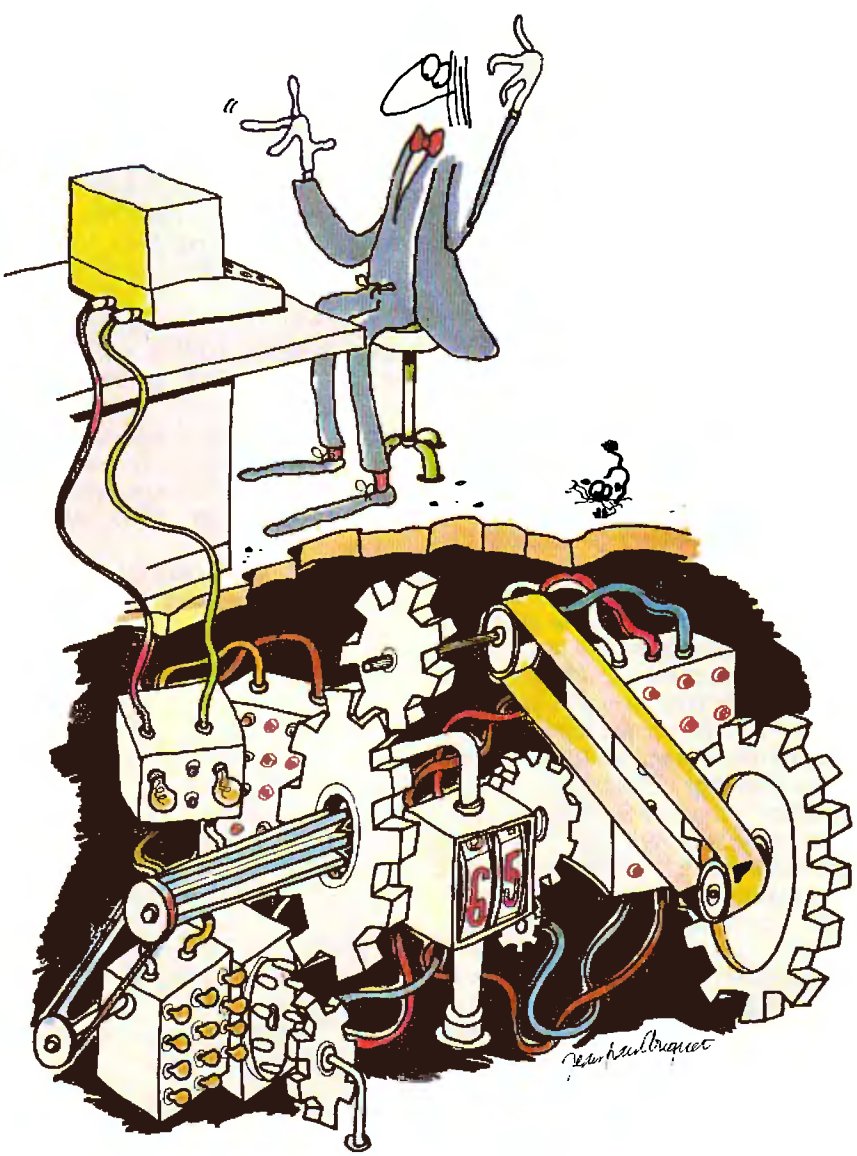
# UN ORDINATEUR, COMMENT ÇA MARCHE ?

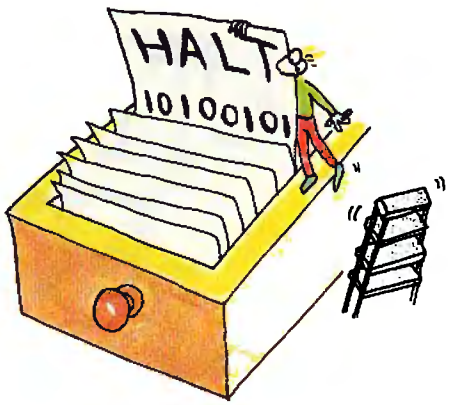
S.V.M. présente l'Ordidactic, un programme pour voir à l'intérieur de votre ordinateur.



**Nous commençons ici une série qui, pour la première fois, permet de comprendre simultanément le fonctionnement d'un ordinateur, sa structure, son langage et ses techniques de programmation, et qui contient un élément unique en son genre : "l'Ordidactic". Grâce à lui, vous visualiserez sans peine sur votre écran de télévision, les composants majeurs d'un ordinateur, et pourrez suivre ainsi pas à pas le déroulement d'un programme. Car l'Ordidactic décompose l'infiniment rapide au rythme de votre compréhension et vous montre l'évolution des registres, de la mémoire, des compteurs, en vous initiant au plus performant et au plus méconnu des langages, le langage machine. C'est donc vraiment un outil de vulgarisation assistée par ordinateur dont le premier objet (et ce ne sera pas le seul) est de vous initier à la technique même qui lui a donné naissance. Ordidactic est un programme qui simule un ordinateur légèrement simplifié, mais très démonstratif et, bien sûr, fonctionnant comme un vrai ! Alors, bon voyage avec S.V.M. au centre du microprocesseur...**

LE LANGAGE MACHINE EST À LA micro-informatique familiale ce qu'est le Graal aux Chevaliers de la Table Ronde : sa quête paraît irréalisable. Or, le langage machine est certainement le plus simple des langages informatiques, sa difficulté apparente résidant dans le fait qu'un certain nombre de concepts de base doivent être assimilés avant d'approcher la partie programmation proprement dite.

Tout d'abord, et pour soulever un coin du voile qui masque ce mystère quasi impénétrable, sachez qu'un ordinateur ne comprend à la base qu'une seule chose : le langage machine. Première objection : comment se fait-il que "PRINT", instruction que mon micro-ordinateur accepte parfaitement, puisse être comprise par ce dernier ? Excellente observation, qui prouve votre vivacité d'esprit, mais comment se fait-il que vous-même compreniez parfaitement le mot "manger" ? Tout simplement parce qu'on vous l'a *appris* ! De la même manière, nos micro-ordinateurs ont appris la signification précise de l'instruction PRINT. La comparaison homme-ordinateur étant toutefois terrain glissant, nous abandonnerons tout de suite ces rapprochements discutables pour faire remarquer que si le micro-ordinateur comprend effectivement PRINT, c'est que, quelque part dans sa mémoire, s'opère la traduction de cette instruction dans un langage qu'il connaît bien.


Feuilletez maintenant ce numéro de S.V.M. et parcourez les publicités : les spécifications techniques comportent pratiquement toujours une rubrique "CPU" ou "microprocesseur central" suivi d'une référence (Z80 A, 6502, etc.). Ces références désignent le "cœur" de chaque micro-ordinateur et définissent le langage machine qui est à la base de la conception de celui-ci, chaque microprocesseur ayant son "parler" particulier. Ainsi, si votre machine favorite parle le Basic, c'est qu'elle dispose d'un précieux auxiliaire appelé "interpréteur", qui traduit votre PRINT en un charabia quasi illisible pour l'être humain mais, par contre, tout à fait clair pour le microprocesseur. Mais j'entends déjà une question pressante se formuler : "Tout cela n'est que théorie ! Comment se présente *matériellement* ce fameux langage machine ?" Au risque de perdre la plupart de nos lecteurs, nous prendrons aujourd'hui l'initiative de dévoiler aux yeux de tous ce spectacle insoutenable qu'est un programme en langage machine. Voici donc un exemple choisi parmi les moins dangereux :

10100100100011111000110001110111001 10.

Les survivants m'auront compris : l'ordinateur n'acceptant à la base que le *binaire*, le langage machine est en réalité une suite de "0" et de "1" ! Vous conviendrez que tout cela n'est guère pratique à manipuler ; aussi, afin de vous éviter de douloureuses heures le nez sur vos écrans, les constructeurs de micro-ordinateurs implantent-ils sur les machines des facilités (le Basic en est une) qui vous permettent de programmer en réalité en langage machine ! Ne refermez pas tout de suite ce journal en disant "je connais le Basic, donc, je connais le langage machine !". Lisez auparavant l'encadré ci-contre. Convaincus ? Eh bien sachez maintenant qu'il existe une facilité *intermédiaire* entre le langage machine et le Basic : c'est le *langage d'assemblage*. En effet, chacun pourra juger que HALT (instruction similaire à notre bon vieux STOP des Basic en langage d'assemblage Z80) est largement plus facile à comprendre que "01110110". Notez au passage que l'instruction HALT est ce qu'il est convenu d'appeler un *mnémonique*, c'est-à-dire une facilité pour l'être humain qui veut manipuler plus facilement notre "01110110" de tout à l'heure. Un programme spécial, appelé "compilateur-assembleur", est par la suite chargé de traduire en binaire ces mnémoniques. Vous l'avez compris : c'est à une étude de ce fameux langage d'assemblage que S.V.M. vous convie maintenant.


