LABORATOIRE d'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M.

Faculté des Sciences Economiques
Université d'Aix-Marseille II

ISSN
0291-5413

INFORMATIQUE FONDAMENTALE & APPLICATIONS

Dépositaire

B.U. Sc. Eco.
Aix-Mars. II

BULLETIN N° 31

SOMMAIRE

Mars 1992

Adresse postale:
Faculté des Sciences Economiques
LITAM
14 rue Puvis de Chavannes 13001 Marseille

Tel. 91 13 96 29 - (20, 21)

EDITORIAL

E. Bianco

Informatique et devenir.

Il importe fortement que l'informatique ne reste, pour l'instant, que ce qu'elle est vraiment : un jeu, un amusement sophistiqué. Son emploi sérieux dans la société devrait être rigoureusement interdit avant plusieurs décénies au moins. J'ai hélas bien peur que le mal ne soit déja très avancé. Ceux qui prétendent penser à notre place n'ont pas compris, et ne comprendront jamais une vérité toute simple, et pourtant qui paraît universelle. Les gens ont horreur de participer à des décisions prises indépendamment, voire à l'encontre de leur volonté. Et si judicieuse que peuvent leur apparaître certaines décisions de nature technocratique, les ministres auront du mal à les faire appliquer. C'est bien le cas des longues suites de réformes de l'éducation nationale, qui, malgré de bonnes intentions parfois, n'ont abouti qu'à une pagaïe du plus bel effet, voire un saccage du peu d'efficacité que pouvait encore comporter l'enseignement. Si je parle d'efficacité, c'est au sens humain du terme, de l'efficacité dans l'élargissement de la connaissance de l'être humain baignant dans un milieu humain. Et non pas dans l'amélioration de l'adaptabilité de l'être humain à sa future situation de pièce de machine, dans une énorme machine à produire de la plus-value.

L'expérience ne sert pas. Les premières grandes préoccupations des bolchéviques concernaient la lutte contre l'anarchie, et partant, la mise au pas de la paysannerie. Résultat, un saccage définitif de l'économie de base du pays, je parle de l'économie qui fait vivre les estomacs, et non pas celle qui dore les golden boys.

Comment intervient l'informatique dans tout ce bazar, en fait d'une manière très simple. Sous des dehors d'une rigueur exemplaire, on la présente comme argument définitif de bourrage de crâne. C'est elle qui désigne les Présidents de Républiques, c'est elle qui définit les axes principaux de l'économie mondiale. Elle est l'argument péremptoire et sans réplique. C'est en fait, un puissant catalyseur de schizophrénie. Qu'il s'agisse des jeux simulés de la guerre, de parcours semés de monstres, ou, à un plus haut niveau d'argumentation à base de sondages minables et de statistiques faussées, ou mal interprétées, le rôle de ce qu'on appelle l'informatique est le même. C'est une caution "scientifique", dont le décalage de culture apparent entre les gens au pouvoir, et l'électeur moyen assure l'impunité dans le mensonge.

Devant un tel constat une question surgit d'elle-même. Que reste-t-il d'une informatique domaine de connaissance à part entière ?

C'est en s'interrogeant de la sorte, qu'on peut évaluer le degré d'incompétence et de légèreté, pour rester courtois, des cerveaux qui se préoccupent de nottre bonheur. Psychologiquement, l'impact des mots sur l'imagination n'est pas une simple vue de l'esprit, ce n'est pas Séguéla qui me contredira sur ce point. Or, que représente ce mot "informatique" dans l'imagination populaire. D'une part il est joliment construit, ensuite il laisse largement sous-entendre que l'information ça se calcule, et que l'électronique est là pour élaborer des vérités définitives contre lesquelles aucun raisonnement humain, par essence faillible, n'a de prise. Pour l'instant, et sans doute pour une assez grande masse de gens, il demeure un concept mystérieux, impressionnant par ce qu'il représente de machinerie écrasante, inquiétant par la masse d'hyper-spécialistes entièrement esclavagisés par les monstres électroniques. Une véritable mythologie est née. Tout le monde connaît plus ou moins un "informaticien" dans ses proches et tout le monde a pu percevoir les difficultés ressenties par ces nouveaux prométhée qui tentent de voler une parcelle de ce feu moderne, et se brûlent plus souvent qu'à leur tour, d'autant que le chômage sévit déja pas mal dans la profession.

Pour ne rien arranger, ce qu'on appelle "informatique" s'est en fait développée à deux niveaux très différents. Celle dont je viens de parler était l'informatique de surface, celle qu'on voit ou croit voir. L'autre était beaucoup plus souterraine, les gros requins ont tout de suite flairé la montagne de fric. Et il ne faut alors pas trop s'étonner si la chose s'est développée d'une manière apparemment bizarre, avec d'étranges bourgeonnements qui n'ont qu'un lointain rapport avec la "science".

Malgré "l'importance révolutionnaire" de la chose, le substrat sociologique ne semble guère touché. Aucune vaguelette, pas même un frisson ne trouble la sérénité cristalline de l'inertie administrative . Les trompettes de la renommée dûment accompagnées de grosses caisses pleines de vacuité intellectuelle, entonnent l'hymne informatique, mais il n'existe même pas trace officielle de cette discipline dans le secondaire. Ce sont en général des amateurs qui font office. Je ne suis pas contre l'amateurisme volontariste, car il vaut largement le professionalisme désabusé. Mais on en retire l'impression que les réfléchisseurs officiels sont un peu dépassés, mais peut-être ne sont-ils que mal conseillés. **Allègrement** ce sont toujours les mêmes experts qui inspirent depuis des décennies, les différentes factions qui se partagent le pouvoir. Malgré les échecs. L'expérience, je le répète, ne semble pas servir beaucoup, ce qui n'est pas le cas des petits copains.

Si le tableau n'apparaît pas spécialement réjouissant, que dire alors de positif, et même peut-être de constructif, car il paraît évident que tout ne saurait être à rejeter. Il faut à tout prix tenir

compte de l'ambiance, et ne pas négliger un fait important. En effet, si l'"informatique" s'est développée de cette manière, c'est en grande partie dû à la clairvoyance de passionnés du début, qui ont bien ressenti que pour faire avancer les choses il fallait donner au calcul automatique un bel empaquetage. On l'a qualifié de "big science". Autrement dit d'immense tas de fric potentiel. Mais pour que ça marche il est nécessaire que l'ensemble constitue une sorte de bloc énorme, sans failles visibles d'un peu loin, le tout noyé dans une sorte de brume luminescente d'où sortent de temps en temps de petits éclairs polychromes du plus bel effet. Une sorte d'arbre de Noël, mais à la dimension planétaire. Les faiblesses seront systématiquement mises au compte de l'humain, si fragile devant une telle puissance. Un progrès capital est venu étayer l'ensemble en même temps qu'il le faisait grimper haut dans les nuages. La micro-informatique permettait en plus d'investir tous les domaines de l'activité humaine, l'autre, la grande, pouvant difficilement percer hors des grands domaines spéculatifs. Maintenant, une machine à laver sans informatique ressemble vraiment à de la ferraille. Le silicium est partout, même dans les prothèses, et mon inquiétude grandit quand je me demande si dans peu de temps il restera le moindre embryon de vie derrière toutes ces prothèses. Même dans l'enseignement supérieur, l'informatique n'est et n'a vraiment jamais été considérée que comme une sorte de label, une étiquette qu'on colle sur un produit, ce dernier serait-il un étudiant. Alors on se demande s'il vaut mieux enseigner tel langage de programmation plutôt que tel autre, la question étant doctement tranchée après d'interminables colloques. Ou encore, pour ceux qui sont plus proches du "hard", quel micro-processeur va-t-on raconter de long en large. Hélas, le marché des langages s'étend, celui des processeurs aussi, que reste-t-il d'un tel enseignement au bout de quelques années ? Alors pour conjurer le sort on déclare péremptoirement: tel langage c'est la gestion, tel autre langage, c'est l'intelligence artificielle. En attendant impatiemment que tous ces langages deviennent un jour l'intelligence naturelle, ce qui résoudrait évidemment nombre de problèmes. Et je ne parle pas du choix d'un système informatique de gestion d'ordinateur, car on s'enfonce alors dans un abîme d'une tout autre nature.

Peut-être faudrait-il enfin faire quelques petites remarques qui découlent d'une pratique vieille de plus de trente ans. Quoiqu'il faille observer que cette durée est bien faible par rapport à l'évolution d'une pensée, la brièveté de ce laps de temps en est toutefois compensée par une accélération de sa diffusion. Il est toujours dangereux de se laisser distancer par une technique. Pour remettre un peu les choses à leur place il me paraît urgent de séparer d'abord des domaines qui ont évolué de telle manière qu'on ne peut plus les confondre. La programmation est aussi distincte de

l'informatique, que la technique de construction des ordinateurs, ou l'électronique qui est à la base de leur assemblage. Bien que tous ces domaines aient des points communs, comment redéfinir l'informatique ? C'est devenu relativement moins malaisé, car le recul me paraît suffisant pour débroussailler un peu. D'abord, pour être une véritable science, l'informatique doit se débarrasser de tous les faux arguments qui n'ont d'autre intérêt que de valoriser un produit. Là le chemin est clair. Si je désigne par objets de l'informatique, des notions telles que langages, programmes, ordinateurs et organes d'ordinateurs, etc, l'informatique se préoccupe des propriétés fondamentales et communes à tous ces objets, quelles qu'en soient leurs différentes occurences matérielles. Par ailleurs l'évolution de la technique a tendance à mettre en lumière un organe favorisé par une innovation ou un perfectionnement, en jetant aux oubliettes un autre organe frappé de péremption. C'est le cas des rubans magnétiques oubliés au profit des disques, disquettes et autres surfaces planes et tournantes, alors qu'ils avaient eux-mêmes détrôné les tambours magnétiques. La seule notion qui demeure universelle, c'est celle de file externe illimitée, qui demeure quel que soit l'avatar qui la matérialise à une époque donnée. La variété des supports capables d'emmagasiner de l'information s'est largement étendue, qu'il s'agisse de mémoires statiques, dynamiques, de mémoires définitives, ou provisoirement définitives munies de méthodes variées d'effacement, ou encore de mémoires à bulles qui ont failli s'attaquer aux disquettes. Mais une seule chose demeure, il existe en fait seulement deux sortes d'organisations de mémoires: celles à nombre fini-borné de cases, dont les noms sont calculés à l'aide d'un index, qui n'est autre qu'une case dont le contenu est le nom d'une autre case, celles à nombre illimité de cases dont l'accession se fait par défilement. Et c'est finalement cela seul qui est important du point de vue fondamental. Imaginons que les mémoires intégrées deviennent si volumineuses en nombre de cases et en commodité de connection, que des mémoires mécaniques du type disque ou disquette deviennent inutiles. Les instructions d'échanges ne seront plus alors de même nature, mais l'accès aux diverses cases se fera toujours par défilement, même si ce défilement est un calcul dans un super index qui permet de sauter d'un banc de mémoire à un autre.

