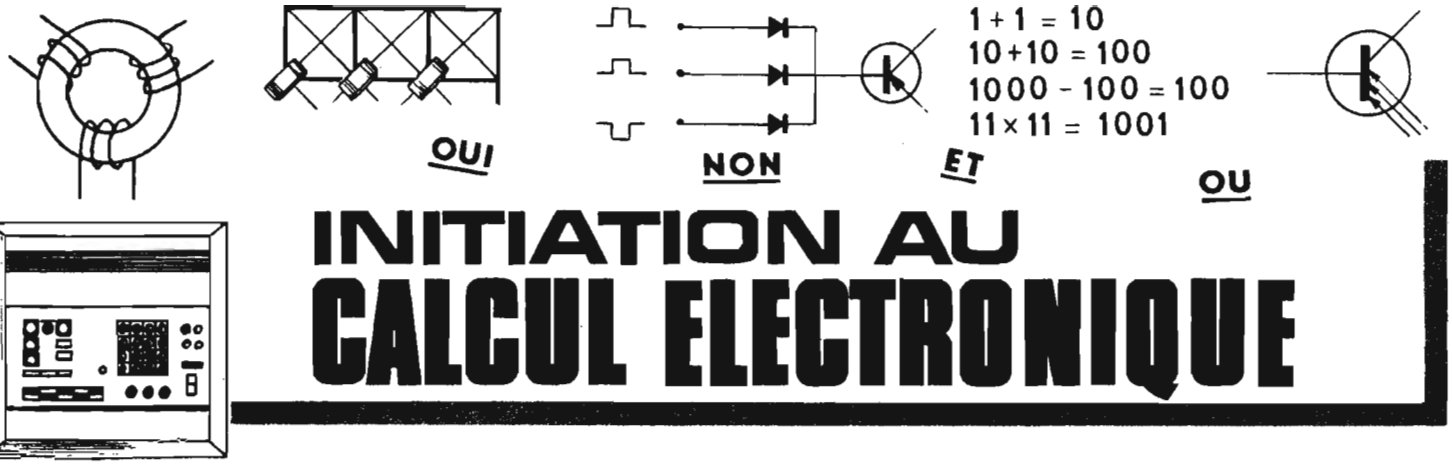


INITIATION AU CALCUL ELECTRONIQUE

LE FIRMWARE

DANS les ordinateurs, les informations à traiter, qui proviennent par exemple d'un lecteur de cartes perforées ou d'un télétype, sont introduites dans l'unité centrale, en passant par un dispositif d'entrée. Entre autres tâches, ce dernier procède à l'adressage des informations, en fixant leur destination, et les communique ensuite à un processeur central. Là sont réglés, en fonction des possibilités de la machine, le rythme d'entrée des informations à traiter, ainsi que la cadence de sortie des informations traitées. On y gère aussi la mémoire centrale, dans laquelle on entrepose les éléments nécessaires au calcul à effectuer, ainsi que certains résultats intermédiaires. Et c'est là, enfin, que l'on procède aux calculs proprement dits.

Toutes ces opérations s'accomplissent grâce à un certain matériel, le hardware, dont on se sert selon un certain nombre de règles, le software.

Hardware et software constituent donc des éléments bien distincts mais interpénétrables : on peut transférer une partie du software sur le hardware. On parle alors du « software câblé » ou encore du firmware.

L'ORDINATEUR DECENTRALISE

Ainsi, la matérialisation de certaines fonctions permet de les répartir différemment au sein de l'ordinateur, autorisant, en quelque sorte, un éclatement des responsabilités. On passe alors d'une machine « centralisée », dont tous les éléments gravitent autour d'une plaque centrale, à une machine « décentralisée » où chaque sous-élément dispose, par rapport à tous les autres, d'une certaine autonomie.

Bien sûr, toute autonomie a ses limites, et, dans une machine décentralisée, comme dans une machine centralisée, c'est toujours le processeur central qui supervise toutes les communications.