## Les langages d'assemblage

Fort de la lecture de ce qui précède, vous êtes certainement maintenant suffisamment au courant pour savoir qu'il existe autant de langages d'assemblage différents que de microprocesseurs. Nous serions donc, en théorie, obligés de choisir l'un d'eux afin d'asseoir notre initiation sur des exemples réels. Mais sur quels critères arrêter notre choix ?

Chaque langage d'assemblage possède en effet ses particularités syntaxiques et ses instructions spécifiques (c'est encore pire que les différences entre un Basic Microsoft et un Basic Sinclair, c'est vous dire !). Aussi, une fois de plus, S.V.M. a-t-il décidé de ne pas céder à la facilité qui consiste à dire : "Afin de vous faire comprendre le langage d'assemblage, nous allons étudier le Z 80 !". Nous avons donc créé de toutes pièces un *"pseudo-assembleur"*, c'est-à-dire un langage d'assemblage qui n'existe nulle part ailleurs que dans le cerveau fou du Processeur Micro, mais qui a le mérite d'aborder la plupart des concepts de base de notre sujet tout en restant extrêmement simple.

Notre langage machine est exécutable bien entendu sur un ordinateur imaginaire, le S001,


sorte d'hyride tenant à la fois du poisson d'avril et du tout nouveau QL de Sinclair. Le pas entre le réel et l'imaginaire est vite franchi : le S001, c'est *votre* micro-ordinateur !...

## Le programme Ordidactic

Nous avons conçu un programme en Basic, exécutable donc sur la plupart des micro-ordinateurs, dont le but avoué est de *simuler* le fonctionnement du S001 ! Mais ne nous méprenons pas : son rôle n'est évidemment que didactique, ce n'est certes pas sa rapidité d'exécution qui le portera au Panthéon des langages de programmation. Le listing, écrit pour un Laser 200 équipé d'une extension 16 Ko, est publié dans cet article. Mais, soucieux de vous éviter une nuit blanche, nous avons pensé vous proposer directement la cassette. Il vous en coûtera 59 F franco de port (le bon de commande est en page 76).

Cette cassette comporte le même programme adapté pour 6 micro-ordinateurs différents : Sinclair ZX 81 + 16 Ko, Sinclair Spectrum 48 Ko, Laser 200 + 16 Ko, Oric-1, 48 Ko, Atmos 48 Ko et TO 7 + 16 Ko. Si vous n'avez pas l'intention d'acquérir cette cassette, reportez-vous alors à la page 79 où nous vous expliquons comment adapter ce programme à votre micro-ordinateur.

Signalons encore que si ce programme n'est pas absolument nécessaire pour suivre le propos et les exemples de cette série d'articles, il est par contre absolument indispensable pour faire fonctionner le logiciel "pseudo-assembleur".


Une fois chargé dans la mémoire de votre micro-ordinateur grâce à l'instruction adéquate (C LOAD "ASM" ou LOAD "ASM" selon la syntaxe de la machine), ce programme se lance automatiquement et affiche un menu comportant 5 options :

1 – Entrée de texte
2 – Exécution pas à pas
3 – Exécution normale
4 – Sauvegarde
5 – Remise à zéro

On s'en doute, en appuyant sur une touche entre 1 et 5, on obtient l'action correspondante !

L'option 1 va vous permettre de lister et d'entrer grâce à un éditeur simplifié des instructions ou des valeurs numériques à raison d'une seule d'entre elles par case mémoire. Nous verrons par la suite ce qu'est exactement une case mémoire.

– Pour obtenir une liste de ce que contient la mémoire du S001, entrez L suivi d'un numéro entre 0 et 99. (L seul signifie L0). L'appui sur la touche RETURN (NEWLINE ou ENTER sur d'autres systèmes) seule permet d'obtenir la page suivante de mémoire.

# DE L'INTÉRÊT DU LANGAGE MACHINE

Le Basic est, certes, un langage pratique et quelquefois très puissant, il n'en reste pas moins lent. De plus, certains Basic n'optimisent pas la place mémoire. Prenons par exemple le cas de DECIBEL (TM). Vous savez, notre fabuleux programme du mois dernier !

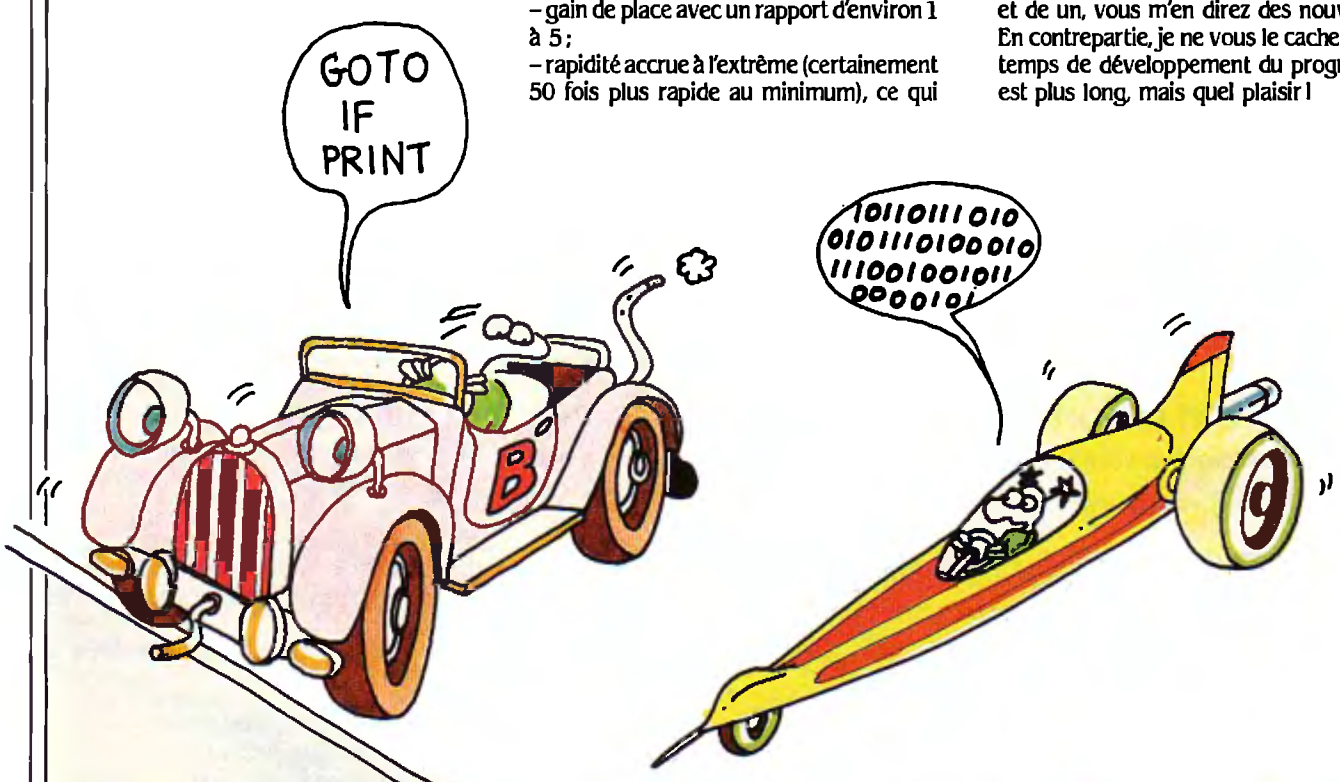
Une version langage machine de ce programme aurait les différences suivantes par rapport au Basic :

– gain de place avec un rapport d'environ 1 à 5 ;

– rapidité accrue à l'extrême (certainement 50 fois plus rapide au minimum), ce qui obligerait le programmeur à inclure des "boucles d'attente" dans le programme principal afin de le ralentir ;

– protection du listing : essayez de comprendre des suites ininterrompues de zéro et de un, vous m'en direz des nouvelles.

En contrepartie, je ne vous le cache pas, le temps de développement du programme est plus long, mais quel plaisir !



– Pour obtenir cette même liste sur imprimante (si vous en possédez une), entrez I.
– Pour retourner au menu principal, entrez F. L'utilisation de En (n est un numéro entre 0 et 99) permet l'entrée de texte en mémoire, en commençant à la base n.

Notez que sera seulement acceptée une instruction S001 valide ou un nombre compris entre 0 et 99.