Pour qui jette un regard sur les trente à quarante dernières années, et qui aborde l'informatique comme nous abordions nous-mêmes le latin, de manière très spéculative, on constate que diverses notions qui peuvent paraître issues de pure abstraction, sont en fait des notions qui ont surgi d'elles-mêmes de par l'usage de l'ordinateur. Aux tous premiers jours du calcul automatique, l'une d'entre elles, clairement mise en lumière par Von Neumann,

touche à la structure intérieure de l'ordinateur, la machine universelle s'y matérialise sous forme de processeur. L'algorithme devient programme, car il faut le coder pour le mettre dans la mémoire, à la portée d'un algorithme définitivement câblé, lui, qui va pouvoir le dérouler. Et c'est ça, la machine universelle, cet algorithme définitivement câblé, qui lit des codes d'instructions. Après, sont apparues des notions logicielles. La construction de l'algorithme a forcé à dégager la notion de procédure, algorithme élémentaire dont la forme est indépendante de la configuration sur laquelle il s'applique. Et puis l'ordinateur devenant un instrument de plus en plus important, coûteux, difficile à gérer, on a pensé à lui faire assurer sa propre gestion, la notion de système était née. Parce qu'on a réussi à construire quelques systèmes qui fonctionnent, mais à quel prix ? d'aucun pensent que tout est dit sur le sujet, il ne reste plus qu'à passer des années à décrire longuement des séries de tels monstres tous différents les uns des autres, où entrent pour beaucoup les particularités des ordinateurs supports. La distribution de cette ineffable connaissance aux néophytes, ressemble alors beaucoup à une transmission d'arcanes. N'oublions pas le secret industriel.

L'œuvre fondamentale, elle, consiste à concevoir la nature même de la notion de système, de telle sorte qu'on dispose d'un modèle théorique construit sur une machine ne comportant que les structures indispensables. Le résultat donne alors un ensemble de critères d'ordre scientifique qui permettent d'évaluer véritablement les systèmes existants, en dehors de toutes autres sortes de critères fantaisistes. Fantaisistes au sens de la connaissance, bien sur, pas au sens du gros blé que le petit "secret" est susceptible de rapporter.

Il va bien falloir, un jour, passer outre un obstacle de taille. Mais quand ? Notre comportement est actuellement régi par une loi universelle: la loi du profit maximum immédiat, quel qu'en soit le moyen. Fondée sur une argumentation imparable: si je ne le fais pas quelqu'un d'autre le fera, alors ... . Et une nécessité tout aussi imparable, si je ne décuple pas ma productivité, mon percepteur me traîne en faillite. Les choses qui risquent d'être un peu importantes pour l'avenir, et qui, en conséquence n'ont pas de "rentabilité" immédiate, passent ainsi au second plan des préoccupations.

Ma foi, il ne reste plus qu'à souhaiter que les turbulences sociales engendrées par la pagaille, le gâchis, l'irresponsabilité, l'incompétence au plus haut niveau, l'incohérence, le désarroi, bref le chaos qui se propage dans un milieu rendu perméable par cet état d'esprit, ne nous entraîne pas dans une Yougoslavisation généralisée.

LE PROCESSEUR DE LA PROCÉDURE FORMELLE.

Subject classification informatics: D33, D34.

Résumé.

Avec ce qu'est devenue l'intégration dans le silicium, la construction effective d'un objet de la dimension d'une machine universelle, disons un processeur, devient une sorte d'exercice de style. Et même si le produit obtenu n'a aucune chance d'avoir, un jour, un quelconque succès commercial, il est bon toutefois de montrer qu'il ne saurait être la production exclusive de quelques rares cerveaux supérieurs dûment stipendiés par les gigantesques trusts de l'électronique.

Indépendamment de tout impact d'un tel travail sur le devenir de la société dont il est un des fruits, il paraît important, dans une théorie qui se veut homogène, et capable de recouvrir tout le champ de l'informatique fondamentale, de montrer que les propriétés avancées, sont vraies quelles que soient les formes possibles adoptées. Un processeur est une sorte de programme écrit dans un langage de câblage, en fait, un jeu de langages. La vérification qu'il fonctionne bien est une sorte de théorème d'existence de la propriété annoncée.

LE PROCESSEUR DE LA PROCÉDURE FORMELLE.

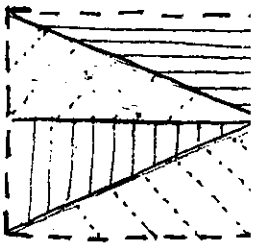
L'ENVIRONNEMENT DU PROCESSEUR.

Concevoir, puis construire un processeur, le langage étant déterminé, n'est pas, tant s'en faut une opération qui aboutit à un produit unique. C'est pourquoi l'exemple traité ici comporte un ensemble de choix déterminants qui dépendent en grande partie de raisons pratiques, les autres raisons sont d'ordre conceptuel. Comme technologie électronique, j'ai choisi le CMOS et la TTL-LS, pour leur facilité d'utilisation, et leur large diffusion. De plus, ces circuits ne sont pas très exigeants pour leur environnement. Pour l'organisation de la machine elle-même, comme partout, une réalisation est un équilibre entre plusieurs compromis. J'essayerai de montrer au fur et à mesure les raisons qui justifient plus ou moins certains sacrifices.

D'abord je décide d'une mémoire à cases de 16 bits. Premier compromis. Le code des instructions du langage de la procédure formelle, se découpe assez bien en tranches de cette dimension. Choisir l'octet aurait peut-être été plus économique en place mémoire, mais le traitement aurait exigé des algorithmes plus lourds, ce qui aurait surchargé le schéma du processeur. De toute manière, la place en mémoire n'est plus une véritable limitation, tant la technologie a rendu bon marché ce genre d'organe. D'un autre point de vue, comme on verra plus en détail, le mot de 16 bits laisse une bonne marge pour les adresses relatives, celles qu'on calcule à la main quand on définit une configuration. Pour ce qui est des adresses calculées automatiquement, j'ai prévu dans la structure du processeur des modules spéciaux pour réaliser ces calculs, de telle sorte que modifier l'espace de travail de l'ordinateur, revient simplement à changer les modules de calcul d'adresse. Pour ce prototype, la mémoire sera de 64 K mots (16 bits), changer les modules de calcul en rajoutant un buffer de 8 bits permet d'arriver tout de suite à 16 K^2 mots, soit de l'ordre de 16 millions de mots. Et, bien entendu, trois buffers en parallèle ne sont pas une limite pour calculer des adresses.

Le processeur fonctionne en statique, ce qui signifie qu'on peut le ralentir tant qu'on veut et même l'arrêter, cela ne détruit pas l'information qu'il traite. Je ne recherche pas la performance en vitesse pure, qui ne me semble pas présenter d'intérêt fondamental, quand on constate que les ordinateurs les plus puissants consomment le plus clair de leur temps à réaliser des tâches dont l'intérêt est plutôt douteux. Il me paraît plus important de faire porter l'effort sur la qualité de base du logiciel. Ici le choix du langage me paraît important dans la mesure où il contient des notions de l'informatique qu'on rencontre partout, et à des niveaux divers. Par exemple il me paraît décisif de disposer d'une véritable instruction d'insertion de procédure, car c'est en ce point que se prépare la gestion de la place en mémoire, c'est également là que se

transfèrent les informations indispensables. J'ai d'ailleurs prévu de compléter à brève échéance cette notion, par la notion d'insertion fractionnée, qui introduit la notion de vecteur information, et celle d'insertion différée. Ce sont ces notions de base qui permettent de gérer de manière systématique à la fois le temps et l'espace, par application de l'autojectivité.



Le langage: la Procédure Formelle.

Je rappelle brièvement la forme des instructions de la pro formelle. Je me limite pour l'instant au calcul en unité centrale.

Les formes:

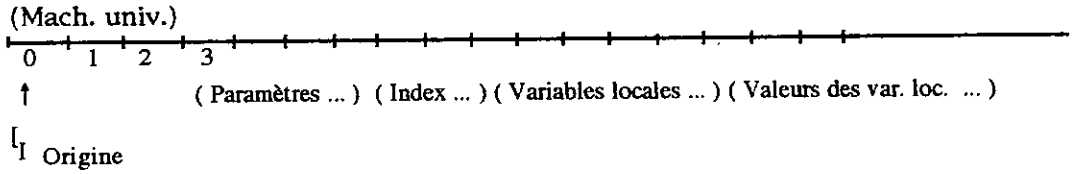
x := y
x := yA y
si y Ryvers Et
insérer(n1, n2, ... ,ni)
Procédure P
paramètres q1
index q2
variable locale s : v1, v2, ... ,vs

Les opérandes:

$$y = \begin{cases} x = \begin{cases} [_a \\ [_{a,b} \\ [_{\omega} \\ [_{\dagger} \end{cases} \\ \text{Constante} \end{cases}$$

A =opérateur arithmétique
R =opérateur de relation

Les indices: n1, n2, ... , ni, q1, q2, s, v1, v2, ... ,vs, sont des entiers. A cela il faut ajouter la structure de la configuration sur laquelle une procédure est susceptible de s'appliquer.

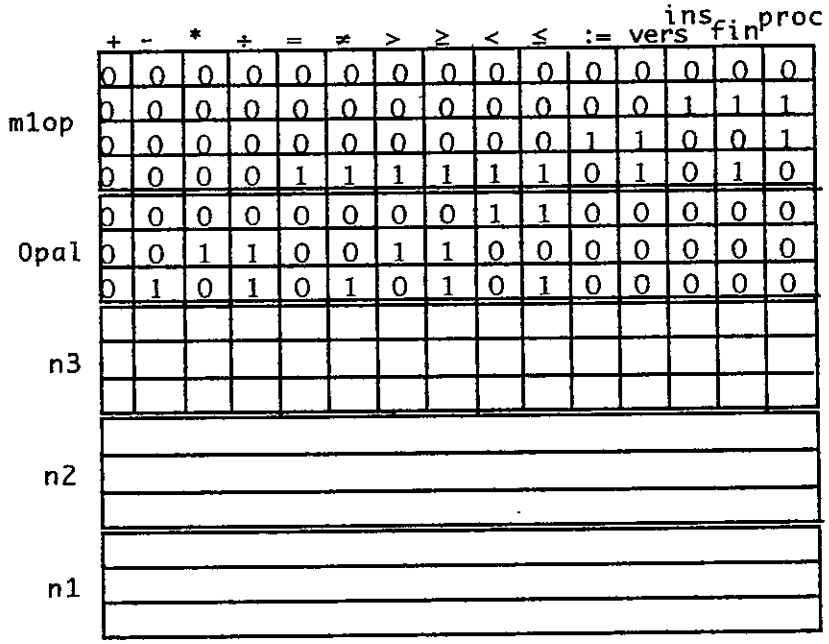


Sur une suite de cases numérotées relativement à l'origine, donc: 0, 1, 2, etc, et l'origine étant définie par le contenu de la case spéciale I, on porte dans l'ordre, des informations organisées. Les 3 premières cases, (numérotées de 0 à 2) sont réservées pour l'usage de la machine universelle. Puis, à partir de la case 3 on réserve les cases des paramètres, à la suite on porte les index et enfin les variables locales. Tout au bout seront logées les valeurs des variables locales. De la sorte, l'instruction de déclaration définit complètement la configuration de la procédure. Le texte d'une telle procédure facilite donc la gestion de la place occupée. Les extensions nécessaires lors des différentes insertions sont contrôlables à un autre niveau, qui est repérable pendant le déroulement car il surgit au déroulement de l'instruction d'insertion, et nulle part ailleurs. Il devient alors facile de faire

intervenir un traitement de nature système en ce point. C'est ce que j'essayerai de systématiser en introduisant l'insertion fractionnée.

