeur immédiate $00;
7) stocker le contenu de l'accumulateur dans l'octet hi.

Lorsque cette procédure est traduite en langage d'assemblage, nous obtenons :

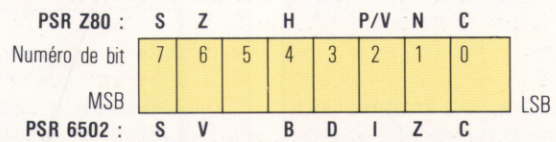
COMMUN AUX DEUX PROCESSEURS		
Label	**Directive**	**Opérande**
BYTE1	EQU	$FF
BYTE2	EQU	$FF
LOBYTE	EQU	$A000
HIBYTE	EQU	$A001
	ORG	$A020

Z80		6502	
Opc	**Opérande**	**Opc**	**Opérande**
LD	A,$00	LDA	#$00
LD	(HIBYTE),A	STA	HIBYTE
AND	A	CLC	
LD	A,BYTE1	LDA	#BYTE1
ADC	A,BYTE2	ADC	#BYTE2
LD	(LOBYTE),A	STA	LOBYTE
LD	A,(HIBYTE)	LDA	HIBYTE
ADC	A,$00	ADC	#$00
LD	(HIBYTE),A	STA	HIBYTE
RET		RTS	

Rappelons que ces valeurs de LOBYTE (octet lo), HIBYTE (octet hi) et ORG sont seulement données à titre d'exemple — à vous de choisir les valeurs appropriées à votre machine. Remarquez que les deux premières instructions du programme chargent $00 dans HIBYTE, pour éviter l'influence de données aléatoires.

Soulignons encore les différences d'approche entre les langages d'assemblage Z80 et 6502, d'après notre exemple. Le langage 6502 se lit tout simplement une fois que vous en avez l'habitude — les mnémoniques eux-mêmes et l'usage de «#» pour indiquer une donnée immédiate rendent clair le sens de chaque instruction. La version Z80 est moins évidente parce que le mnémonique LD est utilisé pour tous les transferts de données, que ce soit en entrée ou en sortie de l'accumulateur. De plus, il n'y a pas le symbole «#» pour signaler une donnée immédiate, c'est seulement l'absence de parenthèses autour de l'opérande qui l'indique. Ainsi LD A, BYTE1 signifie « charger l'accumulateur avec la donnée immédiate BYTE1 », tandis que LD A, HIBYTE, signifie « charger l'accumulateur à partir de l'adresse HIBYTE ». Dans tout le listage en langage d'assemblage, il n'y a pas d'ambiguïté quant à la signification de telles instructions, puisque la valeur hex de l'opc identifie univoquement l'instruction. Une question peut toutefois se poser : l'opc peut être unique, mais s'il y a un choix d'opc uniques, comment l'assembleur (ou la personne qui fait l'assemblage) choisit-il entre eux? La réponse est fournie par le mode d'adressage, dont nous parlerons ultérieurement.

Enfin, il faut noter que le PSR contient d'autres drapeaux que celui de retenue, que nous allons brièvement examiner maintenant et sur lesquels nous reviendrons dans la suite de ce cours :



BIT PSR	Z80	8502	BIT PSR
7	(S) = SIGNE	(S) = SIGNE	7
6	(Z) = ZÉRO	(V) = DÉPASSEMENT	6
5	inusité	inusité	5
4	(H) = DEMI-RETENUE	(B) = RUPTURE	4
3	inusité	(D) = MODE DCB	3
2	(P/V) = PARITÉ/DÉPASSEMENT	(I) = INTERRUPTION	2
1	(N) = SOUSTRACTION	(Z) = ZÉRO	1
0	(C) = RETENUE	(C) = RETENUE	0

Pour le moment, nous nous intéressons aux drapeaux de retenue, de signe et de zéro. Nous avons vu qu'après une addition le premier porte la valeur de la retenue provenant du huitième bit de l'accumulateur. Le deuxième est toujours une copie du huitième bit (bit 7) de l'accumulateur, et le troisième est mis à 1 si le contenu de l'accumulateur est nul, et à 0 sinon.