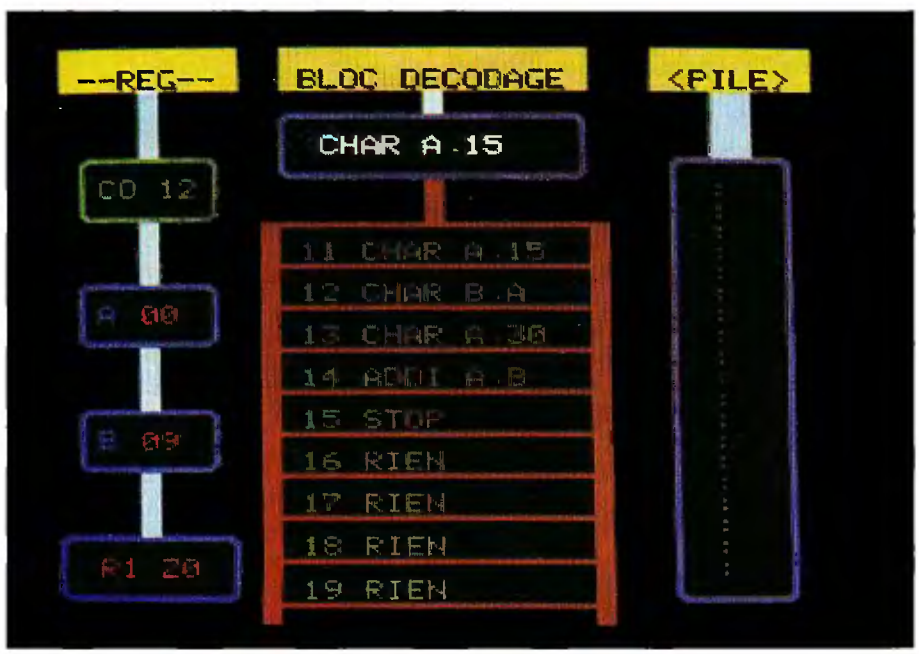
Lorsque vous désirerez en finir avec cette option, entrez une étoile seule (*), vous retournerez alors automatiquement au niveau supérieur. Entraînez-vous à ces différentes options afin de posséder parfaitement le mécanisme de cette partie du programme (pour l'option E, STOP est, par exemple, une instruction valide du S001).

L'option 2 permet d'obtenir (comme son nom l'indique explicitement) une exécution pas à pas des instructions que vous aurez saisies grâce à l'option 1 (sous-option E). Sous vos yeux éblouis, se dessinent alors les différents éléments d'un micro-ordinateur. Ceux-ci sont détaillés dans la suite de l'article, patience... Sachez seulement pour l'instant qu'en appuyant sur une touche quelconque, vous exécuterez *une* instruction S001. La commande * permet de revenir au menu principal.

L'option 3 a le même rôle que l'option 2, mais cette fois sans dessin des différentes parties internes de la machine et sans avoir à presser une quelconque touche lorsque des messages doivent être affichés à l'écran.

L'option 4 vous permet de sauvegarder sur une cassette un fichier contenant le programme que vous avez eu tant de mal à entrer grâce à l'option 1 (sous-option E).

L'option 5 est à utiliser avec précaution, puisqu'elle efface de la mémoire le programme assembleur S001 que vous avez pu entrer. Cette option demande confirmation ; en cas de réponse négative, le retour au menu est automatique. Pour finir, en cas d'accident (par exemple, appui intempestif sur la touche BREAK), GOTO 100 vous ramènera automatiquement au menu principal.


*Appuyez sur la touche RETURN, l'instruction présente dans le bloc de décodage s'exécute, les registres changent, la pile évolue...* ▶

# ORDIDACTIC

*Un outil unique pour comprendre le fonctionnement d'un ordinateur.*

Science et Vie Micro a conçu un programme exclusif pour vous permettre de voir réellement à l'intérieur de l'ordinateur. Ordidactic est un outil unique en son genre pour comprendre simultanément le fonctionnement d'un ordinateur, sa structure, son langage et ses techniques de programmation. Il vous permettra de faire les travaux pratiques de la série d'articles, "Un ordinateur, comment ça marche ?", du Processeur Micro. Attention, la cassette de l'Ordidactic est destinée au possesseur d'un des matériels suivants : Sinclair ZY 81 + 16 Ko, Sinclair Spectrum 48 Ko, Laser 200 + 16 Ko, Oric-1 48 Ko, Atmos 48 Ko et TO 7 + 16 Ko.

Il est possible d'adapter l'Ordidactic à son micro-ordinateur grâce au listing fourni dans ce numéro. Pour obtenir la cassette, il vous suffit de nous retourner le bon de commande ci-dessous accompagné de votre règlement.


## BON DE COMMANDE

A retourner accompagné de votre règlement à S.V.M., 5, rue de la Baume, 75008 Paris.

Veuillez m'adresser _____ cassette(s) Ordidactic.
Ci-joint mon règlement de ____ x 59 F (50 F TTC + 9 F participation de port) par ☐ chèque bancaire, ☐ chèque postal, ☐ mandat-lettre.

Nom : _____ _____ Prénom : _____ _____
Adresse : _____ _____ _____ _____
Code postal : _____ _____ Ville : _____ _____
Type de matériel : _____ _____ _____ _____

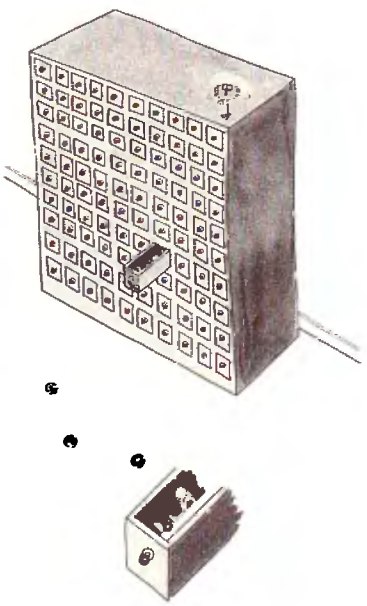
## BULLETIN RÉPONSE

A retourner à S.V.M., 5, rue de la Baume, 75008 Paris.
Nom : _____ _____ Prénom : _____ _____
Adresse : _____ _____ _____ _____
Code postal : _____ _____ Ville : _____ _____
La technique de vulgarisation assistée par ordinateur peut s'adapter à d'autres domaines comme la biologie, l'astronomie, la physique, les statistiques... la couture, etc.
Quels sont les sujets que vous souhaiteriez pouvoir aborder par la technique de l'Ordidactic ?
1 : _____ _____ _____ _____
2 : _____ _____ _____ _____
3 : _____ _____ _____ _____
Sur quel ordinateur ?

Voilà. Vous en savez maintenant suffisamment pour utiliser le programme Ordidactic avec profit. Il est donc temps de passer aux choses sérieuses.

## L'ordinateur S001

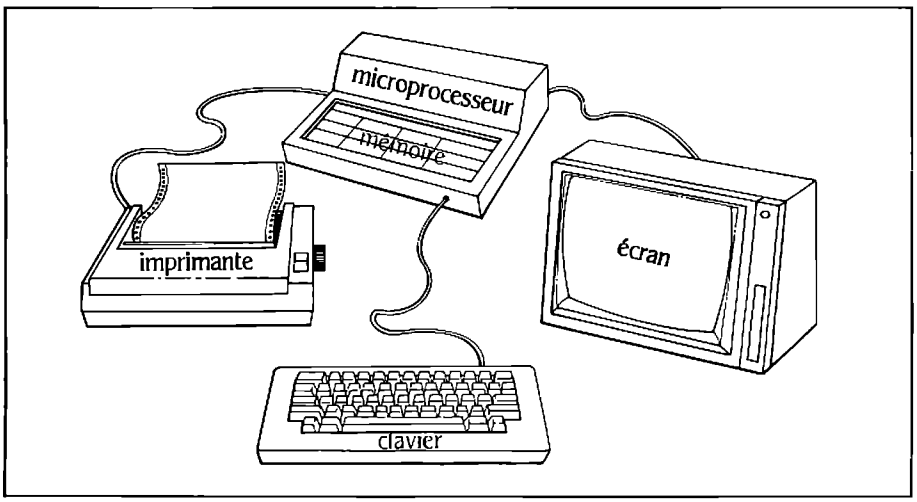
Comme nous l'avons vu précédemment, vous aurez peu de chance de trouver aujourd'hui le S001 chez votre revendeur



habituel. Toutefois, comme tout ordinateur qui se respecte, celui-ci possède des caractéristiques techniques, un processeur central, des périphériques, etc. – le tout ne pouvant fonctionner sans mémoires, qui sont, rappelons-le, généralement de deux types, la morte (ou ROM) et la vive (ou RAM).


Le S001 ne possède qu'un seul type de mémoire : la vive, divisée en 100 "cases". Cette mémoire, comme la plupart des mémoires vives est volatile, c'est-à-dire que la mise hors tension de l'appareil vous en fait perdre irrémédiablement le contenu. A chacune de ces cases est associé un numéro de 0 à 99, ou "adresse". Lorsque l'on désire désigner une case particulière, il suffit d'indiquer son numéro à la machine.