Le code du langage.

Les instructions telles qu'elles sont présentées plus haut sont codées par tranches de 16 bits, à raison de la première tranche pour le code opérateur, qui contient le code opération en deux parties de 4 et 3 bits, puis 3 parts de 3 bits chacune, pour n1, n2, n3 les statuts de chaque opérande, défini par x et y. Pour les opérandes eux-mêmes, les cases I et ω , uniques, ne sont donc notées qu'au niveau du statut, l'indiçage simple et la constante prennent 1 case, et l'indiçage double en prend 2. La première tranche, réservée au code opérateur se code selon le tableau qui suit:



A titre d'exemple, une instruction de la forme:

[6,3 := [5 + 25

se code en sept cases:

(0000000 001 000 100), (0...110), (0...11), (0...101), (0...11001)

Je vais maintenant étudier plusieurs organes qui seront reliés ultérieurement. Je ne commencerai pas forcément par ceux que j'ai construits les premiers, mais plutôt par ceux qui montreront comment aborder la logique de l'ensemble. En premier lieu, je décris la carte mémoire, et ensuite, la face avant, c'est-à-dire le circuit de contrôle. Ces deux organes sont très liés car c'est par eux que se manifeste la première notion de système de gestion d'ordinateur. En effet lorsque l'ordinateur ne dispose pas d'un

logiciel déja complexe qui le relie à l'univers extérieur, il faut bien commencer par lui introduire les premières données soit: un programme, qui est le code d'un algorithme, ensuite les données du problème soumises à l'algorithme, et enfin la configuration qui permet au programme d'accéder aux données.

La face avant, dotée d'une circuiterie de contrôle, aura pour but de permettre:

1) d'interrompre le déroulement automatique du travail du processeur,

2) de choisir une adresse en mémoire pour en lire le contenu,

3) de modifier éventuellement ce contenu,

4) les données étant ainsi chargées et vérifiées, de relancer le calcul automatique.

Pour mettre en œuvre un tel programme, méta-programme devrais-je dire, il faut d'abord bien voir comment fonctionne une mémoire électronique.

La carte mémoire.

Je me limiterai à de la technologie TTL ou MOS, bien entendu. Il existe divers types d'organisation pour les circuits mémoires, mais la capacité de celles-ci s'accroit très fortement. En dix ans on est passé pour un circuit, de 256 cellules de 4 bits à 2 K octets, voire 32 K octets, je parle pour de la mémoire statique. Je laisse de côté la mémoire dynamique, car celle-ci exige un rafraîchissement régulier. Mon but est de construire un ordinateur qui fonctionne parfaitement de l'arrêt jusqu'à sa vitesse maximale qui doit être atteinte quand le cycle de base de l'horloge est de l'ordre de 300 à 400 nanoseconde. Il serait possible d'employer de la mémoire dynamique, bien moins coûteuse, pour des coefficients d'intégration plus importants. Mais cela complique la circuiterie inutilement pour moi. Non seulement j'utilise de la statique mais encore une bonne partie de la mémoire est composée de CMOS, une technologie dite Zéro power, qui conserve l'information même quand l'alimentation est coupée. On imagine aisément l'intérêt d'une telle propriété.

Paramètres de la mémoire.

Un circuit mémoire est une sorte de procédure dont chaque broche représente un paramètre binaire. Les paramètres sont rassemblés par groupes fonctionnels, chacun de ces groupes est attaqué par un BUS. On dispose des BUS suivants:

Le BUS adresse, qui est une donnée fournie au circuit, si celui-ci comporte 2 K octets, le BUS adresse est à 11 moments (11 conducteurs).

Le DATA BUS, à 8 moments puisqu'on traite l'octet. Il est unique et sert à la fois pour l'entrée et pour la sortie de l'information.

Le BUS de commande à 3 moments, dont on va étudier le détail.

Enfin le VCC et le GND, respectivement l'alimentation en courant régulé, généralement à +5 volts, et la masse. Ce circuit est à 24 broches.

Rythmes.

Bien que les temps de réponse soient faibles, variant de 100 à 500 nanoseconde, selon le type, ce temps n'est pas nul, et peut même dépendre de la nature de l'information transmise: un 0 ou un 1. En principe, le circuit est prêt à fonctionner dès que les potentiels transmis par le bus adresse sont stables. A partir de cet instant, le circuit est en attente de trois ordres distincts.

OE la validation de sortie; tant que cet ordre n'est pas arrivé, le circuit est coupé du data bus. La sortie est "flottante".

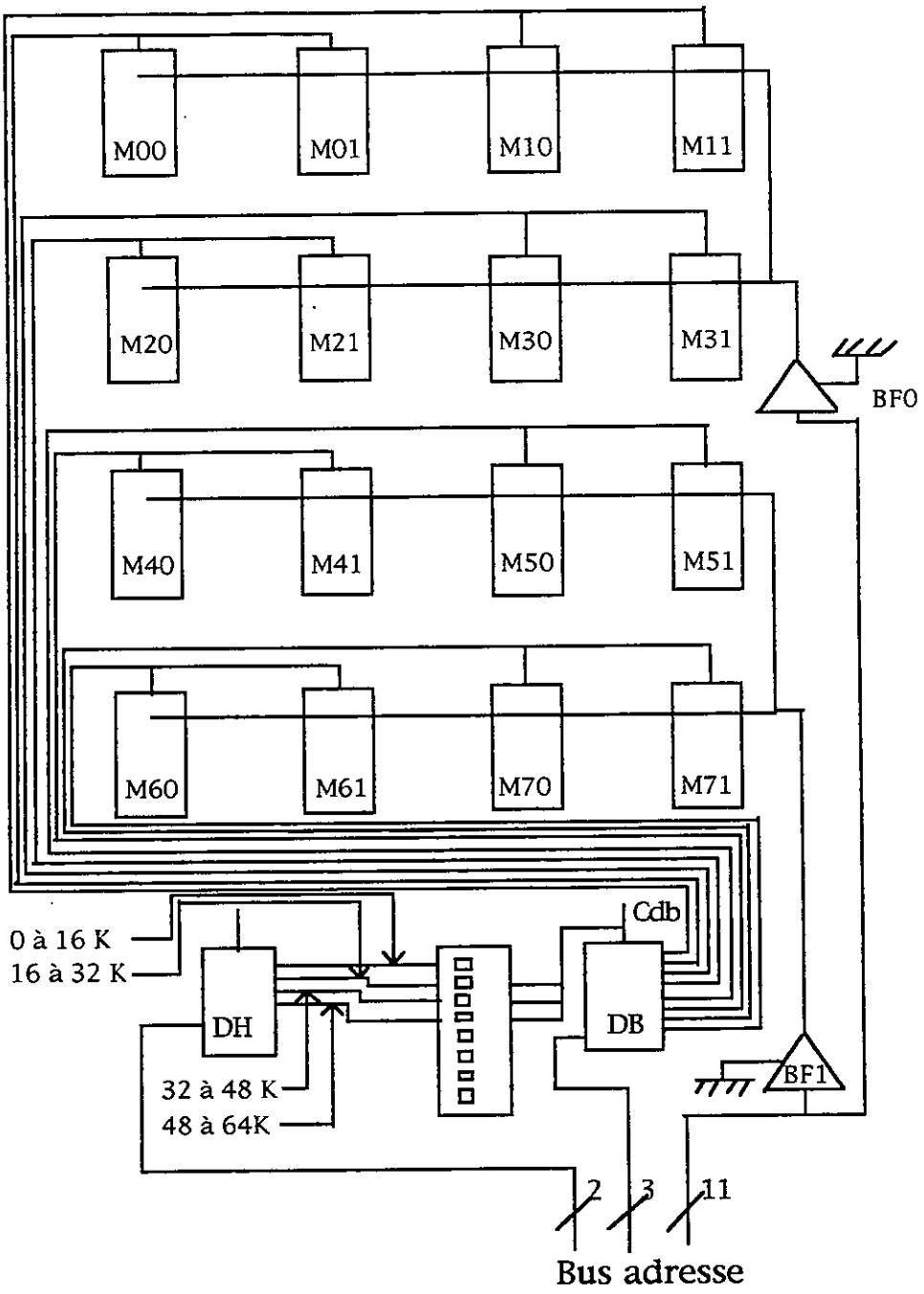
G et R/W deux signaux de commande de lecture et d'écriture, l'un des deux, par changement de valeur lance la lecture (l'émission d'information par le circuit), l'autre, par changement de valeur lance l'écriture (l'enregistrement dans le circuit de ce qui est stabilisé sur le data bus. L'un de ces deux signaux, G, informe le circuit du changement de sens de l'information portée par le data bus. Il est nécessaire de respecter un décalage de temps défini par le fabricant, dans les déclanchements relatifs.

Ceci est le schéma d'une carte à 16 K octets. Chacun des circuits Mij est un CMOS à pile qui conserve l'information même hors alimentation, et il comporte 2 K octets. Ces circuits sont montés par paires: Mi0 et Mi1 pour constituer des cellules à 16 bits. On comprend aisément qu'il soit dommage que des circuits à 32 K octets soient annoncés mais non encore commercialisés. On s'aperçoit de l'économie de câblage.

Afin de permettre l'attribution d'une adresse choisie pour la carte, (soit 0, soit 16 K, soit 32 K soit 48 K) le bus adresse est fractionné en trois parties, dont la plus basse, à 11 moments attaque directement les circuits à travers deux buffers 244, afin de conserver au signal adresse une puissance suffisante. Le latch PC du processeur ne pourrait commander les 64 circuits d'une mémoire à 64 K mots. Les 3 fils intermédiaires attaquent le décodeur DB qui, par l'une de ses sorties va sélectionner l'une des huit paires de circuits mémoire.