PREMIERE APPLICATION :

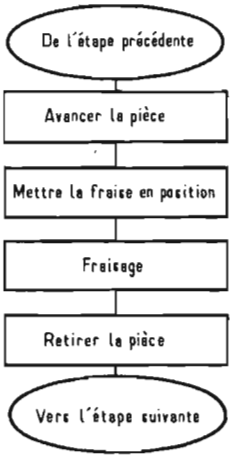


FIG. 1

PILOTAGE D'INSTALLATIONS INDUSTRIELLES

La figure 1 représente un diagramme d'organisation, ou ordinogramme, d'un programme de commande de machine-outil. Ici, une seule opération de commande correspond à chaque pas du programme. En général, on associe les mêmes opérations de commande à plusieurs pas différents : les opérations de commande se ramènent alors à un répertoire d'opérations parmi lesquelles le programme choisit celles qui correspondent au pas en cours d'exécution.

Par exemple, le répertoire d'opérations du programme représenté à la figure 1 comprend les instructions « avancer la pièce », « retirer la pièce »... Plus généralement, supposons que le répertoire d'opérations se compose des quatre instructions élémentaires A, B, C et D. Le programme suivant : ACBABCDA peut être câblé dans le « hardware », à l'aide d'une matrice de diodes (Fig. 2). Chaque diode assure la sélection des opérations de commande ; l'ensemble est relié à un dispositif qui alimente successivement les lignes de la matrice, par exemple un relais électromécanique pas à pas, ou encore un compteur à anneau.

SECOND EXEMPLE : LE PDP11

Dans le petit ordinateur PDP11 de la firme Digital Equipment, la décentralisation des fonctions est fort prononcée. Tout d'abord, et comme dans une machine centralisée, on y introduit les programmes qui définiront l'agencement des opérations à effectuer. Ces programmes, toujours comme dans un ordinateur classique, sont stockés dans la mémoire centrale.

Supposons que les informations à traiter soient déposées sur des cartes perforées. C'est donc le lecteur de cartes qui sera chargé de capter ces informations. Après l'avoir alimenté en cartes perforées, on donne l'ordre à la machine de commencer le traitement. Aussitôt, le « processeur central » donne l'instruction de commencer la lecture. Cet ordre étant donné, le processeur se désintéresse de la question et prépare les autres systèmes composant l'ordinateur. Le lecteur de cartes poursuit l'opération, jusqu'à achèvement des lectures, puis lance, de sa propre initiative, un signal d'interruption, qui met automatiquement le canal de transmission des données à sa disposition. Les données, stockées temporairement dans une mémoire-tampon, située à proximité du terminal, sont alors transférées dans la mémoire centrale de l'ordinateur.

C'est justement parce que le PDP11 est muni d'une logique câblée qu'il est possible à l'ordinateur d'effectuer plusieurs travaux simultanément. Le processeur central, véritable maître des lieux, donne des ordres aux divers sous-ensembles, ses esclaves dotés néanmoins d'un certain degré de liberté.

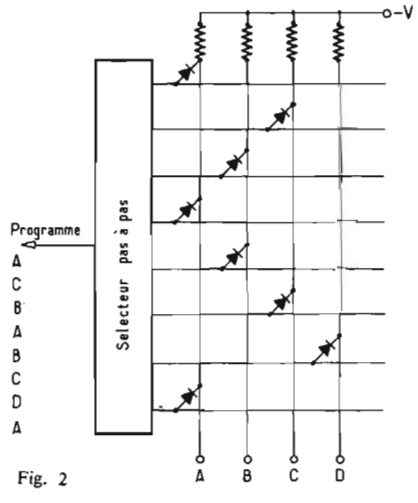


Fig. 2

LE ROLE PREPONDERANT DES MEMOIRES MORTES

Le degré de liberté donné aux sous-ensembles de l'ordinateur est bien entendu programmé par le constructeur. On parle de « micro-programmation ». La micro-programmation peut servir à la commande de processus internes à l'ordinateur, ou encore à certains calculs arithmétiques.

Ainsi, par exemple, dans les mini-ordinateurs de bureau, les constructeurs prévoient souvent un certain nombre de fonctions câblées : exponentielles, logarithmes, sinus, cosinus, tangente..., peuvent être calculés simplement en appuyant sur une touche. La machine contient dans sa mémoire tout le hardware nécessaire au calcul de telles fonctions mathématiques.

Le statisticien aura besoin de calculer des moyennes et des variances : le constructeur du mini-ordinateur pourra prévoir une plaquette qui s'insérera dans la machine, et qui contiendra, câblées, les fonctions demandées par le statisticien.