Cette opération, omniprésente dans la programmation en langage machine, s'appelle *l'adressage*. Nous verrons plus tard très précisément les différents modes d'adressage existants. Pour définir brièvement ce qu'est un mode d'adressage, imaginons que vous désiriez injurier sauvagement l'un de vos contemporains, votre percepteur par exemple. Plusieurs méthodes (ou modes) sont à votre disposition : la poste, le téléphone ou encore la confrontation directe. Quel que soit le moyen employé, vous pouvez être sûr que l'individu visé aura reçu vos invectives et vous en saura gré.

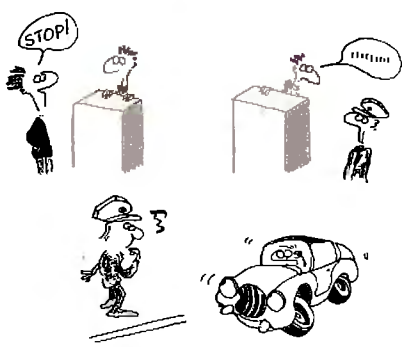



Outre la mémoire, le S001 incorpore un microprocesseur central, unique et inédit (soyez certain qu'il n'a pas été fabriqué à Hong-Kong, celui-là !). Sa référence est tenue secrète (la preuve : personne ne la connaît !).

Ces deux éléments – microprocesseur et mémoire – mis ensemble dans un boîtier esthétique et reliés par diverses connections, ne suffisent toutefois pas à donner un ordinateur en état de marche : celui-ci doit encore pouvoir communiquer avec l'extérieur grâce à des *périphériques* : clavier, écran de visualisation et imprimante. Nous obtiendrons donc le schéma ci-dessus.

Chacun de ces périphériques peut indifféremment être *lu* ou *écrit* : il va de soi toutefois que lire l'*écran* et écrire sur le *clavier* ne donnent pas de résultats grandioses. De la même manière qu'une case mémoire porte un numéro et peut être lue ou écrite, à chaque périphérique est associé un nombre, qui correspond à un "port d'entrée-sortie". Le S001 est capable là aussi, d'*adresser* l'un de ces


## L'ÉCHELLE DES LANGAGES




Vous n'êtes pas sans savoir que de nombreux langages informatiques différents existent de par le monde. Ceux-ci peuvent être classés sur une échelle en fonction de leur puissance. A la base sont les langages non-évolués, très proches de la machine, et vers le haut sont les langages évolués qui se rapprochent le plus du langage humain. Au maximum sont les langages de l'avenir, qui vous permettront de converser dans la langue de Molière avec votre ordinateur favori...

Echelle d'évolution

- ?
- Pascal
- Forth
- Basic évolués (type Electron)
- Basic standard (type Commodore)
- Macro-assembleurs
- Langage d'assemblage (mnémonique)
- Langage machine (binaire)

"ports d'entrée-sortie" afin de communiquer avec ses périphériques.

Après avoir planté le décor de notre action, intéressons-nous maintenant au cœur du système, car c'est là que résident les insondables mystères du langage machine : nous avons nommé le microprocesseur central et les différents éléments qui le composent. A cet effet, examinons le programme Ordidactic et sélectionnons l'option 2 du menu. Comme tout à l'heure, un ensemble de graphismes se forme sous nos yeux (moins éblouis quand même, nous commençons à être blasés !). Ceux-ci représentent les différentes parties internes de notre ordinateur. Trois éléments principaux se détachent : *les registres, la mémoire* (en dessous du "bloc de décodage") et *la pile*. Une


partie plus spécifique au microprocesseur concerne les registres et le bloc de décodage.

**Les registres.** Il s'agit de petites portions de mémoire vive *interne* au microprocesseur, et par lesquelles transitent les données à traiter. Vous vous en souvenez certainement, une case mémoire particulière ne peut contenir qu'une instruction *ou* une valeur numérique comprise entre 0 et 99. Imaginons maintenant que nous désirons additionner les contenus de deux cases contenant des valeurs numériques. Eh bien, sachez ceci : ces deux nombres, pour être additionnés, doivent obligatoirement transiter par un ou plusieurs registres, chacun ayant un rôle particulier. C'est dire l'importance de ces petites choses apparemment innocentes. Elles sont au microprocesseur ce que le diamant est à la tête de lecture de votre platine tourne-disque : sans registre, finie la musique...

**CO : Compteur ordinal.** Son rôle est de mémoriser l'adresse (vous vous souvenez certainement : chaque case mémoire possède une adresse...) de la prochaine instruction à exécuter. Par exemple, comme vous pouvez le constater sur votre écran, la case mémoire d'adresse 1 contient l'instruction particulièrement ésotérique RIEN. Comme vous n'avez pas encore appuyé sur une quelconque touche du clavier, c'est l'instruction contenue par la case 1 qui est en cours d'exécution. Toutefois, prévoyant, le compteur ordinal affiche gaillardement la valeur 2. A moins qu'un mauvais plaisant ne modifie ce contenu, la prochaine instruction à exécuter sera donc celle d'adresse mémoire 2.

**L'accumulateur** porte le doux nom de "A" (allez savoir pourquoi ?). C'est par lui que transitent toutes les données qui seront utilisées pour les calculs, les comparaisons, etc. Vous pouvez d'ailleurs observer que, bien que n'ayant encore rien fait, celui-ci contient déjà une valeur comprise entre 0 et 99. Nous verrons tout à l'heure l'explication de ce phénomène. Afin de rendre plus compréhensible le fait que toutes les opérations passent obligatoirement par l'accumulateur, hasardons-nous à faire une comparaison avec le Basic. Imaginez qu'un revendeur mal intentionné vous vende le dernier cri en matière de micro-informatique : il s'agit d'un Fpx 327 bis (modifié 53). Or, arrivé chez vous, vous constatez avec horreur que son Basic ne comporte qu'une seule variable ! Pour effectuer l'opération 3 + 3 et conserver le résultat, vous serez donc obligé de taper :

```
10      A = 3
20      A = A + 3
```

Ce qui vous donne une idée assez réaliste de la gymnastique que vous serez obligé de faire pour effectuer vos opérations arithmétiques en langage machine.

**Le registre annexe B** est dit annexe par rapport à l'accumulateur. En effet, ses possibilités sont beaucoup plus réduites que celles de ce dernier : en particulier, il n'est même plus question d'effectuer avec lui des opérations arithmétiques, son rôle étant, dans ce cadre, réduit à conserver de manière pratique des données numériques. Si nous reprenons ici le tristement célèbre exemple du Fpx 327 bis


(nouvelle version, puisque son Basic autorise dorénavant deux variables – à propos, le constructeur échange la mémoire morte des anciens modèles pour – seulement – 1 478,20 F TTC (+ port) ! suivez mon regard...), nous aurons :

```
10      A = 3
20      B = A
30      A = A + B
```

Toutefois, consolons-nous : nous verrons qu'il est en fait très utilisé dans le mode d'adressage dit *"indirect"*, ainsi que pour la recherche en table.

**Les indicateurs d'état R et Z,** registres d'un type particulier, ne peuvent prendre que les valeurs 0 ou 1. Malgré ces restrictions, leur


rôle reste très important : ils donnent sur les résultats d'opérations arithmétiques et sur certaines instructions des renseignements appréciables.

Z (zéro) est l'indicateur de passage à zéro. Si, par exemple, A-B = 0, alors, Z = 1 ; c'est simple, non ? Le tout est de bien comprendre que Z indique un résultat égal à zéro, lorsqu'il est positionné à 1. Dans tous les autres cas, Z se positionne à 0.

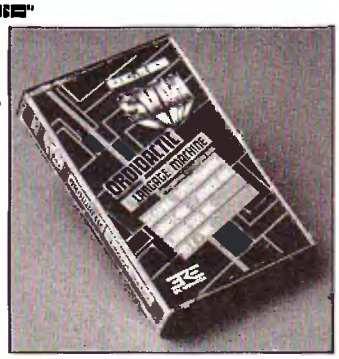
R (retenue) est l'indicateur dit de "débordement". Expliquons-nous : il est maintenant temps de vous parler de l'une des particularités du S001. Celui-ci (et vous l'avez peut-être deviné) ne peut manier des nombres qu'à l'intérieur d'une fourchette allant de 0 à 99. Fort bien, me direz-vous, mais que se passe-t-il lorsque, le registre A étant à 99, on décide de lui ajouter la valeur 1 ? Eh bien essayons ! Sur le papier, tout du moins, car nous ne connaissons pas encore suffisamment d'instructions S001 pour l'essayer sur notre merveilleuse machine. Ainsi 99 + 1 font, pour chacun, 100. Or, le S001 ne peut aller que jusqu'à la valeur 99. La solution est simple : lorsque le résultat est supérieur à 99, il suffit de lui soustraire 100 et de positionner l'indicateur R à 1, qui indique alors effectivement une *retenue*. Dans notre exemple, le résultat sera donc 0, et R = 1 indiquera alors qu'un *débordement* a eu lieu. Essayons d'autres valeurs : 56 + 67 = 123, 123 – 100 = 23, qui est donc votre résultat final.