Le décodeur DB ne sera actif que si la commande issue de la batterie d'interrupteurs I est active. Et cette dernière sera active seulement si la sortie de DH attaqué par les deux poids forts de

l'adresse, coïncide avec l'interrupteur positionné. C'est le choix de l'un des 4 interrupteurs qui affecte une adresse à la carte.



Circuit de contrôle, tableau de commande.

A l'âge préhistorique de l'ordinateur, le tableau de commande est un organe essentiel, c'est par lui que va se manifester la première occurence de la notion de système. C'est par la main du

manipulateur que va s'accomplir le déroulement de l'algorithme de ce système. A l'origine, c'est l'utilisateur qui a l'initiative. Maintenant, avec les systèmes clefs-en-main c'est l'ordinateur qui a l'initiative. Ce sont des système compliqués, préenregistrés qui proposent un choix de tâches, qui se doit d'être suffisamment riche pour que l'utilisateur ne se sente pas trop frustré.

L'étude du circuit de contrôle montre bien ce qu'est, fondamentalement, un système. D'abord ce circuit met l'ordinateur dans deux états d'initiatives opposées. Soit l'automate est en "manuel", l'initiative est du côté du manipulateur. Cette phase correspond au chargement initial de l'information de base: le code de l'algorithme, les données du problème, et la configuration qui assure le lien entre code algorithme et jeu de données. Elle correspond également à l'exploration de la mémoire pour extraire des résultats de calcul.

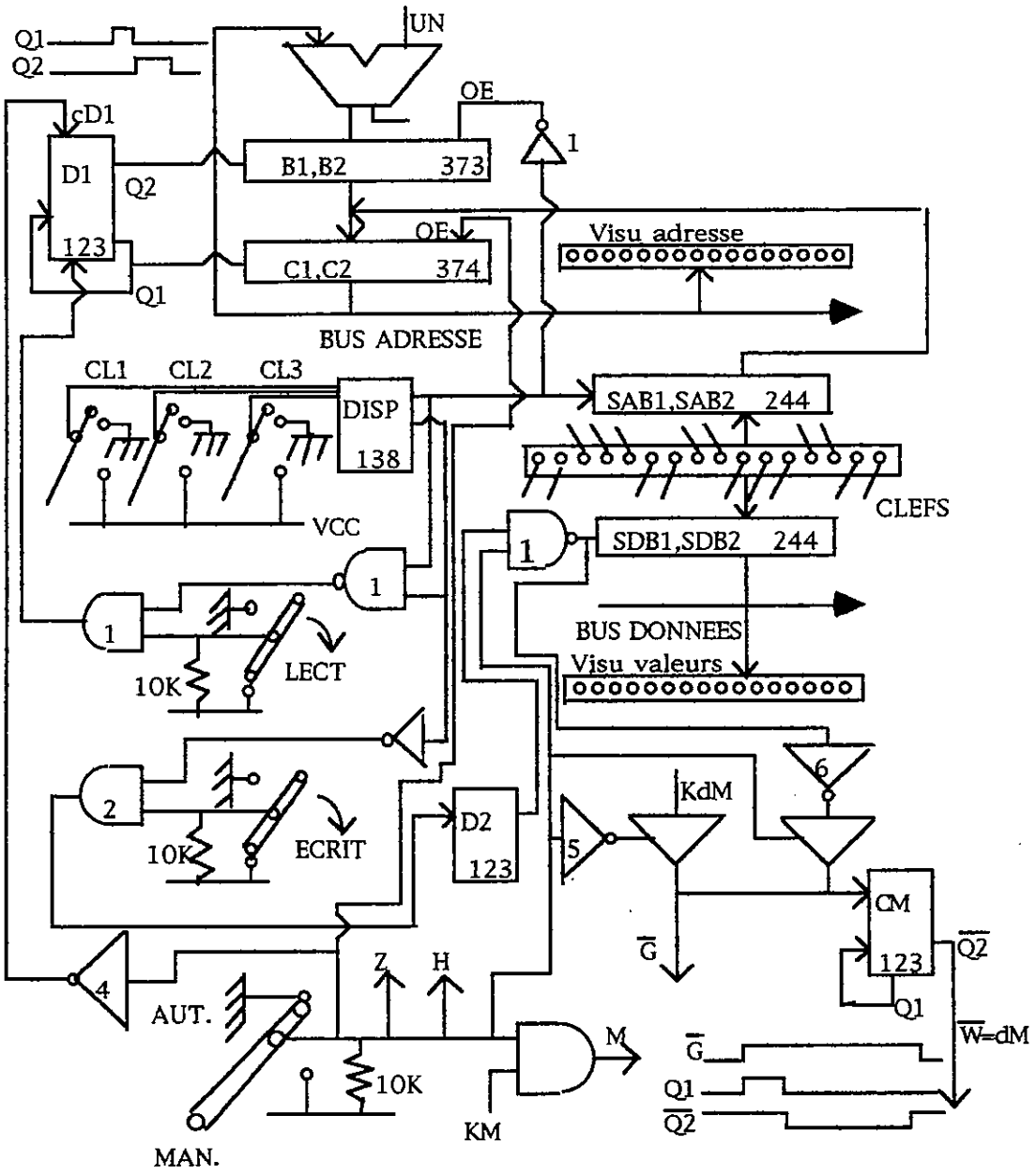
Soit l'automate est en "automatique", auquel cas c'est le processeur qui a l'initiative, et qui va dérouler du code programme tant qu'il rencontrera du code programme. Dans une période primitive dite l'âge de la pierre à calcul, le travail de l'automate est fini-borné: on lance une tâche, on arrête pour extraire les résultats et charger une nouvelle tâche, et on recommence. Dans une période plus moderne, le processeur dans son travail, se trouve alternativement en deux états, il déroule une tâche illimitée, c'est le système, et de temps en temps il déroule, sur initiative du système, une tâche d'utilisateur et revient en état système dès qu'elle est achevée. A la mise sous tension de l'appareil le programme système se charge automatiquement, de telle sorte que l'utilisateur moyen ne peut plus prendre le contrôle du calcul. Il est vrai que cela lui ote en général bien des soucis, mais aussi quelle joie sardonique pour les fabricants de virus.

Ce schéma de principe montre surtout un circuit économique en clefs. Le jeu de clefs CL1, CL2, CL3 servent à sélectionner l'un des huit états possibles. Les deux premiers sont les plus utiles:

Etat 0, noté avec les trois clefs à Zéro, représente l'état du chargement d'une adresse. Les latch C1, C2 maintiennent cette adresse sur le bus adresse, et elle est visualisée sur les voyants lumineux "adresses".

Etat 1, noté avec la clef CL1 à un, l'adresse ayant été fixée à l'état zéro, le bus données visualise sur les voyants lumineux "valeurs", la donnée contenue en mémoire dans la cellule sélectée par l'adresse. Appuyer alors sur la clef "LECT" fait passer automatiquement à l'adresse suivante, qui est visualisée sur les voyants "adresses", et c'est le contenu de la case suivante qui apparaît sur les voyants "valeurs". Quand on désire modifier ce contenu, il suffit de positionner les 16 clefs sur la nouvelle valeur,

et appuyer sur la clef "ECRIT" enregistre le contenu représenté par ces 16 clefs dans la case dont l'adresse est visualisée en "adresse".

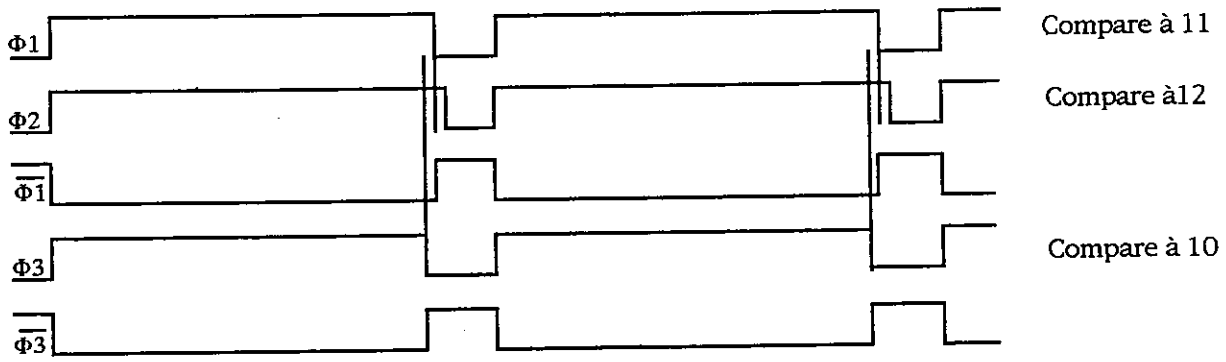


Les 6 autres états pourront servir par exemple à pratiquer diverses manipulations sur certains des registres du processeur ou de la machine de Dupuy couplée.

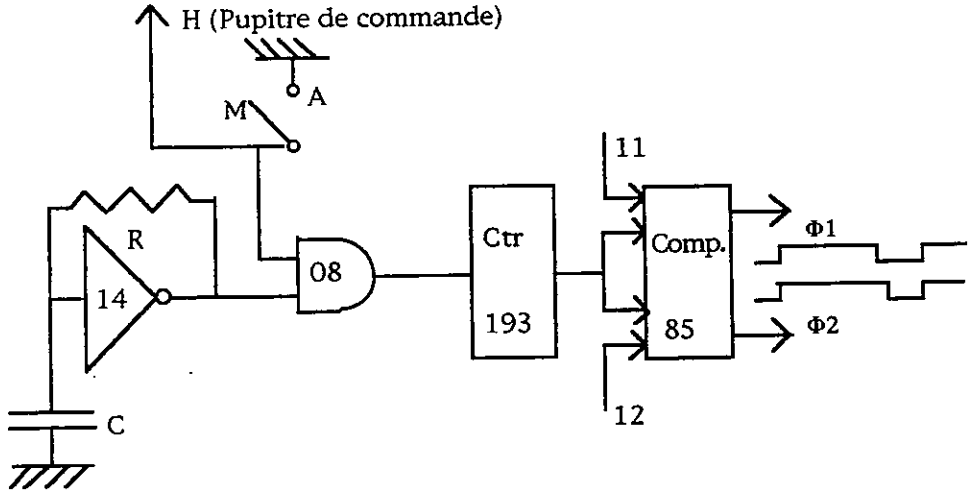
Bien entendu ce circuit n'est actif que si la clef "AUT,MAN" est en position "manuel" , sinon c'est le processeur qui est maître de la situation.

L'horloge.