Et il en sera de même des fonctions demandées par une quelconque profession, qu'elle soit technique, scientifique, mathématique ou commerciale.

En somme, la mini-calculatrice électronique sera constituée d'un bloc de base, auquel l'utilisateur ajoutera les micro-programmes dont il a besoin.

La micro-programmation peut coûter très bon marché grâce à l'utilisation de mémoires semi-conductrices à lecture seulement. Ces mémoires (1) sont dites mortes, car, le plus souvent, il n'est guère possible d'y inscrire d'autres informations que celles y étant déjà imprimées.

Fig. 3. — *La représentation binaire des nombres décimaux.*

Nombres décimaux	Représentation binaire
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
10	00001010
11	00001011
12	00001100
13	00001101
14	00001110
15	00001111
16	00010000
17	00010001
18	00010010
19	00010011
20	00010100
21	00010101
22	00010110
23	00010111
24	00011000
25	00011001
26	00011010
27	00011011
28	00011100
29	00011101
30	00011100
31	00011111

MULTIPLICATIONS CABLEES

Considérons deux nombres binaires A_8 et B_8. Supposons que chacun de ces nombres binaires contienne 8 bits, c'est-à-dire que leur représentation binaire se compose de 8 chiffres égaux à 0 ou 1 (Fig. 3).

Un chiffre binaire quelconque peut s'écrire de la façon suivante :

$$A_8 = A_4 + (\Delta A)_4$$

ou $A_4 = XXXX0000$

et $(\Delta A)\ 4 = XXXX$

Ici chaque X représente une valeur binaire 0 ou 1.

Par exemple, si A_8 représente la valeur binaire du nombre binaire 31, on a :

$$A_8 = 00011111$$

On prendra alors :

$$A_4 = 00010000$$

et $(\Delta A)_4 = 1111$

Le nombre A_8 a finalement été divisé en deux nombres binaires de plus faible dimension contenant, le premier, les 4 bits de gauche de A_8, suivis de 4 zéros, et le second, les 4 bits de droite de A_8.

Le nombre B_8 peut subir le même traitement :

$$B_8 = B_4 + (\Delta B)_4$$

Pour multiplier les deux nombres A_8 et B_8, on procède comme en algèbre classique :

$$A_8 \cdot B_8$$

$$[A_4 + (\Delta A)_4] \cdot [\Delta B_4 + (\Delta B)_4]$$

soit $A_4 \cdot B_4 + A_4 \cdot (\Delta B)_4 + (\Delta A)_4 \cdot B_4 + (\Delta A)_4 \cdot (\Delta B)_4$

Le produit de 2 nombres de 8 bits se ramène à une somme de 4 produits de nombres de 4 bits. Ce produit peut être effectué à l'aide de mémoires à lecture seulement, contenant chacune 2 048 bits, auxquelles on adjoindra 5 additionneurs à 4 bits. Ces composants sont commercialisés par de nombreux fabricants : Texas Instruments, Electronic Arrays, Fairchuld, National Semiconductors... Le schéma de la figure 4 utilise des mémoires MM523 de National Semiconductor et des additionneurs SN7483 de Texas Instruments.

L'association des mémoires aux additionneurs se fait convenablement en reliant chaque sortie des mémoires à une polarisation de − 12 V, au travers d'une résistance de 7 500 Ω.

Cet exemple est simple, certes, mais il est significatif : à l'aide de mémoires mortes, il est possible d'effectuer des opérations mathématiques. De la même façon que précédemment on pourrait réaliser des dispositifs de division, ou encore d'exponentiation.

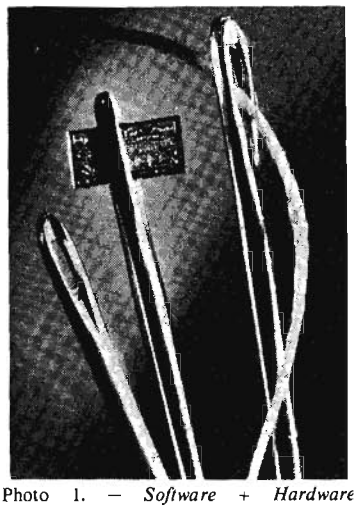


Photo 1. — *Software + Hardware = Firmware*

LE FIRMWARE, DANS LES ANNEES 70

Le firmware pourrait être considéré, non plus seulement comme un software câblé, mais aussi comme un ensemble d'informations de commande, réalisées jusqu'alors avec du hardware exclusivement. On regroupe alors, dans une mémoire à lecture seulement, l'ensemble des commandes, de sorte que la modification du jeu de commandes s'obtient simplement en changeant la mémoire sans modifier le reste du hardware — alors que, jusqu'à présent, il était nécessaire de recâbler le hardware.