Réponses aux exercices d'assemblage

1) Les programmes assemblés sont donnés dans le tableau ci-contre, à droite.
Notez que les symboles BYTE1 et BYTE2 sont utilisés à la fois comme données symboliques immédiates et comme adresses symboliques. Dans le second cas, cependant, ils doivent être assemblés sous la forme de deux octets.
2) L'instruction « retour de sous-programme » manque à la fin des deux programmes. En version 6502, le code complet nécessiterait la ligne suivante :

```
A00D            60              RTS
```

et la version Z80 :

```
A00D            C9              RET
```

3) La valeur $45 est d'abord chargée dans le registre accumulateur comme une donnée immédiate, puis $45 est ajoutée à son sommet, de sorte que l'accumulateur contient la valeur $8A. Ce total accumulé est ensuite stocké en RAM à l'adresse $0045. La valeur $38 est additionnée dans l'accumulateur comme une donnée immédiate, et celui-ci contient la valeur $C2($45+$45+$38). Enfin, ce total est stocké en RAM à l'emplacement $0038.
4) Une « donnée immédiate » est une donnée qui est stockée dans l'instruction. Dans les instructions que nous avons données dans les programmes d'exercices (telles que LDA #$9C et LD A,$E4) les valeurs $9C et $E4 sont des données à charger dans l'accumulateur. Elles sont stockées dans les instructions dont elles sont les opérandes, et comprennent le contenu de l'octet suivant immédiatement l'opc. Si la donnée n'est pas valable, elle doit être stockée ailleurs dans la mémoire, et on doit s'y référer par son adresse plutôt que par sa valeur.
5) La valeur de BYTE1 est donnée égale à $45, ce qui correspond à l'emplacement mémoire $0045. En clair, cette adresse est à la page zéro de la mémoire.

Exercice

Si nous désirons examiner le contenu du registre d'état du processeur (PSR), il est commode d'écrire ce nombre sous forme binaire plutôt que hex. Voici la version Spectrum d'un « sous-programme de conversion décimal-binaire ». Cet exercice vous demande de raccorder ceci au programme moniteur vu précédemment.

```
7000 REM****** S-P OCTET BINAIRE ******
7001 REM* CONVERTIT UN NOMBRE N (< 256) *
7002 REM* EN REPRESENTATION BINAIRE
7003 REM* A 8 CARACTERES DANS B$
7010 B$ = «»
7020 FOR D = 8 TO 1 STEP-1
7030 LET N1 = INT(N/2)
7040 LET R = N - 2 * N1
7050 LET B$ = STR$(R) + B$
7060 LET N = N1
7070 NEXT D
7080 RETURN
```

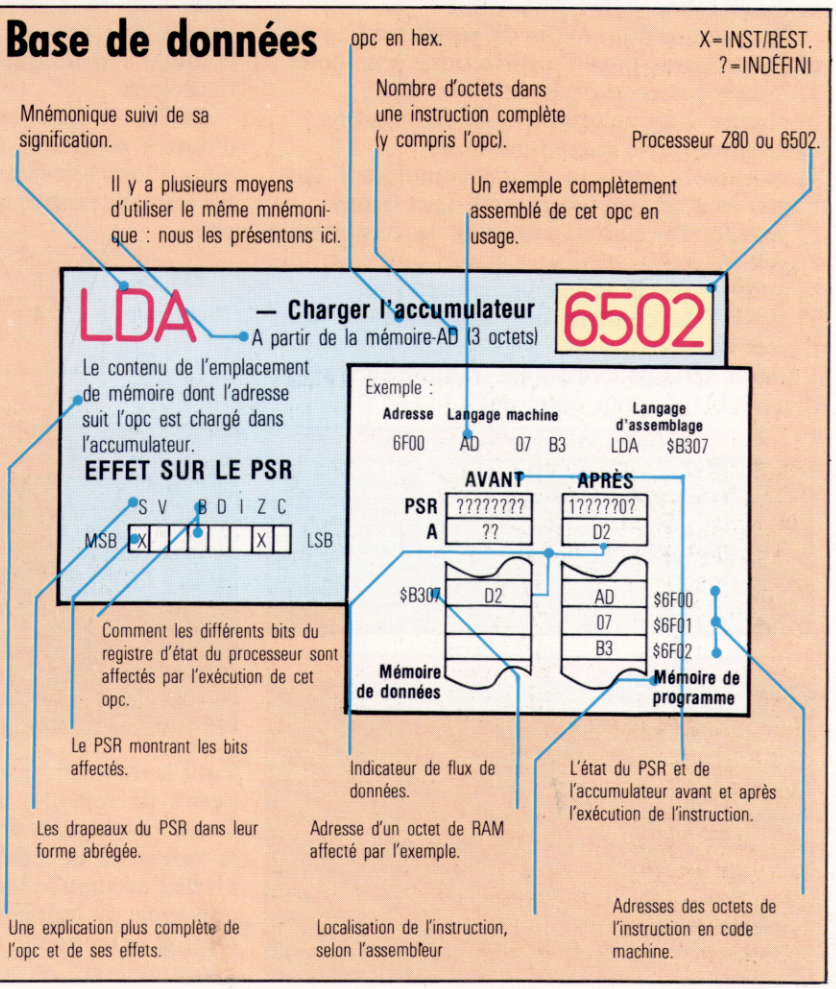
Variantes de basic

Sur le Commodore 64, remplacez la ligne 7050 dans le sous-programme par :

```
7050 B$ = MID$(STR$(R),2) + B$
```