Pour commander les blocages et les ouvertures il est nécessaire de disposer d'un jeu de signaux d'inégales longueurs. En effet, un signal qui contrôle une émission doit être légèrement plus long qu'un signal qui contrôle une mémorisation. Une fermeture simultanée pourrait à la rigueur marcher à cause du jeu sur les temps de retard à la réponse. Il s'agit là d'un équilibre un peu acrobatique. Il vaut mieux prévoir un décalage suffisant. L'émetteur d'horloge est constitué d'un oscillateur à fréquence réglable, obtenue par un couple résistance-capacité qui pilote un circuit 14. Les signaux carrés sont comptés par un 193 et le résultat comparé par < avec une valeur constante. Le comparateur, un 85 émet en conséquence un signal qui change de niveau quand la valeur comptée atteint la valeur donnée, puis rebascule au passage à zéro du compteur qui travaille modulo-16. J'ai besoin de 3 signaux Φ1, Φ2, et Φ3 ainsi que de leurs inverses.



Le schéma de principe montre comment obtenir les décalages sur la fin des signaux:



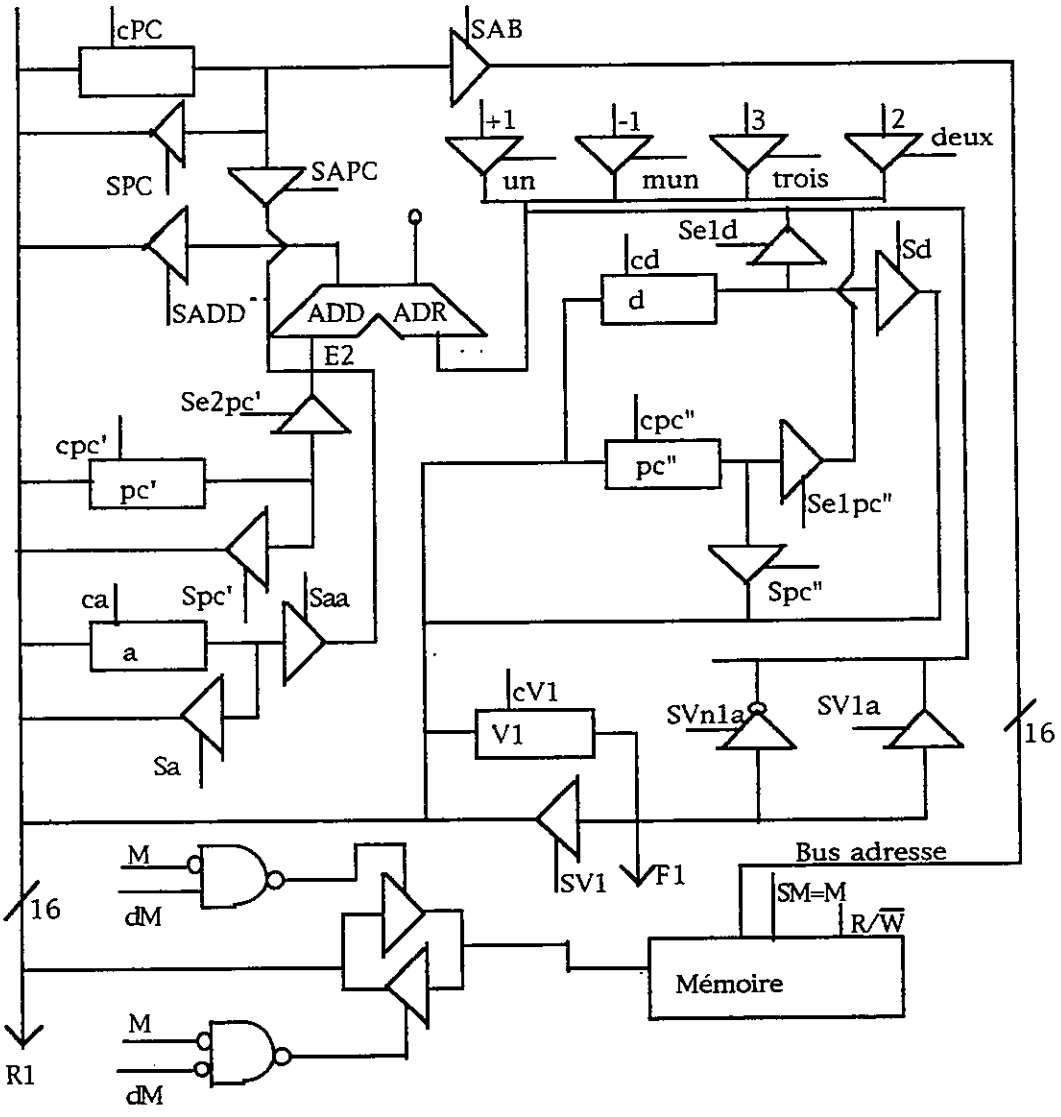
Le circuit comporte un autre 85 qui, attaqué de la même manière par le 193, fournit le signal Φ3. Une série de couples R/C permet de choisir diverses valeurs de la fréquence de base. Il faut observer que la fréquence de travail du processeur est égale à la fréquence de base divisée par 16, à cause du principe choisi.

LE PROCESSEUR.

LES CIRCUITS DE CALCUL.

J'ai séparé le calcul des adresses du calcul des valeurs. Dans cette première expérience, pour simplifier la réalisation, je conçois une machine carrée, c'est-à-dire dont la case peut contenir soit une adresse soit une valeur. Une version plus étendue permettra de calculer une adresse qui porterait sur plusieurs cases. Il ne suffit alors que de compléter la circuiterie du calcul d'adresse.

Ce processeur est du type statique, ce qui lui permet de fonctionner à toutes les vitesses, de l'arrêt à la vitesse maximum.



L'additionneur ADD ADR, qui comporte ici quatre circuits 74 HC 83, calcule les adresses par les sommes à 16 moments et 16 moments dans 16 moments. Le débordement n'est pas utilisé ici, pour la simple raison qu'un calcul d'adresse ne doit pas explorer la mémoire circulairement.

La cellule PC constituée de deux circuits 74 HC 373, contient en permanence l'adresse qui permet d'atteindre une case en mémoire. Son contenu peut avoir deux sens alternativement, soit c'est un accès à un code du programme, ce que d'aucuns appellent le 'fetch', soit c'est un accès à une donnée, dont l'adresse a été calculée au moyen du code du programme.

Les quatre cellules **p'c, d, a, p"c,** également des latches 373, servent au stockage et au calcul des adresses. En particulier, **p'c** contient l'accès au code programme, et **d** l'accès à la configuration.

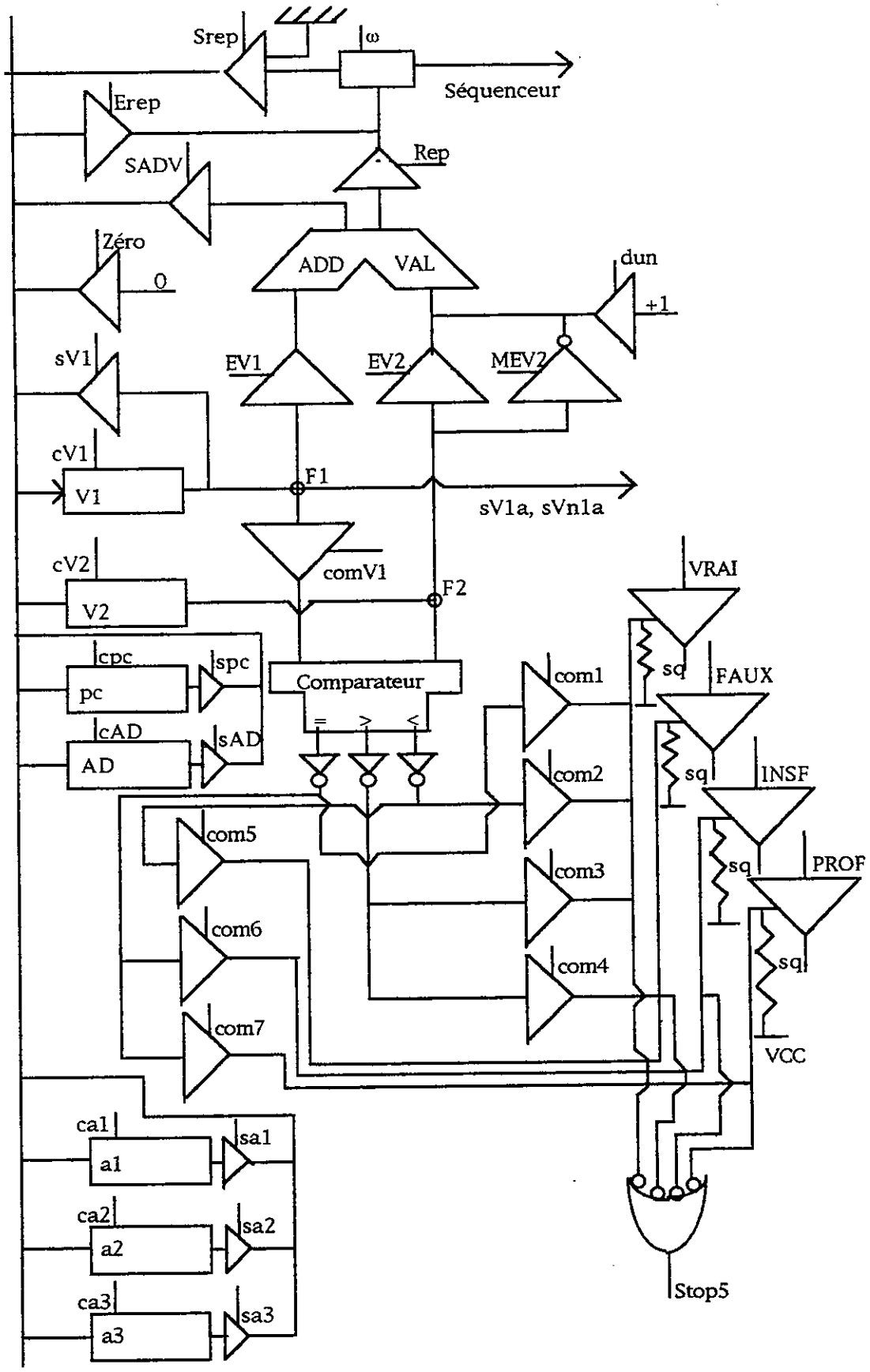
La deuxième partie du circuit, qui possède le micro-bus en commun avec la première, ainsi que le point F1, réalise les calculs des valeurs. Les cellules V1 et V2 contiennent les valeurs en jeu, soit pour l'arithmétique, soit pour la comparaison.

