LABORATOIRE D'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L. I. T. A. M.
DEPARTEMENT DE MATHEMATIQUE-INFORMATIQUE
LUMINY
UNIVERSITÉ AIX-MARSEILLE II

ISSN 0291 - 5413

INFORMATIQUE FONDAMENTALE ET APPLICATIONS

Comité de rédaction:

E. Bianco
R. Cusin
P. Isoardi
J.P. Lehmann
R. Stutzmann

Dépositaire:
G. Ambard

Sommaire

Décembre 1984

ADRESSE POSTALE : FACULTÉ DES SCIENCES DE LUMINY
MATHÉMATIQUE-INFORMATIQUE - LITAM - BAT. TPR 2 - 9ème ETAGE
CASE 901 - 70, ROUTE LEON LACHAMP - 13288 MARSEILLE CEDEX 9 - ☎ (91) 41.01.40 POSTE 3224 - 3110

Editorial

e. bianco

Informatique et paradoxe

Dire à brûle-pourpoint que l'informatique est un paradoxe aurait de quoi choquer,encore qu'avec la perspective de la Guerre des étoiles (notons la "charge" poëtique de cette expression) il faille s'attendre à beaucoup de choses surprenantes.Mais dire que l'informatique est la conséquence d'un paradoxe,pourrait laisser davantage perplèxe.

Raisonnablement -mais qu'est-ce que la raison ?- on peut imaginer justement que cet ilôt de connaissances a surgi logiquement -mais selon quelle logique ?- dégagé par la raison au moment opportun.

Le calcul -arithmétique,je précise- est né avec le pouvoir. La nécessité s'est fait sentir très tôt de compter ses richesses dès qu'il y a eu richesses.A la fois pour les constater toujours présentes,et puis aussi pour évaluer celles du voisin afin d'en prélever,si possible,une bonne partie.Aux fins déclarées,bien entendu,d'en assurer la protection,par exemple.Finesse dont il est impossible de dater l'apparition,mais qui a dû s'imposer rapidement pour faciliter la transaction.

Dites comme ça,simplement,on pourrait croire que ces pratiques remontent aux temps reculés où la vie sociale était rudi-

mentaire.Mais il suffit d'observer d'un peu près,et on les retrouve avec toute leur fraîcheur dans la vie moderne.Seulement elles ne portent pas le même nom selon que l'on consulte le code pénal ou le code fiscal,ou encore le code du milieu.

Bien sûr,on n'obtient pas des autres une partie de leurs biens -chèrement acquis- sur simple réquisition,sauf en cas de guerre,mais encore l'issue est là souvent hasardeuse.

Alors des lois sociales se créent,complèxes,évolutives,qui canalisent les richesses donc: le pouvoir.On vend ses filles, on échange des avantages abstraits contre d'autres bien concrêts etc. De dot en héritage,d'héritages en spoliation,les biens ont floculé et se sont accumulés en des points précis.La complèxité des lois facilite la division,qui,à son tour pousse à rechercher l'arbitrage,tout prêt tout organisé mais qui coûte cher.

Le contraste entre l'excès de présence de biens,et forcément en face l'excès d'absence de biens provoque des tentations.C'est inévitable.Alors la parade surgit sous deux formes inégalement efficaces: création d'une police,mercenaires et grandes compagnies en tous genres,fabrication d'un état d'esprit respectueux à l'aide d'une morale.Sur ce dernier chapitre tout est bon,et plus c'est criant d'injustice flagrante,plus ça a des chances de marcher.L'efficacité est dans l'outrance.Cette vérité a des références: Herr Doctor Goebbels déclarait que plus le mensonge est gros plus il a de chances d'être cru.

Il n'y a pas comme un "bon pauvre" pour être"honnête" et donc "respectueux" de la "richesse établie".Ce coup-ci je cite la Comtesse de Ségur.Le tout est donc de convaincre ce personnage que s'il fait montre d'une "bonne conduite" lui aussi,plus tard aura droit à un morceau du gâteau.Mais pas trop gros car il est bien connu que la richesse gâte le caractère.Il est bien

connu également qu'il faut être un vrai pauvre pour croire à de telles vérités.Peut-être la non miscibilité des langues a-t-elle facilité les choses.