*(Suite page 81)*

# LE PROGRAMME ORDIDACTIC SUR LASER 200

Si vous êtes un spécialiste du Basic Microsoft, vous trouverez certainement ce programme peu élégant. Sachez cependant qu'il a été conçu de manière à être facilement adapté sur d'autres machines
En particulier, les instructions PRINT possèdent leur équivalent sur d'autres micro-ordinateurs,
par exemple: PRINT 2*32+14, "MENU" se traduit par:
PRINT AT 2,14; "MENU" en Basic Sinclair,
LOCATE 14,2,1: PRINT "MENU" sur T07,
PLOT 14,2, "MENU" sur Oric-1.
De plus, dans la mesure du possible, des instructions souvent absentes sur plusieurs micro-ordinateurs ont été omises volontairement (cas du ELSE par exemple ou du ON... GOTO). Surtout n'oubliez pas, si le programme ne fonctionne pas du premier coup, de bien vérifier que toutes les instructions sont présentes avant de téléphoner!

```
5 '  ***** ORDIDACTIC ******
10 '    1- LANGAGE MACHINE
15 '  LASER 200 16K - V0.1
20 ' (C) S.V.M ET E.SARTORI
30 '
40 '
45 POKE30744,1:CLEAR2000
50 CLS:GOSUB8000 ' INITIALISATION
90 CLS
100 ' MENU----------
110 '
120 PRINT@2*32+14,"[illegible]"
130 PRINT@4*32+5,"[illegible]-ENTREE DE TEXTE"
140 PRINT@6*32+5,"[illegible]-EXECUTION PAS A PAS"
150 PRINT@8*32+5,"[illegible]-EXECUTION NORMALE"
160 PRINT@10*32+5,"[illegible]-GESTION FICHIER"
170 PRINT@12*32+5,"[illegible]-REMISE A ZERO"
180 PRINT@15*32+10,"[illegible]";
190 R$=INKEY$:IFR$<"1"ORR$>"5"THEN190
200 CLS
202 IFR$="1"THENGOSUB1000:GOTO230
204 IFR$="2"THENGOSUB2000:GOTO230
206 IFR$="3"THENGOSUB3000:GOTO230
208 IFR$="4"THENGOSUB4000:GOTO230
210 IFR$="5"THENGOSUB5000:GOTO230
230 CLS
240 GOTO100
1000 ' ENTREE TEXTE----------
1010 '
1080 CLS
1090 PRINT"[illegible]"
1100 FORI=D+1TOD+12
1105 IFI>=101THENPRINT:GOTO1130
1110 PRINTLEFT$(T$(I),2);" ==> ";MID$(T$(I),3,20)
1130 NEXTI
1140 PRINT:INPUT"[illegible] ";R$
1150 IFR$=""THENR$="L"+STR$(D+12)
1155 R1$=LEFT$(R$,1)
1160 IFR1$="L"THEND=ABS(VAL(MID$(R$$,2,20)))
1170 IFR1$="I"THENCOPY
1180 IFR1$="F"THENGOTO1900
1190 IFR1$="E"THENGOSUB8600
1200 GOTO1080
1900 RETURN
2000 ' EXEC PAS A PAS----------
2010 '
2020 CLS
2100 PRINT@0,"[illegible]"
2110 PRINT@32+8,"[illegible]";
2120 PRINT@32*2+8,"[illegible]";
2130 PRINT@32*3+8,"[illegible]";
2150 FORI=1TO14
2160 PRINT@32*I+24,"[illegible]";
2170 NEXTI
2180 PRINT@15*32+24,"[illegible]";
2190 FORI=4TO12STEP2
2200 PRINT@32*I+8,"[illegible]";
2210 PRINT@32*(I+1)+8,"[illegible]";
2220 NEXTI
2225 PRINT@32*14+8,"[illegible]";
2226 PRINT@32*15+8,"[illegible]";
2230 FORI=2TO12STEP4
2235 PRINT@(I-1)*32+3,"[illegible]";
2240 PRINT@I*32,"[illegible]";
2245 PRINT@(I+1)*32,"[illegible]";
2250 PRINT@(I+2)*32,"[illegible]";
2260 NEXTI
2270 PRINT@3*32+1,"[illegible]";
2275 PRINT@7*32+1,"[illegible]";
2280 PRINT@11*32+1,"[illegible]";
2285 PRINT@13*32,"[illegible]";
2290 PRINT@14*32,"[illegible]";
2295 PRINT@15*32,"[illegible]";
2390 CO=0
2392 FP=1
2395 PT=0
2396 ST=0
2400 PRINT@14*32+1,RIGHT$(STR$(R),1)
2405 PRINT@14*32+5,RIGHT$(STR$(Z),1)
2410 FORI=1TO3
2420 PRINT@32*3+4,LEFT$(T$(CO+2),2)
2421 PRINT@32*7+4,LEFT$(T$(A+1),2)
2423 PRINT@32*11+44,LEFT$(T$(B+1),2)
2430 FORI=CO+1TOCO+6
2440 PP=(I-CO)*2+2
2445 N=I+100*(I>100)
2450 PRINT@PP*32+8,CHR$(ASC(LEFT$(T$(N),1))+192)
2451 PRINT@PP*32+9,CHR$(ASC(MID$(T$(N),2,1))+192)
2452 PRINT@PP*32+11,"  "
2453 PRINT@PP*32+11,MID$(T$(N),3,20)
2460 NEXTI
2470 D$=MID$(T$(CO+1),3,20)
2475 PRINT@32*2+10,"                "
2480 PRINT@32*2+10,D$
2500 IFINKEY$<>""THEN2500
2501 IFINKEY$=""THEN2501
2502 IFINKEY$="*"THENRETURN
2505 PRINT@32*2+10,"[illegible]"
2510 GOSUB7000
2600 IFCO=100THENCO=0
2605 IFST=1THENRETURN
2610 GOTO2400
3000 ' EXEC. NORMALE----------
3010 '
3020 CO=0
3025 FP=0
3030 PT=0
3040 ST=0
3050 D$=MID$(T$(CO+1),3,20)
3051 IFINKEY$<>""THEN3051
3052 IFINKEY$="*"THENRETURN
3055 GOSUB7000
3060 IFCO=100THENCO=0
3065 IFST=1THENRETURN
3070 GOTO3050
4000 ' GESTION FICHIER
4010 '
4020 PRINT:PRINT:PRINT
4030 PRINT"    [illegible]  "
4040 PRINT:PRINT:PRINT
4050 PRINT  "  [illegible]- SAUVEGARDE PROGRAMME S001"
4060 PRINT
4070 PRINT"  [illegible]- LECTURE PROGRAMME S001"
4080 PRINT:PRINT
4090 PRINT"        [illegible]"
4100 IFINKEY$<>""THEN4100
4110 R$=INKEY$:IFR$<>"1"ANDR$<>"2"ANDR$<>"*"THEN4110
4115 IFR$="*"THENRETURN
4120 IFR$="2"THENFORI=1TO101:INPUT#"S001",T$(I):NEXTI:RETURN
4130 CLS:PRINT@32*6+1,"[illegible]"
4140 PRINT@32*8+12,"[illegible]"
4150 IFINKEY$<>""THEN4150
4160 IFINKEY$=""THEN4160
4165 PRINT@32*12+6,"[illegible]"
4170 FORI=1TO101:PRINT#"S001",T$(I):NEXTI
4900 RETURN
5000 ' REMISE A ZERO----------
5010 PRINT@7*32+5,"[illegible] ":PRINT
5020 IFINKEY$<>""THEN5020
5025 R$=INKEY$:IFR$=""THEN5025
5030 IFR$="O"THENCLEAR1000:GOSUB8000
5040 CLS:GOTO100
6000 ' CONTROLE SYNTAXE--------
6010 '
6020 ER=0
6030 X$=R$
6040 FORH=1TOLEN(X$)
6050 IFMID$(X$,H,1)>="0"ANDMID$(X$,H,1)<="9"THEN6500
6060 NEXTH
6070 GOTO6600
6500 IFNOT(MID$(X$,H+1,1)>="0"ANDMID$(X$,H+1,1)<="9")THEN6550
6510 X$=LEFT$(X$,H-1)+"*"+MID$(X$,H+2,20)
6520 GOTO6600
6550 X$=LEFT$(X$,H-1)+"*"+MID$(X$,H+1,20)
6600 BI=0
6620 BS=43
6630 V=21
6640 IFX$=I$(V)THENRETURN
6690 IFX$)I$(V)THEN6730
6700 BS=V
6710 V=V-((BS-BI)/2)
6720 GOTO6750
6730 BI=V
6740 V=V+((BS-BI)/2)
6750 A1=INT(BS-BI)
6760 IFA1=0THEN6780
6770 GOTO6640
6780 ER=1
```