On observe sur ce schéma que le comparateur commande l'émission de l'une des quatre adresses: VRAI, FAUX, INSF, PROF. Ce sont des adresses dans le programme pilote du séquenceur. Les sorties 'sq' de ces buffers attaquent le circuit du séquenceur en "adresse suivante". Liste de ces adresses, exprimées en hexadécimal:

VRAI: AD
FAUX: A0
INSF: 14A
PROF: 120

Si aucune des commandes: com1, com2, com3, com4, com5, com6, com7, n'est à zéro, cela signifie que les sorties des buffers correspondants sont flottantes. Il en est donc de même des commandes des buffers qui émettent: VRAI, FAUX, INSF, PROF. Or, les sorties de ces derniers sont toutes reliées au séquenceur, d'où, pour éviter un court-circuit la nécessité de maintenir leurs commandes au potentiel VCC (+5 volts) par des "pull up" de 10 K Ω. Il est également nécessaire de bloquer le buffer: BUFSQ qui contrôle l'adresse-suivante. S'il apparaît ainsi un zéro sur l'une des commandes des buffers VRAI, FAUX, INSF, ou PROF, et il ne peut apparaître au maximum qu'un seul zéro à la fois, il faut qu'un "un", vienne bloquer le buffer BUFSQ. C'est le signal "stop5" qui doit jouer ce rôle, qui est précisé dans le schéma du circuit de commande de BUFSQ.

Le comparateur est une batterie de quatre 74 HC 85 qui possède trois signaux de sortie dont la valeur est fixée selon que la comparaison des deux données à seize bits chacune donne "égal", "plus grand que", ou "plus petit que". Ainsi, quand les deux valeurs sont égales, le bit "égal" est à 1 les deux autres à 0. Si la première donnée est constatée plus grande que l'autre, c'est le bit "plus grand que" qui est à 1 et les deux autres à 0. Même chose pour le troisième cas.

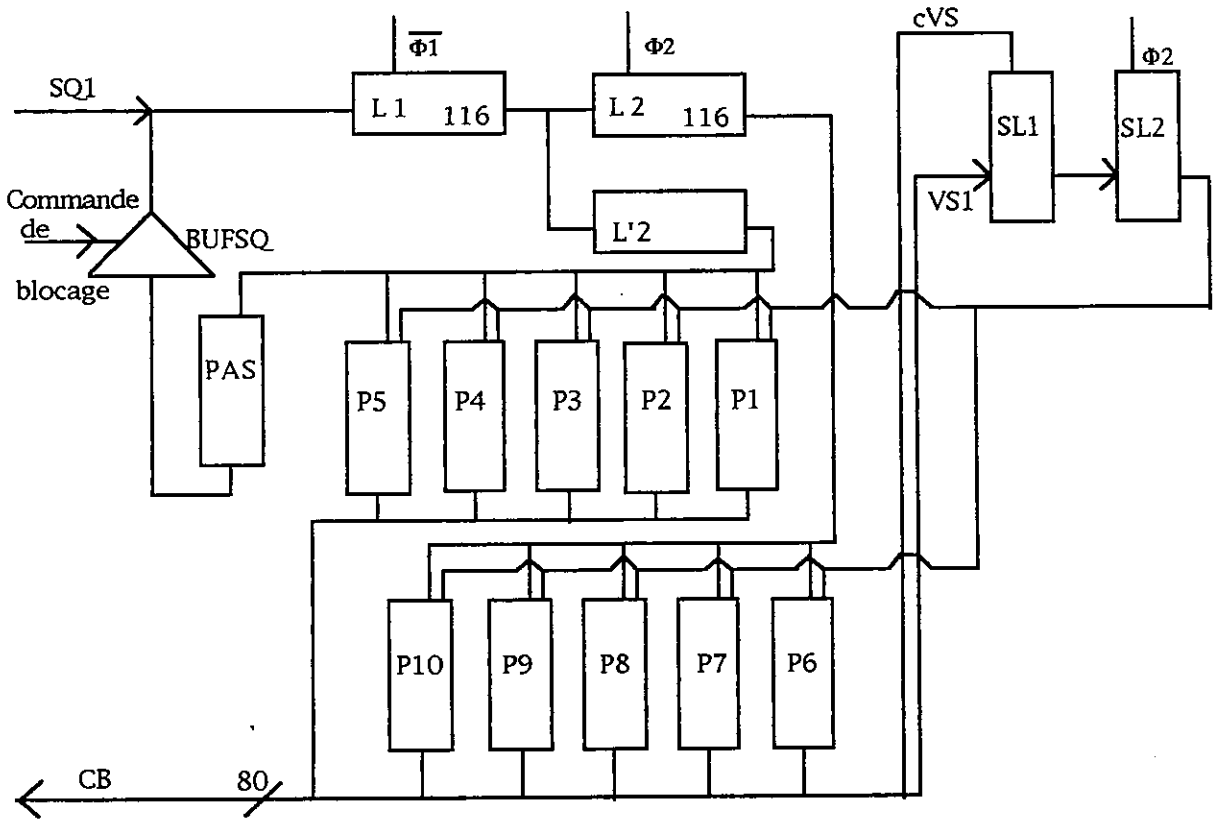


Les deux schémas ci-dessus se raccordent par le micro-bus, ainsi que par le point F1.

LE SEQUENCEUR.

Le circuit précédent, présenté en deux parties est l'organe de calcul de chacune des opérations élémentaires du processeur. L'enchaînement de ces diverses opérations est commandé par le séquenceur. C'est dans le séquenceur que se trouve enregistré définitivement le programme dont le déroulement organise le fonctionnement du processeur.

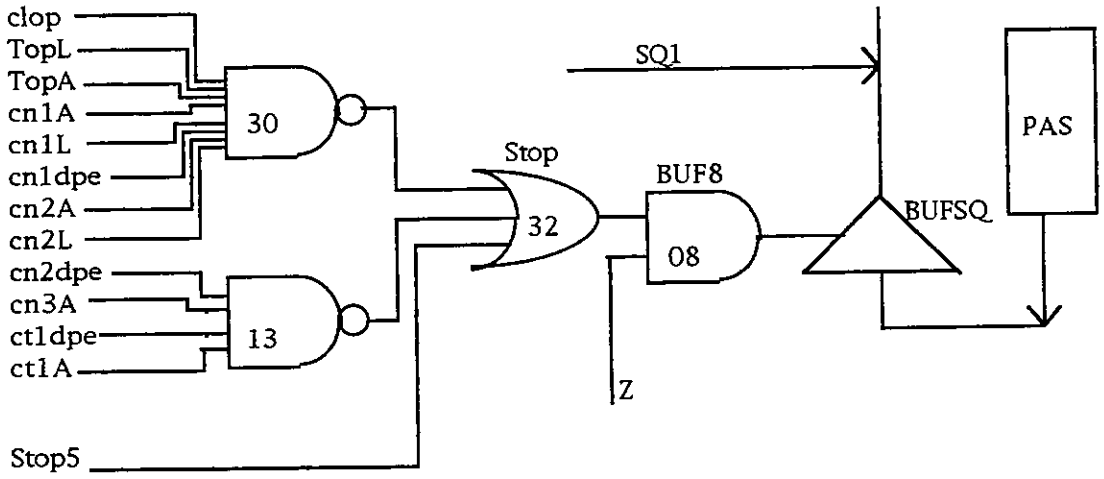
Chaque instruction du langage de la Machine Formelle, codée en mémoire, est interprétée par un chapitre particulier de ce programme. On distingue les instructions qui ne font que du calcul de données, et celles qui font du calcul sur l'algorithme lui-même, les instructions d'aller-à systématiques et conditionnels. Le traitement de cette deuxième catégories d'instructions réagit sur le fonctionnement du séquenceur, car le choix du code de l'instruction suivante dépend de la valeur d'une variable calculée. La batterie des PROM qui émet les signaux de contrôle du circuit de calcul, sont montées en parallèle avec une PROM désignée par PAS et qui contient pour chaque instruction interne le numéro de ligne de l'instruction interne suivante.



Dans le cas du traitement des instructions conditionnelles de la Procédure Formelle, le numéro de l'instruction interne suivante

n'est plus fourni par PAS mais provient d'un calcul. C'est le buffer BUFSQ qui coupe le circuit pour permettre l'introduction de la nouvelle adresse suivante.

Les PROM de P1 à P10 émettent chacune huit signaux, le CB, le bus de commandes est constitué de 80 moments. En particulier les deux commandes VS1 et cVS qui servent à l'étalement du programme interne sur plus de 256 lignes. En effet le circuit séquenceur principal, composé des deux latches L1 et L2, est un bus à huit moments, donc ne permet que d'atteindre 256 lignes dans la batterie de PROM, un moment de plus est fourni par SL1 et SL2 pour atteindre à 512 lignes ce qui est suffisant pour l'instant, mais pourrait être largement étendu. La commande de blocage est issue d'un circuit dont le schéma est le suivant:



Les commandes qui apparaissent dans la colonne de gauche, et qui attaquent les circuits 30 et 13, des NAND, sont issues, chacune, d'un calcul de condition. Le code du langage Procédure Formelle, explicité au début, est arrangé pour permettre de distinguer aisément au sens du câblage, les différents codes d'opérations, et les différentes valeurs des statuts n1, n2 et n3 des opérandes.

On analyse dans l'ordre:
Les 4 bits de poids fort, et la commande "clop" selon la valeur de code opérateur rencontrée dérive vers les étiquettes du programme interne:

EA: Arithmétique,
ER: Logique,
DPE: Affectation,
VERS: Commutation,
INS: Insertion,
FIN: Fin de procédure,
PRO: Déclaration de procédure.

Les cas de EA et ER exigent une deuxième exploration qui porte alors sur les 3 bits suivants du code, pour déterminer le type d'opérateur. Dans le cas du EA, la commande "TopA", fait dériver vers les 4 possibilités:

PL: Addition,
M: Soustraction,
MUL: Multiplication,
DIV: Division.

Pour le cas du ER, c'est la commande "TopL" qui fait dériver vers les étiquettes:

LE1: Egal,
LE2: Différent,
LE3: Plus petit que,
LE4: Plus petit ou égal,
LE5: Plus grand que,
LE6: Plus grand ou égal.

Le contenu de n1 est alors exploré, ce sont les 3 bits suivants vers la droite, dans le code. Et selon qu'on a rencontré un opérateur arithmétique, logique ou d'affectation, c'est-à-dire qu'on est dans les sections étiquettées EA, ER ou DPE, ce sont les commandes "cn1A", ou "cn1L", ou "cn1dpe" qui vont par l'intermédiaire d'un décodeur, aiguiller vers les étiquettes correspondantes. Premier cas: opérateur arithmétique donc commande""cn1A":

AU0: Opérande $[_a$
AU1: Opérande $[_{a,b}$
AU2: Opérande $[_I$
AU3: Opérande $[_\omega$

Deuxième cas: opérateur logique, donc commande "cn1L":

LU0: Opérande $[_a$
LU1: Opérande $[_{a,b}$
LU2: Opérande $[_I$
LU3: Opérande $[_\omega$
LU4: Constante

Dans le cas d'une affectation, le traitement se fait en DPE, et là un aiguillage contrôlé par la commande "cn1dpe", renvoie aux 4 cas possibles:

DPE0: Opérande $[_a$
DPE1: Opérande $[_{a,b}$
DPE2: Opérande $[_I$
DPE3: Opérande $[_\omega$

Le calcul est de même nature pour l'exploration de n2. En cas d'opérateur arithmétique, c'est la commande "cn2A" qui aiguille vers les étiquettes:
AD0, AD1,AD2, AD3, et AD4, où vont se traiter respectivement les 5 versions possibles d'opérandes: $\lceil_a$, $\lceil_{a,b}$, $\lceil_I$, $\lceil_\omega$ ou la constante.