Adresse	Langage machine	Langage d'assemblage		
6502				
0000		START	EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	START
A000	A9 45		LDA	#BYTE1
A002	18		CLC	
A003	69 45		ADC	#BYTE1
A005	8D 45 00		STA	BYTE1
A008	69 38		ADC	#BYTE2
A00A	8D 38 00		STA	BYTE2
Z80				
0000		START	EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	START
A000	3E 45		LD	A,BYTE1
A002	A7		AND	A
A003	CE 45		ADC	A,BYTE1
A005	32 45 00		LD	(BYTE1),A
A008	CE 38		ADC	A,BYTE2
A00A	32 38 00		LD	(BYTE2),A

Base de données



LDA - CHARGER L'ACCUMULATEUR — 6502

Immédiat. -A9 (2 octets)

Le contenu de l'octet suivant l'opc est chargé dans l'accumulateur.

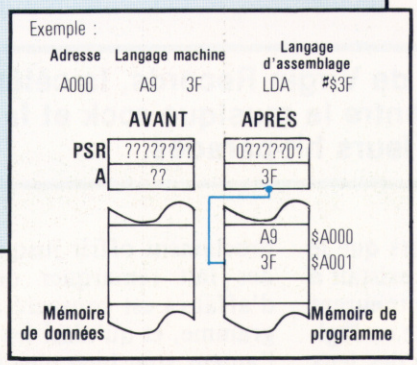
EFFET SUR LE PSR

S V B D I Z C

MSB [X][ ][ ][ ][ ][X][ ] LSB

Exemple :

Adresse	Langage machine		Langage d'assemblage	
A000	A9	3F	LDA	#$3F



LD A, - CHARGER L'ACCUMULATEUR — Z80

Immédiat. -3E (2 octets)

Le contenu de l'octet suivant l'opc est chargé dans l'accumulateur.

EFFET SUR LE PSR

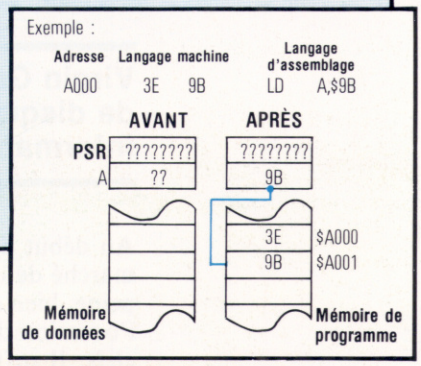
S Z H B N C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

PAS D'EFFET

Exemple :

Adresse	Langage machine		Langage d'assemblage	
A000	3E	9B	LD	A,$9B



LDA - CHARGER L'ACCUMULATEUR — 6502

A partir de la mémoire -AD (3 octets)

Le contenu de l'emplacement mémoire dont l'adresse suit l'opc est chargé dans l'accumulateur.

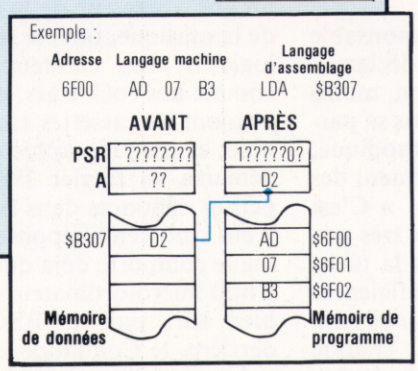
EFFET SUR LE PSR

S V B D I Z C

MSB [X][ ][ ][ ][ ][X][ ] LSB

Exemple :

Adresse	Langage machine			Langage d'assemblage	
6F00	AD	07	B3	LDA	$B307



LD A, - CHARGER L'ACCUMULATEUR — Z80

A partir de la mémoire -3A (3 octets)

Le contenu de l'emplacement mémoire dont l'adresse suit l'opc est chargé dans l'accumulateur.