*(Suite du programme page 80)*

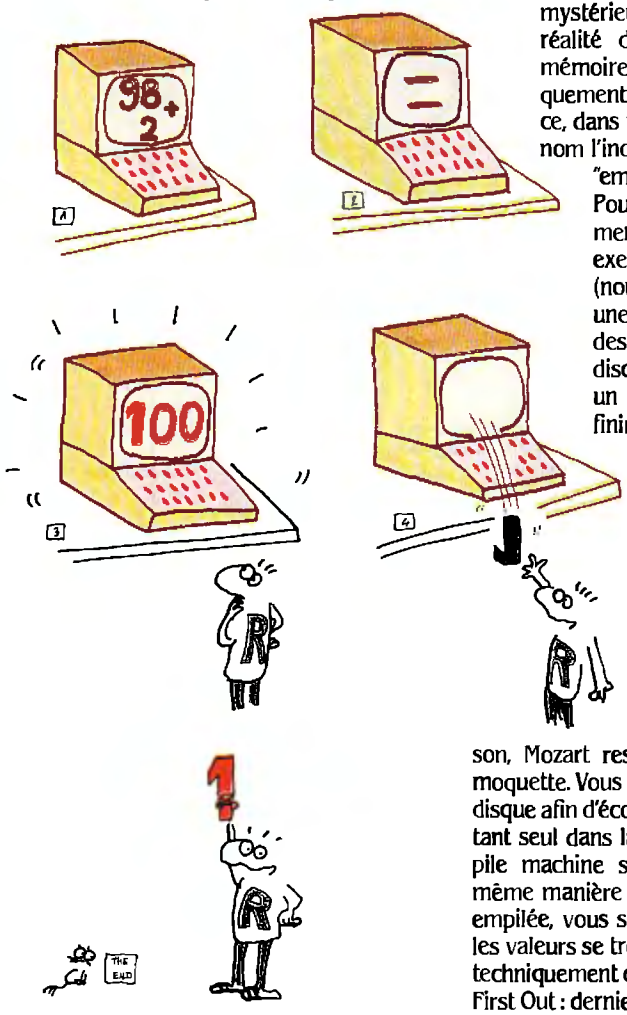
```
6790 RETURN
7000 ' DECODE ET EXECUTE-------
7010 '
7020 FORI=0TO16
7030 IFLEFT$(D$,4)=MID$(M$,I*4+1,4)THEN7050
7035 NEXTI
7040 GOTO7980 ' NOMBRE
7050 IFI=0THEN7100
7051 IFI=1THEN7150
7052 IFI=2THEN7200
7053 IFI=3THEN7250
7054 IFI=4THEN7300
7055 IFI=5THEN7350
7060 IFI=6THEN7400
7061 IFI=7THEN7450
7062 IFI=8THEN7500
7063 IFI=9THEN7550
7064 IFI=10THEN7600
7065 IFI=11THEN7650
7066 IFI=12THEN7700
7067 IFI=13THEN7750
7068 IFI=14THEN7800
7150 ' COMP==================
7155 GOSUB9000:R=-((A-X)<0):Z=-((A-X)=0):GOTO7980
7200 ' DECR==================
7210 A=A-1:GOSUB9100:GOTO7980
7250 ' DEPI==================
7260 IFPT<1THENP(PT)=RND(99)
7275 IFMID$(D$,6,1)="A"THENA=P(PT)
7278 IFMID$(D$,6,1)="B"THENB=P(PT)
7280 IFPT>0THENGOSUB9200
7284 GOTO7980
7300 ' DIVI==================
7310 GOSUB9000
7315 IFX=0THENR=1:Z=0:A=99:GOTO7330
7320 A=INT(A/X)
7325 GOSUB9100
7330 GOTO7980
7350 ' ECRI==================
7355 IFMID$(D$,7,1)="0"THENPRINTCHR$(A);
7360 IFMID$(D$,7,1)="1"THENLPRINTCHR$(A);
7370 GOTO7980
7400 ' EMPI==================
7405 IFPT=14THENGOTO7980
7410 R$=MID$(D$,6,1)
7420 GOSUB9300
7425 GOTO7980
7450 ' EXEC==================
7455 IFMID$(D$,6,1)="R"ANDR=0THEN7480
7456 IFMID$(D$,6,1)="Z"ANDZ=0THEN7480
7460 R$=STR$(CO+1)
7470 IFPT<14THENGOSUB9300
7480 GOSUB9400
7490 GOTO7990
7500 ' INCR==================
7510 A=A+1
7520 GOSUB9100
7530 GOTO7980
7550 ' LIRE==================
7555 IFMID$(D$,9,1)<>"2"THEN7560
7556 R$=INKEY$:IFR$=""THENA=0
7557 IFR$<>""THENA=ASC(R$)
7560 GOTO7980
7600 ' MULT==================
7610 GOSUB9000
7620 A=A*X
7625 GOSUB9100
7630 GOTO7980
7650 ' RETR==================
7655 IFD$="RETR R"ANDR=1THEN7670
7657 IFD$="RETR Z"ANDZ=1THEN7670
7660 IFD$="RETR"THEN7670
7665 GOTO7980
7670 IFPT<1THENCO=RND(99):GOTO7680
7675 CO=P(PT):GOSUB9200
7680 GOTO7990
7700 ' RIEN==================
7710 GOTO7980
7750 ' SAUT==================
7760 GOSUB9400
7770 GOTO7990
7800 ' SOUS==================
7810 GOSUB9000
7820 A=A-X
7825 GOSUB9100
7830 GOTO7980
7850 ' STOP==================
7860 ST=1
7870 GOTO7980
7900 ' CHAR==================
7910 IFMID$(D$,6,1)<>"A"THEN7960
7915 GOSUB9000
7920 IFMID$(D$,8,1)="("ANDMID$(D$,9,1)<>"B"THENGOSUB9360
7925 A=X
7930 GOTO7980
7960 IFMID$(D$,6,1)="B"THENB=A
7965 IFMID$(D$,6,2)="(B"THENT$(B+1)=LEFT$(T$(B+1),2)+STR$(A)
7966 R$=MID$(D$,7,1):NO=VAL(MID$(D$,7,20))+1
7967 IFR$>="0"ANDR$<="9"THENT$(NO)=LEFT$(T$(NO),2)+STR$(A)
7980 CO=CO+1
7990 RETURN
8000 ' INITIALISATION---------
8010 '
8015 PRINT@32*7,"[illegible]"
8020 DIMI$(42):FORI=1TO42:READI$(I):NEXTI
8025 DIMT$(101)
8030 FORI=0TO99
8032 IFI<=9THENR$="0"ELSER$=""
8035 T$(I+1)=R$+MID$(STR$(I),2,2)+"RIEN"
8050 NEXTI
8055 T$(101)="00"
8060 M$="ADDICOMPDECRDEPIDIVIECRIEMPIEXECINCRLIREMULTRETRRIEN"
8065 M$=M$+"SAUTSOUSSTOPCHAR"
8070 D=0
8080 Z=RND(2)-1:R=RND(2)-1:A=RND(99):B=RND(99)
8120 DIMP(20):P(0)=RND(99)
8500 RETURN
8600 ' ENTREE TEXTE-----------
8610 '
8620 I=ABS(VAL(MID$(R$,2,20)))
8625 IFI>=100THEN8800
8630 D=I
8635 CL=-5
8640 GOSUB8900
8650 PRINT@P-8,LEFT$(T$(I+1),2);:PRINT" ===> ";
8652 PRINT@14*32,"[illegible]"
8653 PRINT@14*32+13,"";:INPUTR$
8670 IFR$="*"THEN8800
8671 IFR$=""THENR$="RIEN"
8672 GOSUB6000
8673 IFER=0THEN8679
8674 PRINT@P,"[illegible]":P=P+32
8675 GOTO8720
8679 T$(I+1)=LEFT$(T$(I+1),2)+R$
8690 PRINT@P,MID$(T$(I+1),3,20)+"      "
8700 I=I+1:P=P+32
8710 IFI>99THENI=99
8720 CL=CL+1
8730 IFCL<=5THENGOTO8650
8745 GOSUB8900
8750 FORJ=-(I-5)*((I-5))0) TO -(I-1)*((I-1))0)
8760 PRINT@P-8,LEFT$(T$(J+1),2);
8765 PRINT" ===> ";MID$(T$(J+1),3,20)
8766 P=P+32
8770 NEXTJ
8780 CL=0
8790 GOTO8650
8800 RETURN
8900 ' ENTETE ENTREE----------
8905 '
8910 CLS
8920 PRINT"[illegible]"
8930 P=72:RETURN
9000 ' GROUPE OPER------------
9010 '
9050 R$=MID$(D$,8,1)
9055 IFR$="B"THENX=B:RETURN
9056 IFR$="("THENX=VAL(MID$(T$(B+1),3,20)):RETURN
9060 X=VAL(MID$(D$,8,20))
9090 RETURN
9100 ' TESTE DEPA CAPA--------
9110 '
9120 Z=-(A=0)
9125 D$=LEFT$(D$,4):IFD$="DIVI"THENR=0:GOTO9170
9126 IFD$="ADDI"ANDA>99THENA=A-100:R=1:GOTO9170
9128 IFD$="SOUS"ANDA<0THENA=100+A:R=1:GOTO9170
9130 IFA<0ORA>99THENGOTO9140
9135 R=0
9136 GOTO9170
9140 R$=MID$(STR$(ABS(A)),2,20)
9150 A=VAL(RIGHT$("00"+R$,2))
9160 R=1
9170 RETURN
9200 ' DEPILE-----------------
9210 IFFP=0THENGOTO9220
9212 IFPT<>1THENPRINT@(16-PT)*32+25,">"
9214 PRINT@(15-PT)*32+27,"  ";
9216 PRINT@(15-PT)*32+25," ";
9220 PT=PT-1
9230 RETURN
9300 ' EMPILE-----------------
9310 PT=PT+1
9320 IFR$="A"THENP(PT)=A
9321 IFR$="B"THENP(PT)=B
9330 IFR$<>"A"ANDR$<>"B"THENP(PT)=VAL(R$)
9340 IFFP=0THENRETURN
9345 PRINT@(15-PT)*32+25,">"
9346 PRINT@(15-PT)*32+27,LEFT$(T$(P(PT)+1),2)
9347 IFPT<>1THENPRINT@(16-PT)*32+25," "
9350 RETURN
9360 ' INDIRECT---------------
9365 X=VAL(MID$(T$(VAL(MID$(D$,9,20))+1),3,20))
9370 RETURN
9400 ' BRANCHE----------------
9410 IFMID$(D$,6,1)="Z"ORMID$(D$,6,1)="R"THEN9440
9420 CO=VAL(MID$(D$,6,20))
9430 RETURN
9440 IFMID$(D$,6,1)="Z"ANDZ=1THENCO=VAL(MID$(D$,8,20)):RETURN
9445 IFMID$(D$,6,1)="R"ANDR=1THENCO=VAL(MID$(D$,8,20)):RETURN
9450 CO=CO+1
9460 RETURN
9500 DATA"*","ADDI A.(B)","ADDI A.*","ADDI A.B","CHAR (*).A"
9510 DATA"CHAR (B).A","CHAR A.(*)","CHAR A.(B)","CHAR A.*"
9520 DATA"CHAR A.B","CHAR B.A","COMP A.(B)","COMP A.*"
9530 DATA"COMP A.B","DECR A","DEPI A","DEPI B","DIVI A.(B)"
9540 DATA"DIVI A.*","DIVI A.B","ECRI (*).A","EMPI A","EMPI B"
9550 DATA"EXEC *","EXEC R.*","EXEC Z.*","INCR A","LIRE A.(*)"
9560 DATA"MULT A.(B)","MULT A.*","MULT A.B","RETR","RETR R"
9570 DATA"RETR Z","RIEN","SAUT *","SAUT R.*","SAUT Z.*"
9580 DATA"SOUS A.(B)","SOUS A.*","SOUS A.B","STOP"
```