En cas d'opérateur logique, c'est la commande "cn2L" qui aiguille vers les étiquettes:
LD0, LD1, LD2, LD3, LD4 où vont se traiter, de la même manière les mêmes versions d'opérandes: $\lceil_a$, $\lceil_{a,b}$, $\lceil_I$, $\lceil_\omega$ ou la constante.

Pareil pour l'opérateur d'affectation, pour lequel la commande "cn2dpe", aiguille vers les étiquettes:
DPES0, DPES1, DPES2, DPES3, et DPES4 où vont se traiter les versions d'opérandes: $\lceil_a$, $\lceil_{a,b}$, $\lceil_I$, $\lceil_\omega$ ou la constante.

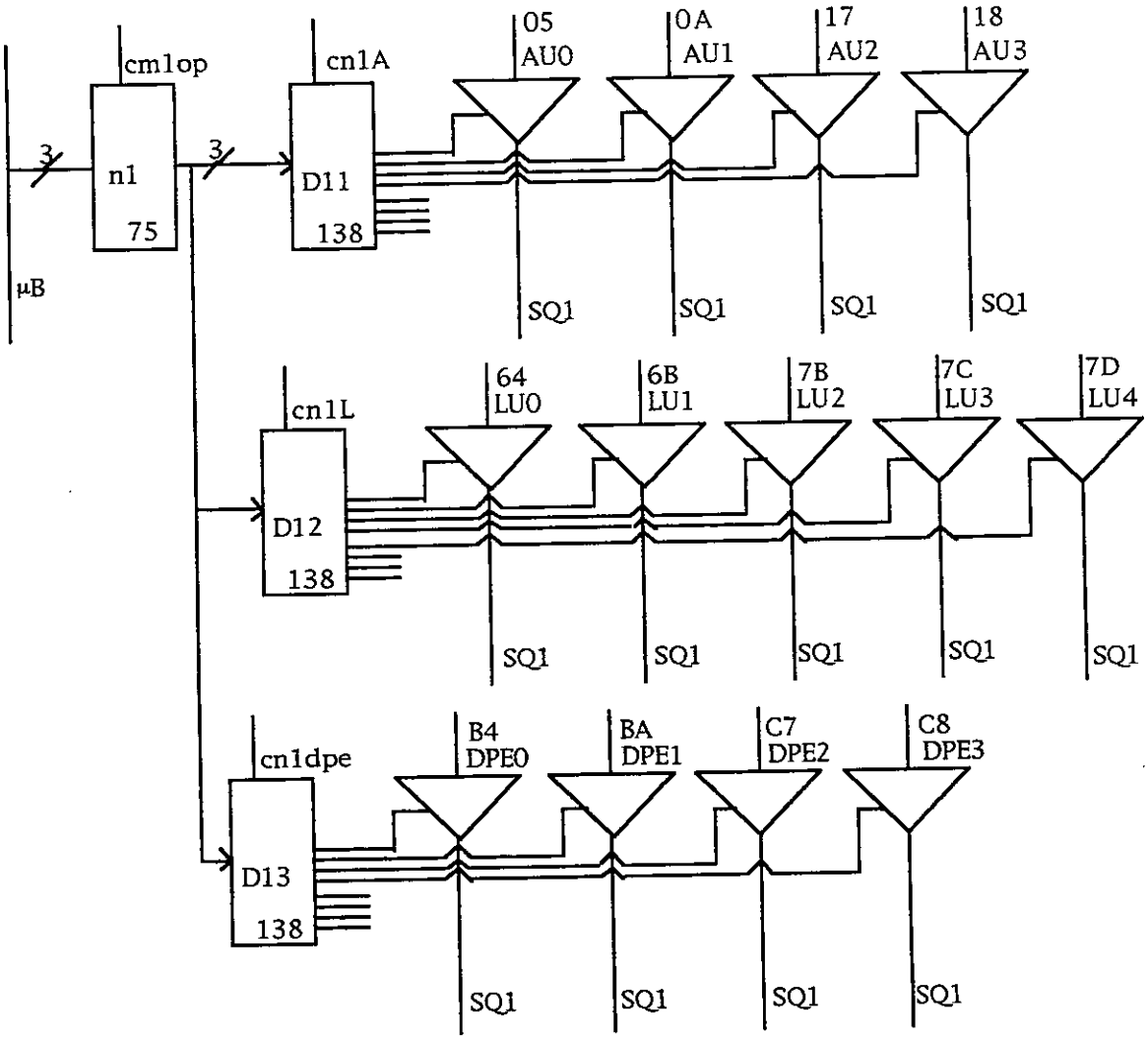
L'analyse de n3 se fait au moyen de la commande "cn3A" qui aiguille vers les 5 étiquettes:
AT0, AT1, AT2, AT3, AT4, chacune d'elles correspond, respectivement aux opérandes: $\lceil_a$, $\lceil_{a,b}$, $\lceil_I$, $\lceil_\omega$ ou la constante.

Une cellule t1 sert pour le calcul d'adresse qui correspond aux opérandes de la forme: $\lceil_I$, $\lceil_\omega$. pour les distinguer des trois autres cas dont le calcul consiste à évaluer une adresse valable en mémoire. La cellule t1 prendra donc trois valeurs:

0: Le résultat du calcul est une adresse en mémoire,
1: On traite directement avec la case ω ,
2: On traite avec la case d qui joue le rôle de I.

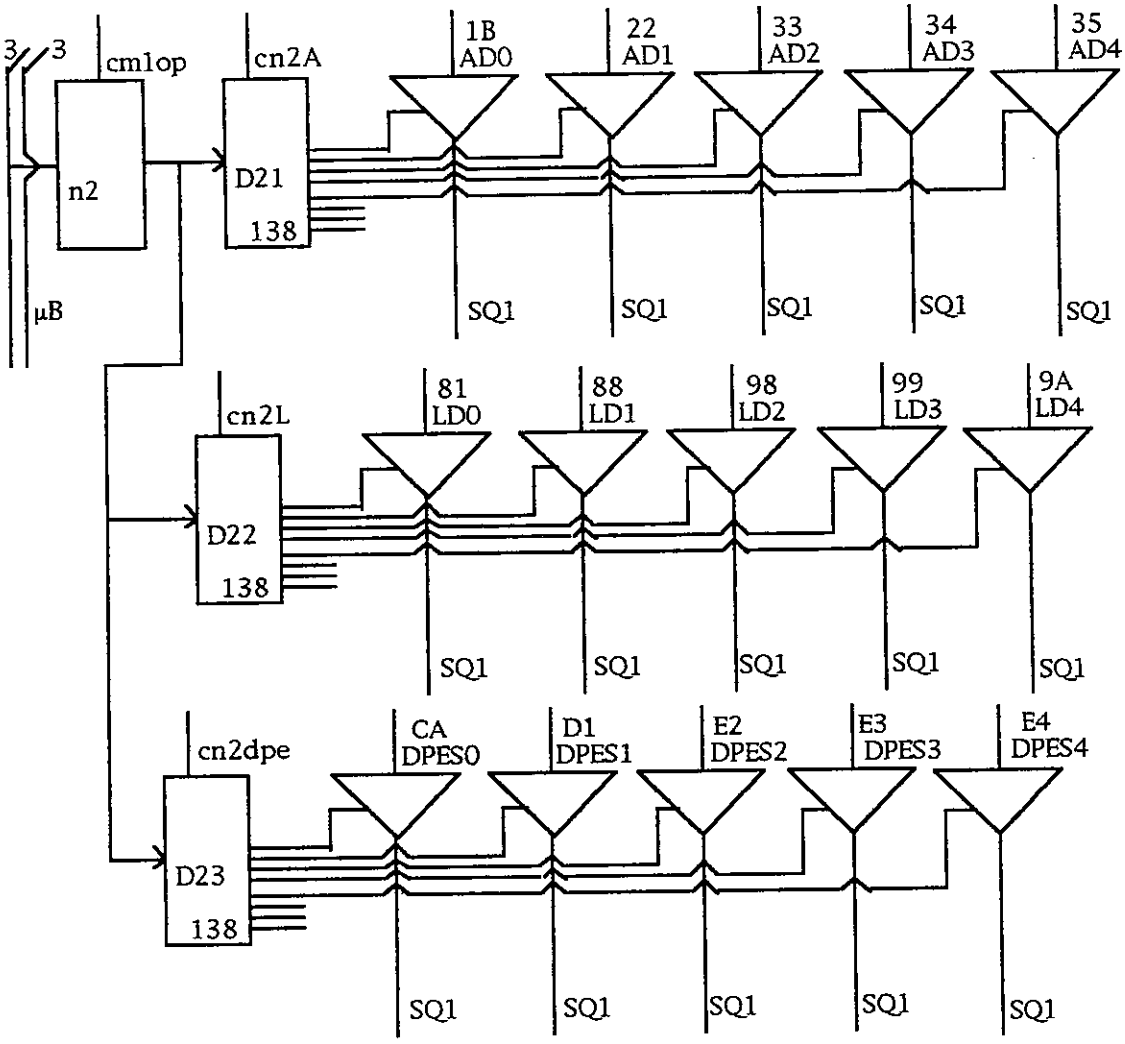
La cellule t1 est mise à zéro avant chaque calcul d'adresse d'opérande, et il est ensuite mis à 1 ou à 2 sur la rencontre du type respectivement de $\lceil_\omega$ ou de $\lceil_I$.