Il est très curieux d'observer à ce sujet comment une néces. sité vitale à une certaine époque,et d'autant plus dure à faire admettre,peut devenir à une époque où elle n'est plus aussi aigüe,un objet de vénération par intégration à un certaine morale. Exemple: le droit au travail.

Mais,me direz-vous et l'informatique dans tout ça ! elle arrive,la voila !

L'accumulation de la richesse ne se fait pas sans risques.Il faut spolier mais pas trop,car un estomac trop creux perd ses oreilles,et sans oreilles,plus de morale.Alors il faut supputer. Un mercenaire coûte cher et peut devenir à son tour dangereux, d'autant qu'on le choisit dans la bande d'en face et souvent parmi les pires.Evaluation du risque à courir,dosage au plus juste des frais et charges diverses,calcul de plus en plus minutieux de la dîme...Et l'on voit poindre de nouveaux corps de métiers où s'accumulent les richesses: les assurances,les banques,armées d'une méthode: la statistique,et d'un moyen: l'informatique.

Au fur et à mesure que l'information circule,et elle circule, ne serait-ce que par ce qu'on appelle les média,le "vulgum pecus" prend de plus en plus conscience de ce qui passe à portée de sa main,aussi faut-il reculer de plus en plus dans l'abstraction.La monnaie était déja une abstraction,la monnaie de papier, donc sans valeur intrinsèque,en était une plus forte encore, puis les chèques,maintenant la carte de crédit.

Qui a dit: seul,le pauvre est obligé d'avoir de l'argent ?

La fortune de l'individu devient un nombre perdu -pas pour

La fortune de l'individu devient un nombre perdu -pas pour tout le monde- au fond d'un ordinateur,et une soustraction sur ce nombre est moins douloureuse que de se voir arracher son porte-monnaie plein de piécettes.

La science est née de la nécessité pour une classe sociale d'établir son pouvoir.Ainsi le développement de la manufacture puis de la fabrique,puis de l'usine a éxigé l'étude de la matière,au XIX^{e} siècle de simples industriels se sont révélés des découvreurs,l'université passait de la philosophie à la science. Puis celle-ci s'est développée très largement au-delà de ses origines,mais malgré tout dans le même cadre social.

Peut-être m'objecterez-vous que parfois la connaissance s'est développé par curiosité pure comme l'astronomie,par exemple, mais oui,c'est bien ce que je disais: l'astronomie a permis de mesurer avec quelle précision le temps de travail,cette denrée chère.

Et la science a tendu vers l'abstraction.Toute connaissance est abstraite par essence,car c'est nécessaire à son assimilation par l'esprit.Mais là c'était l'abstraction afin de se rendre inaccessible au plus grand nombre.Les spécialistes bâtissaient les arcanes de la science,organe de pouvoir.La connaissance se condensait dans un jargon impénétrable.L'exemple le plus frappant est celui des mathématiques: voila une branche de la connaissance composée exclusivement d'évidences à partir du moment où le problème est posé.A constater la quantité d'échecs qui sanctionnent son apprentissage,on est sidéré de l'efficacité de la langue de bois et de sa manipulation,et il faut reconnaitre que bien des spécialistes eux-mêmes s'y sont complètement paumés.

Et l'informatique est née là,au milieu.Combien de ventres

ont revendiqué cet accouchement ?
l'économie ? la physique ? la physique théorique ? la mathématique ? le ministère de l'intérieur ?

Comme toujours en france,c'est une cotte mal taillée qui lui a servi de layette.Elle a été collée,noblesse oblige,aux mathématiques pour ce qui est de la théorie.Mais pour ce qui est de l'outil,le centre de calcul,on a trouvé à sa tête à peu près n'importe quoi,mais surtout l'administration.Jusqu'à ce qu'on soit à peu près sûr de disposer de suffisamment de scientifiques ratés,donc inconditionnels de l'administration,pour en faire d'honnorables directeurs.

La notion de paradoxe est-elle paradoxale ? prenons l'exemple d'Epiménides le Crêtois.Ce n'est pas l'état mental d'Epiménides,ni ce qu'il affirme qui est en cause,mais bien la notion de mensonge.Et il n'y a problème "logique" que dans la circonstance caricaturale suivante: Epiménides ment précisément au moment où il porte un jugement sur tous les Crêtois.