EFFET SUR LE PSR

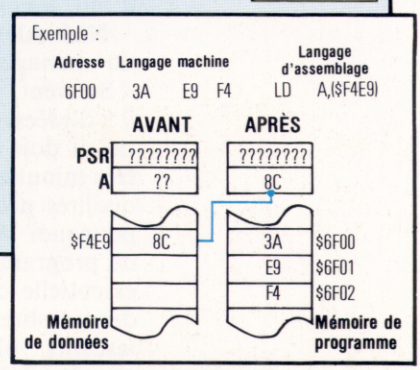
S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

PAS D'EFFET

Exemple :

Adresse	Langage machine			Langage d'assemblage	
6F00	3A	E9	F4	LD	A,($F4E9)



STA - STOCKER L'ACCUMULATEUR — 6502

Vers la mémoire -8D (3 octets)

Le contenu de l'accumulateur est chargé dans l'emplacement mémoire dont l'adresse suit l'opc.

EFFET SUR LE PSR

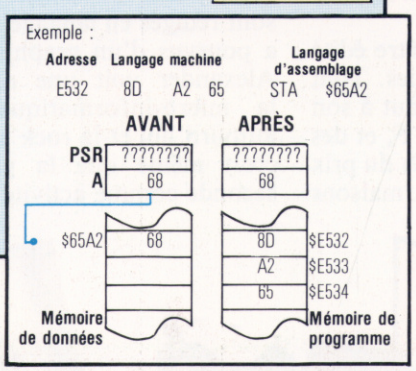
S V B D I Z C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

PAS D'EFFET

Exemple :

Adresse	Langage machine			Langage d'assemblage	
E532	8D	A2	65	STA	$65A2



LD( ),A - CHARGER L'ACCUMULATEUR — Z80

Vers la mémoire -32 (3 octets)

Le contenu de l'accumulateur est chargé dans l'emplacement mémoire dont l'adresse suit l'opc.

EFFET SUR LE PSR

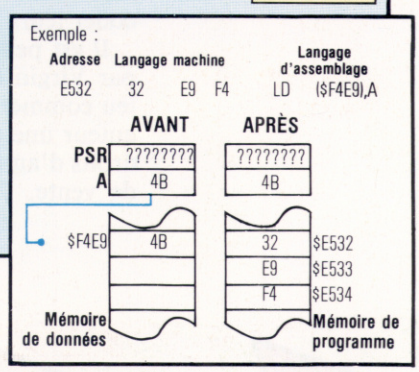
S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ] LSB

PAS D'EFFET

Exemple :

Adresse	Langage machine			Langage d'assemblage	
E532	32	E9	F4	LD	($F4E9),A



ADC - ADDITIONNER AVEC RETENUE — 6502

Immédiat. -69 (2 octets)

Le drapeau de retenue plus le contenu de l'octet suivant l'opc sont additionnés au contenu de l'accumulateur.

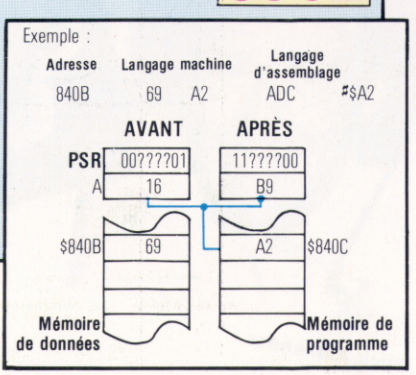
EFFET SUR LE PSR

S V B D I Z C

MSB [X][X][ ][ ][ ][X][X] LSB

Exemple :

Adresse	Langage machine		Langage d'assemblage	
840B	69	A2	ADC	#$A2



ADC A, - ADDITIONNER AVEC RETENUE — Z80

Immédiat. -69 (2 octets)

Au contenu de l'accumulateur sont additionnés le drapeau de retenue plus le contenu de l'octet suivant l'opc.

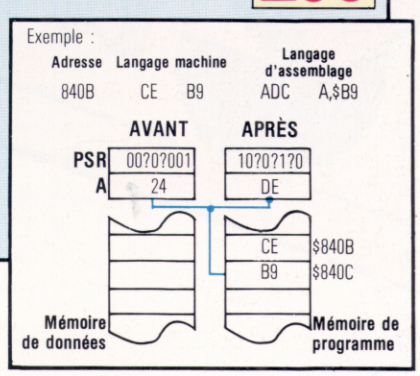
EFFET SUR LE PSR

S Z H V N C

MSB [X][X][X][ ][X][ ][X] LSB

Exemple :

Adresse	Langage machine		Langage d'assemblage	
840B	CE	B9	ADC	A,$B9



Des disques aux jeux

Virgin Games est une filiale de Virgin Records, la célèbre maison de disques. Quels rapports èntre la musique rock et la micro-informatique? Les deux ont leurs hit-parades.