Le changement de mémoire peut être physique : on débranche une plaquette électronique, et on en met une autre à la place ; cela peut être aussi une opération électrique. Ainsi, la mémoire morte RMM256 développée par ENERGY CONVERSION DEVICES, est basée sur l'utilisation de semiconducteurs amorphes, dits « ovoniques » : son contenu peut être modifié électriquement.

Les « ovoniques » sont des verres complexes, à base de tellure et de germanium, présentant la propriété de passer de l'état amorphe à un état beaucoup plus ordonné, lorsqu'il leur est fait un apport énergétique extérieur ; cette modification de structure est réversible et

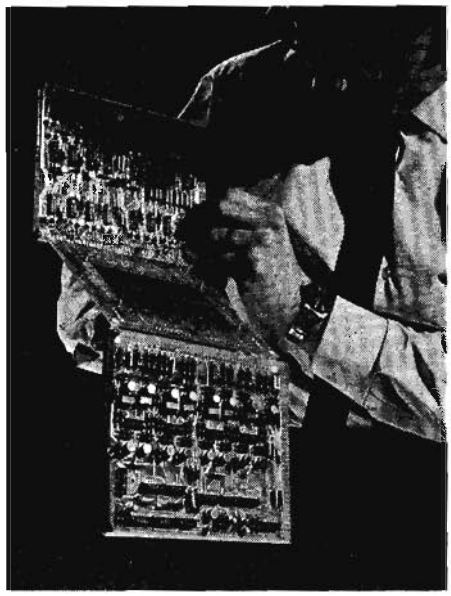


Photo 2. — *La Mémoire-tampon stocke temporairement les données lues par le terminal, puis les retransmet à la mémoire centrale, sur ordre du terminal même !*

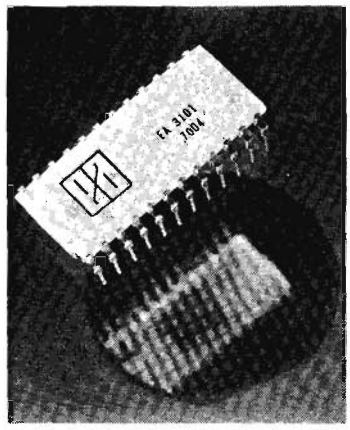


Photo 3. — *Cette mémoire à lecture seule est la clé fu firmware.*

s'accompagne d'une modification considérable de la résistance électrique, phénomène directement employé dans le RMM256. Le matériau peut ainsi basculer d'un état à un autre, et conserver indéfiniment son dernier état.

Initialement, le firmware était destiné à la commande et au contrôle du fonctionnement des gros ordinateurs. Cependant, son rôle s'est considérablement accru également dans les mini-ordinateurs, pour deux raisons essentielles : d'une part, on peut réaliser une mémoire de contrôle de volume beaucoup plus faible que celui de la mémoire principale ; ensuite d'une petite augmentation des dimensions de la mémoire de