*(Suite de la page 78)*

Comme dans le cas de l'indicateur Z, en cas de *non-débordement* d'opération arithmétique, R se positionne à zéro.

Nous en avons maintenant fini avec les registres et leurs dépendances.

**Le bloc de décodage.** Cette partie spéciale de notre microprocesseur symbolise le




traitement de chaque information. Chaque instruction en provenance de la mémoire et qui doit être exécutée, transite obligatoirement par lui : son rôle est d'analyser et de décoder les ordres en vue de leur exécution. Il contient systématiquement l'instruction en cours, c'est-à-dire celle contenue par la case mémoire d'adresse C0-1 (le compteur ordinal pointant toujours sur la prochaine case à lire...).

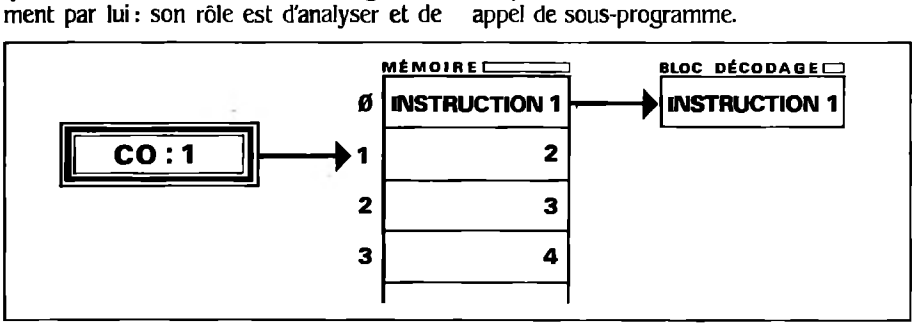
Notez que l'information contenue par le bloc de décodage étant un double parfait de celle d'adresse C0-1, l'instruction correspondante est donc à cet instant en *double* dans la mémoire (prise au sens large et non plus seulement les cases 0 à 99) de notre ordinateur.

## La pile machine

Nous avons donc passé en revue toutes les parties internes du microprocesseur central. Voyons maintenant un élément aussi utile que mystérieux : la pile machine. Il s'agit en réalité d'une partie spéciale de la mémoire, prévue pour recevoir uniquement des données numériques, et ce, dans un certain ordre. Comme son nom l'indique, les informations y sont "empilées" les unes sur les autres. Pour comprendre le fonctionnement d'une pile, prenons un exemple concret. Vous possédez (nous l'avons vu tout à l'heure) une platine tourne-disques, donc des disques. Posez sur le sol un disque de Mozart, par-dessus lui un disque des Beaties et pour finir, au-dessus des deux précédents, le dernier Michael Jackson. Vous constaterez alors avec émotion que, pour écouter le disque de Mozart, vous serez obligé d'enlever d'*abord* Michael Jackson, *puis* les Beatles. De la même manière, pour écouter les Beatles, vous devrez ôter Michael Jackson, Mozart restant finalement seul sur la moquette. Vous aurez donc "dépilé" le premier disque afin d'écouter le second, le dernier restant seul dans la pile. Eh bien sachez qu'une pile machine se manie exactement de la même manière : pour lire la dernière valeur empilée, vous serez obligé de *dépiler* toutes les valeurs se trouvant au-dessus. On appelle techniquement ce type d'accès "LIFO" (Last In, First Out : dernier entré, premier sorti). La pile est généralement utilisée dans deux cas :

1 – pour stocker temporairement des données numériques de façon pratique et rapide ;

2 – pour stocker l'adresse de retour lors d'un appel de sous-programme.




Prenons pour le point n° 2 l'exemple d'un programme Basic comportant un appel de sous-programme :

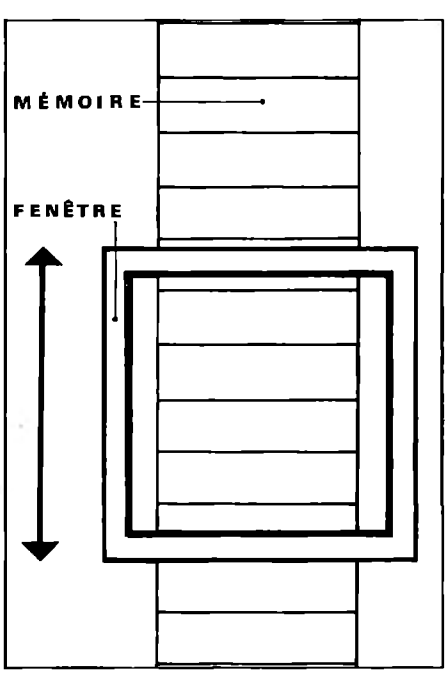
```
10    GOSUB 100
20    PRINT "RETOUR"
30    END
100   PRINT "SOUS-PROGRAMME"
110   RETURN
```

Vous savez certainement qu'au moment de l'exécution de ce programme, le "RETURN" de la ligne 110 nous fera revenir à la ligne qui suit immédiatement le "GOSUB 100", c'est-à-dire en 20. Pourquoi ? Tout simplement parce que ce GOSUB a deux actions :

– *Empiler* le numéro de ligne qui suit immédiatement (c'est-à-dire 20),

– Exécuter un saut à la ligne 100.