Au début des années soixante-dix, alors que le marché de l'informatique individuelle existait à peine, bien des jeunes gens entreprenants eurent l'occasion de tirer profit de la demande en logiciels. Il suffisait alors de savoir écrire des jeux amusants en BASIC, de les reproduire rapidement sur cassettes et de vendre le tout par correspondance grâce aux petites annonces. Aujourd'hui, les choses sont moins simples.

Nick Alexander (vingt-huit ans), responsable du management de Virgin Games, déclare : « Souvent, plus le programmeur est bon, moins il a d'idées. Pour être vraiment bon dans sa partie, il doit être très logique, très méthodique, très minutieux, et ce ne sont pas forcément des qualités propres à l'individu créateur. » C'est pourquoi Virgin Games n'emploie que très peu de programmeurs à plein temps, dont la tâche essentielle consiste à remédier aux déficiences des nombreux programmes que de jeunes adolescents pleins d'espoir envoient chaque semaine à la firme. Le processus est à double sens : des programmeurs qualifiés se voient présenter des idées neuves; des esprits inventifs reçoivent toute l'assistance nécessaire pour coder leurs jeux.

Il est peut-être moins intéressant d'être édité par Virgin que par d'autres compagnies. Tout jeu commercialisé par la compagnie vaut à son auteur une avance de 10 000 à 30 000 F, et des droits d'auteur d'un montant de 7,50 % du prix de vente. Par comparaison, d'autres maisons proclament offrir jusqu'à 25 %. Mais Alexander fait remarquer qu'un quart du chiffre d'affaires est consacré à la promotion du programme, et qu'ainsi les ventes et les revenus de l'auteur sont bien plus importants.

La promotion est une activité dont Virgin n'ignore plus rien. Les techniques si judicieusement utilisées par Richard Branson, trente et un ans, directeur de la firme, dans le domaine de la musique, ont été reprises et appliquées aux logiciels. Les créateurs de jeux sont traités comme des rock stars, et les prospectus accompagnant les cassettes comportent une photographie, et une biographie succincte. Virgin Games démarra en février 1983, faisant paraître des petites annonces dans la presse informatique. Il y eut cinq cents réponses. Aujourd'hui le catalogue comporte déjà quarante-six titres destinés à huit micro-ordinateurs : d'abord le Spectrum, bien sûr, puis le BBC Micro et, non loin derrière, le Commodore 64.

La nouvelle vedette de Virgin Games est Martin Wheeler, qui à quinze ans vient juste de faire paraître deux jeux pour le Spectrum : « Dr. Franky and the Monsters » et « Sorcerers ». Ils sont rédigés en langage machine, et Wheeler les a pourvus d'un graphisme remarquable. Nick Alexander voit une relation profonde entre la micro-informatique telle qu'elle existe aujourd'hui et la rock music d'il y a dix ans; il croit même que la première remplacera la seconde comme activité favorite des jeunes.

Nick Alexander

Richard Branson

LES PASSEPORTS DE L'ART

La splendeur des photographies et la grande qualité, tant littéraire que scientifique, des textes de présentation font de la nouvelle «série rouge» des PASSEPORTS DE L'ART un événement dans le monde de l'édition d'art. Chaque volume fait revivre pour vous les plus beaux monuments et les œuvres des grands artistes.

Avec les PASSEPORTS DE L'ART, partez à la découverte du Mont-Saint-Michel, du Yucatán et de la civilisation des Mayas, de la Cité interdite de Pékin, des villes sacrées de Rome, Kyōto et Jérusalem, des hôtels du Marais, de la Sainte-Chapelle et de Notre-Dame.

La collection des PASSEPORTS DE L'ART vous permet d'entreprendre un magnifique voyage dont les escales sont les chefs-d'œuvre de l'esprit humain.

Une collection riche de 24 volumes de 76 pages entièrement en couleurs.

Le 1er et le 15 de chaque mois chez votre marchand de journaux

EDITIONS ATLAS

EDITIONS ATLAS

les encyclopédies...
les beaux livres...
les reportages de la revue AIR FRANCE atlas

tour maine-montparnasse 33, avenue du maine
75755 paris