Les schémas des analyseurs des valeurs de n1, n2, n3 sont les suivants:



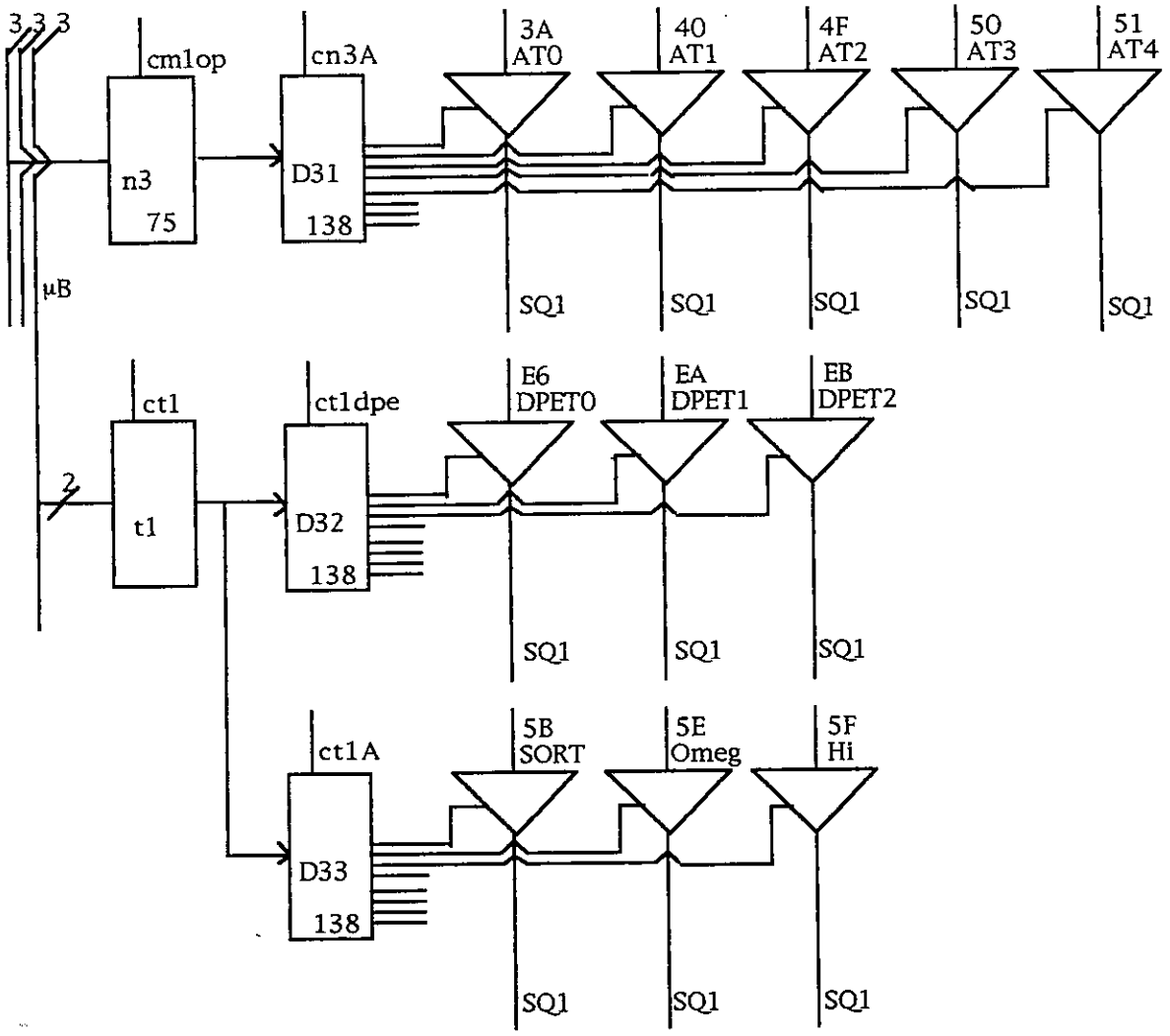
Analyse de n1.

L'analyse de n2 se fait avec le circuit:



Analyse de n2.

L'analyse de n3 et de t1 se fait, elle, avec le circuit:



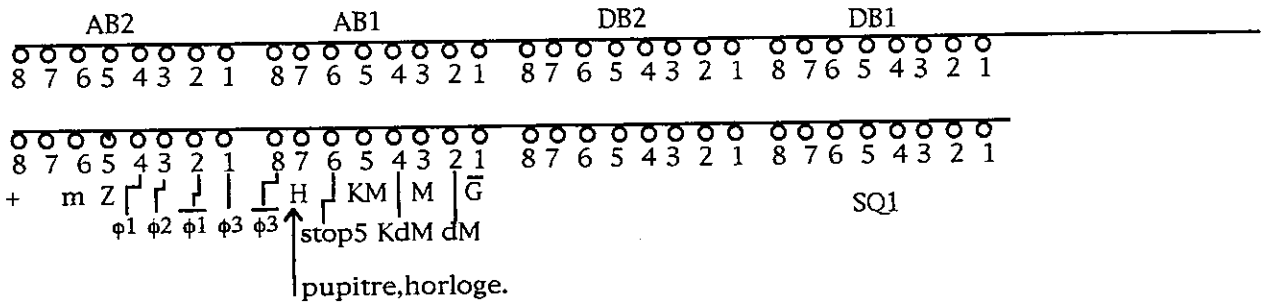
Analyse de n3.

Ainsi, selon qu'on a affaire à une simple affectation, deux opérandes, ou une instruction arithmétique, trois opérandes, on agit soit sur le décodeur D32, soit sur le décodeur D33. Dans le premier cas, c'est la commande "ct1dpe" qui actionne D32. Et alors selon que t1 contient 0, 1 ou 2, c'est le buffer DEPET0, ou DPET1, ou DEPET2 qui est ouvert. Cela signifie que le calcul qui doit suivre se trouve en adresse hexadécimale E6, ou EA, ou EB, respectivement, du programme interne. En E6 c'est un calcul d'adresse en mémoire, en EA on traite de la case report, et en EB, de la case qui joue le rôle de I.

CONNECTEUR.

Chaque carte est reliée par un connecteur au fond de panier, qui, comme on le sait est le bus commun à toutes les cartes. On peut observer le connecteur soit du côté de la connection qui s'enfiche

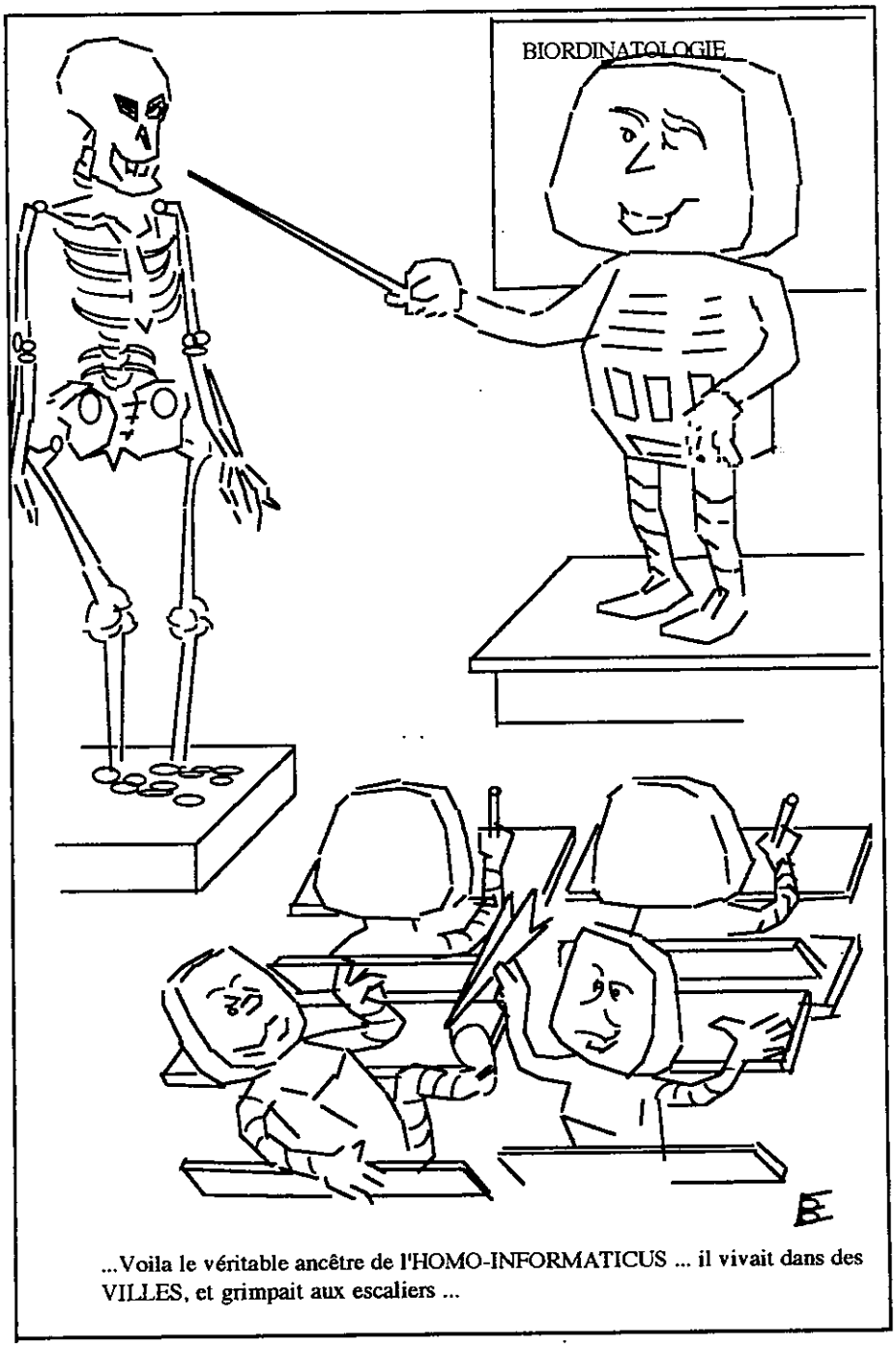
dans le fond de panier, soit du côté des broches fixées sur la carte. Donc, vu côté broches, les signaux se répartissent comme suit:



Avec AB1, AB2 le bus adresse, DB1, DB2 le bus des données, SQ1 le bus à 8 moments du séquenceur, destiné à transmettre l'adresse suivante calculée. L'alimentation + pour le 5 volts régulés, et m la masse. Les signaux d'horloge, la commande du pupitre, et les diverses commandes de la mémoire.

A suivre.

DOUZZAVEDIBISAR



...Voila le véritable ancêtre de l'HOMO-INFORMATICUS ... il vivait dans des VILLES, et grimpait aux escaliers ...

Communication moderne.

Une curieuse pratique semble se répandre avec une certaine vigueur, qui paraît d'ailleurs liée avec l'art de la pub. On trouve des petites phrases toutes prêtes, mais dans lesquelles un mot, en général un verbe, est remplacé par un petit dessin. Le sujet est représenté dans une autre langue qui lui donne aussi l'aspect d'un petit dessin. Seul le complément est écrit comme d'habitude, par exemple:

I ♥ mer

qu'on traduit instantanément par: j'aime la mer. On est en train de réinventer les idéogrammes, les kanji du japonais. Etonné que personne ne l'ait encore fait, je me sens obligé de proposer un antonyme à ce verbe. Par exemple:

I  Giscard, Chirac, Mitterand ...

Qui pourrait se traduire, s'adressant à qui-de-droit, par la formule: je t'emmerde. Voila un petit jeu innocent qu'il me paraît intéressant de développer, aussi je propose d'ouvrir cette colonne à tout amateur qui aurait des idées (ogrammes) allant dans ce sens.

E.B.