Le rôle de RETURN est donc évident : cette instruction "dépile" la première valeur trouvée en haut de la pile et utilise cette valeur pour effectuer un "GOTO". Essayez cette méthode avec plusieurs sous-programmes imbriqués et vous constaterez qu'elle explique parfaitement pourquoi RETURN fait revenir le programme en dessous du dernier GOSUB exécuté à l'exception de tout autre. Nous verrons donc par la suite qu'il est primordial de comprendre et d'avoir constamment à l'esprit le fonctionnement de la pile système lorsque nous programmerons en langage machine : une erreur de manipulation conduit en général aux plus somptueux "plantages" qui soient.




Pour finir, sous le bloc de décodage, apparaît une partie de la mémoire. Ceci permet de visualiser rapidement quelles sont les pro-

chaines instructions à exécuter et de constater immédiatement les effets de la modification d'une case par une instruction. Il faut considérer cette portion de mémoire visible comme une fenêtre qui se déplace sur la totalité de la mémoire (les écrans n'étant généralement pas suffisamment grands pour en afficher la totalité). Les numéros à gauche indiquent l'adresse correspondante et la partie droite en dévoile le contenu (valeur numérique en instruction).

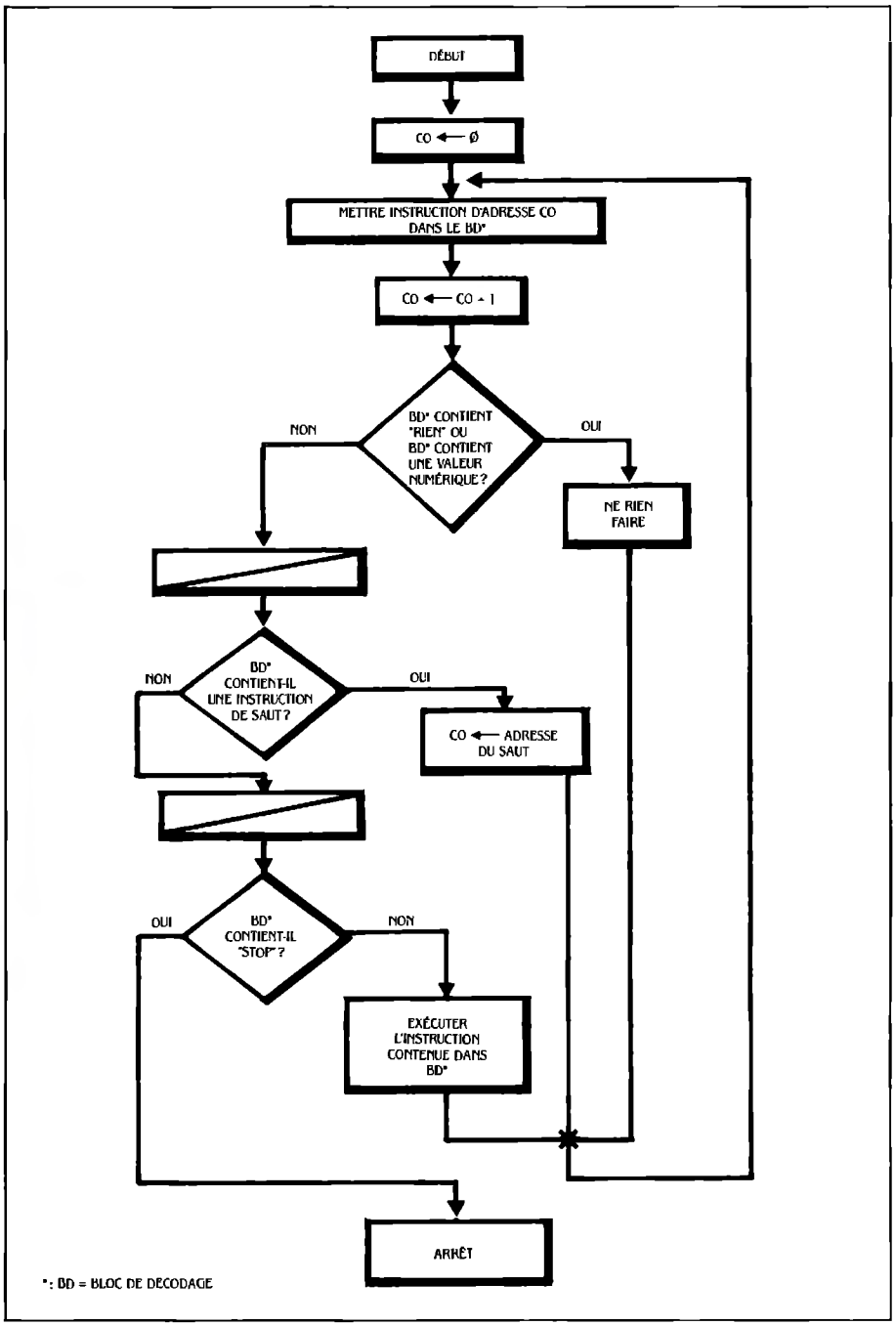
A ce propos, il est possible que vous vous posiez quelques petites questions depuis que vous avez chargé et exécuté le programme Ordidactic, la mémoire est en effet remplie (littéralement) de RIEN. Sachez donc dorénavant qu'une mémoire d'ordinateur a une sainte horreur du vide : une case *ne peut pas* ne rien contenir, en conséquence, on y trouve toujours quelque chose. En l'occurrence, dans le cas du S001, existe une instruction "nulle" :


RIEN. (Rassurez-vous : tous les langages d'assemblage en possèdent l'équivalent : le 6502 et le Z80 utilisent l'instruction "NOP" (No Operation : littéralement, pas d'opération : ne rien faire). Au moment de l'initialisation du S001, un mécanisme automatique remplit la mémoire de RIEN, ce qui explique l'apparence ésotérique de vos écrans.

Mais peut-être qu'actuellement un autre sujet vous angoisse, car nous avions promis d'en parler plus haut : il s'agit des valeurs apparemment aléatoires contenues dans les registres A, B et les indicateurs. Contrairement à la mémoire, lors de la mise sous tension, rien ne peut laisser en effet présager du contenu des registres. C'est pourquoi ils contiennent effectivement une valeur aléatoire comprise entre 0 et 99 (0 ou 1 pour les indicateurs). Seul le compteur ordinal (CO) s'initialise automatiquement à zéro au moment de l'exécution : il faut bien commencer quelque part... C'est pourquoi, lorsque vous exécuterez un programme sur le S001, pensez toujours à *initialiser* les registres avec les valeurs que vous souhaitez : ne partez surtout pas avec l'hypothèse que ceux-ci contiennent, par exemple, zéro...

## Fonctionnement général du S001

Nous avons étudié les différents éléments qui composent notre machine. Il est, dès lors, temps d'en comprendre le fonctionnement général (plusieurs éléments sont à prendre en compte) :




– le S001 commence l'exécution d'un programme à l'adresse 0 ;
– les instructions sont exécutées (comme en Basic) dans la séquence des adresses mémoire (de la plus petite vers la plus grande) ;
– le compteur ordinal contient toujours l'adresse de l'instruction n + 1 dès lors que le bloc de décodage contient l'instruction d'adresse n ;
– le compteur ordinal peut être modifié par une instruction dite de "SAUT" (ex Basic : GOTO, GOSUB, RETURN...) ;
– l'exécution se poursuit jusqu'à ce que le bloc de décodage contienne l'instruction STOP.

Cet organigramme s'applique en général à tous les ordinateurs. Il laisse apparaître une instruction que nous n'avons jusqu'ici qu'effleurée : STOP. Comme on s'en doute, son seul rôle est d'interrompre le déroulement du programme.

Si vous utilisez le programme Ordidactic, sélectionnez l'option 5 du menu (remise à zéro), puis, l'option 2. Chaque fois que vous appuierez sur une touche, une instruction RIEN s'exécutera (après une remise à zéro, la mémoire est pleine de RIEN, aussi curieux que cela puisse paraître !).

Vous pourrez constater que le compteur ordinal affiche gaillardement en permanence la valeur correspondant à la prochaine adresse à exécuter. Et lorsque, au bout de quelques heures, vous serez lassé d'exécuter des RIEN, une étoile frappée au clavier vous permettra de revenir au menu.

Nous venons de voir les éléments de base qui vont vous permettre de programmer en langage machine S001 dès le mois prochain. Dans cette attente insupportable, n'en oubliez pas pour autant votre Basic !

Siméon-Victor MICRO
et Emmanuel SARTORI