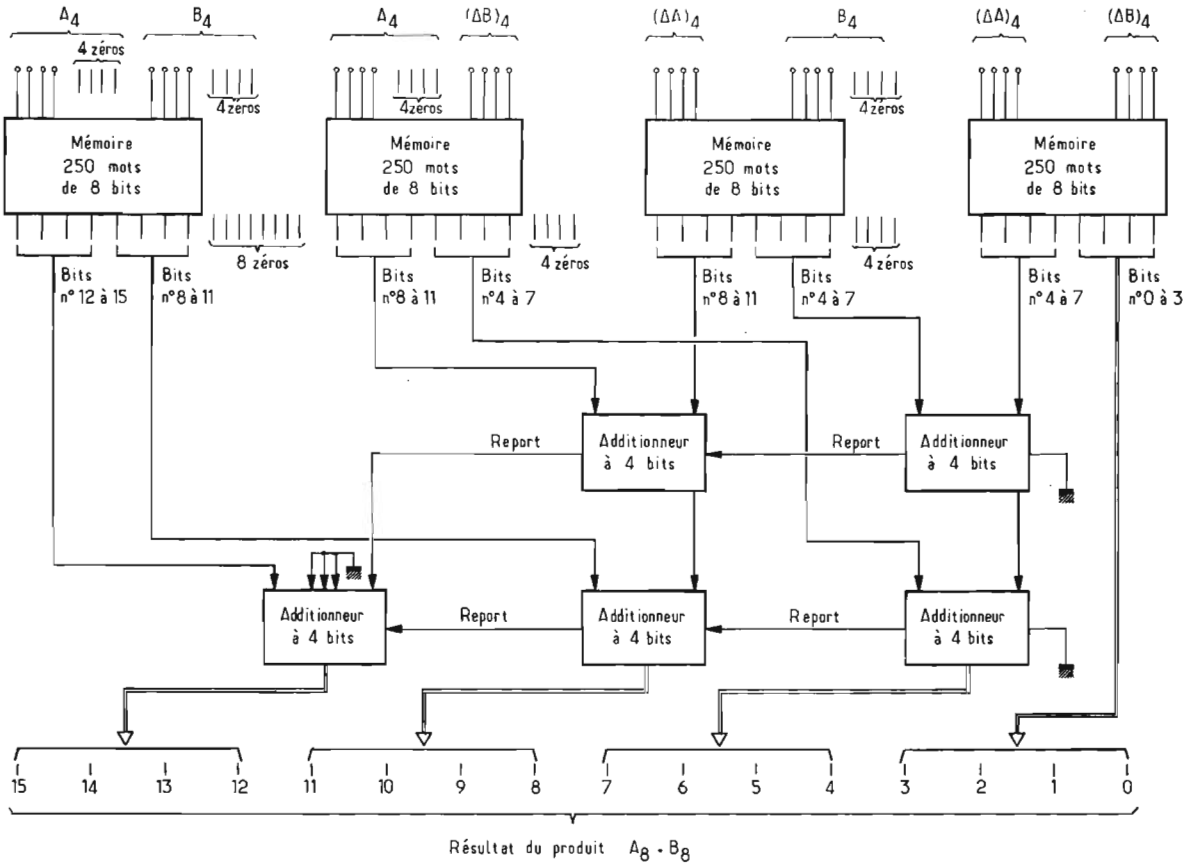
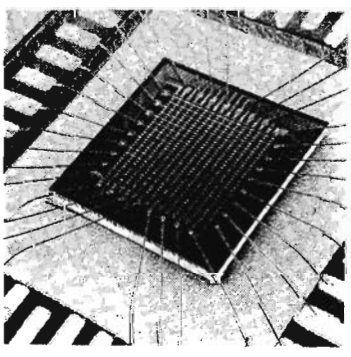


Fig. 4

contrôle, résulte une diminution appréciable des dimensions de la mémoire principale, et une augmentation considérable des performances de la machine.

Certains envisagent une croissance du firmware jusqu'au point où il ne serait plus possible de distinguer mémoires et processeurs. On pourrait ainsi construire un mini-ordinateur disposant d'un jeu d'instructions micro-programmées, que l'utilisateur pourrait lui-même altérer. Il suffirait d'insérer dans la machine, des mémoires mortes reprogrammables spécialement conçues pour fonctionner en opérateurs logiques. Ainsi le programmeur pourrait stocker dans ces mémoires des tables de vérité de fonctions logiques, même les plus complexes.

Photo 4. — *Voici la première mémoire morte reprogrammable.*



Ainsi donc, les mêmes mémoires mortes pourraient servir au contrôle du fonctionnement logique des machines électroniques, mais aussi aux calculs arithmétiques nécessaires lors du traitement des informations. Donnez à ces mémoires la possibilité d'être reprogrammées, comme dans le cas des « ovoniques », et il est probable que leur utilisation pourrait être généralisée à une très grande échelle, conduisant à une diminution considérable des coûts de fabrication !

Marc Ferretti.

Photo 5. — *Dans la Super Nova, des informations sont stockées dans les mémoires mortes. L'utilisateur, après avoir écrit son programme, peut commander au constructeur les modules de mémoire à lecture seulement réalisant ce programme. Il libère ainsi la mémoire centrale du mini-ordinateur.*