Or,pour en revenir au mensonge,que pourrait-on dire de quelqu'un qui ment tout le temps.Il y aurait,là,je crois,un véritable paradoxe lié à une impossibilité matérielle et psychologique.Le mensonge est un acte qui vise à l'éfficacité et en tant que tel il doit rester discret.Sinon il s'agit d'un délire mythomaniaque qui déplace la question.

Un paradoxe ou un mensonge: le Centre Intergalactique de l'Iformatique ?

On va d'abord dérober "les meilleurs spécialistes américains" qu'on sort ensuite à coup de pied quand ils osent exprimer leur désaccord avec "l'administration française".

Quelle que soit la majorité en place et donc l'opposition

qui lui fait face,il ressort du discours officiel que les efforts tendent à l'amélioration de l'économie,en contrepoint le discours opposant démontre ce qui serait beaucoup mieux pour arriver au même résultat.Donc tout le monde souhaite une meilleure économie,source de tous les bonheurs.Mais cela passe par l'avènement du Robot destiné à remplacer l'homme dans les tâches ingrates.Oui dans le Principe.En fait même un esprit qui n'est pas mal tourné constate que l'automatisation du balayage des ordures a bien moins progréssé que le contrôle et la gestion d'immenses entreprises traitant d'objets plus nobles.

D'un point de vue plus profond et le succès de la Science-Fiction le montre parfaitement le robot est appréhendé à la fois comme l'être intelligent par essence et totalement soumis dont chacun rêve qu'il agirait à sa place et pour son compte. Prothèse idéale,synthèse absolue du pouvoir sans danger.

Et puis sous l'angle de la domination spirituelle,cette oreille largement ouverte à la voix de son maître,a de quoi faire rêver les grands esprits qui ont du mal à se faire entendre du rustre,malgré le pouvoir exorbitant des médias.Déja au III^{e} siècle avant notre ère,un grand empereur chinois Tsin che-houang Ti avait appliqué ces excellentes idée avec succès ce qui lui avait valu de réussir l'unification de la chine.(cf. Le Grand Empereur et ses automates. Par Jean Levi.Albin Michel)

L'informatique est devenu le merveilleux outil égalisateur dont peuvent rêver les guides en tous genres.Déja percepteur infaillible,en train de devenir l'inquisiteur intransigeant,en attendant de remplacer cette triste engeance humaine,si remuante,si vaine,si imparfaite.

Paradoxe ? mensonge ? Pour le meilleur profit de quel tyran ?

NOTION DE SYSTEME

Système logiciel et systeme machine

C.R. Subject classification informatics 4.12 4.21 4.22

Résumé.

A propos de la machine universelle de la procédure formelle, on étudie ici comment se présente la notion de système.Après avoir construit un ordinateur qui utilise cette machine universelle comme procésseur,on est tout de suite confronté à cette notion.On va donc la matérialiser sous diverses formes au fur et à mesure qu'on va vouloir résoudre des problèmes de plus en plus larges.

NOTION DE SYSTEME

SYSTEME LOGICIEL,SYSTEME MACHINE.

INTRODUCTION.

Dans cet article j'aimerais matérialiser un peu ces deux notions de système-machine et de système-logiciel,telles qu'elles apparaissent quand on construit une machine d'une part,pour qu'elle puisse raisonnablement fonctionner,et quand on la munit d'un système logiciel pour pouvoir l'utiliser convenablement d'autre part.De toute évidence,le rapport qui existe entre ces notions est de même nature que celui qu'on imagine entre des notions de langage et des notions de métalangage.Il est donc bon de distinguer pour éviter de graves confusions.

Pour raisonner je vais utiliser la transition qui existe entre machine simple et machine universelle.Je pars d'un langage qui me permet de construire des machines simples: la Procédure Formelle.Avec ce langage,je me propose de construire une machine simple qui soit machine universelle pour ce même langage. Démarche bien connue de Von Neumann,mais que se passe-t-il quand on met en oeuvre la puissance de cette notion? inévitablement on en profite pour introduire dans le langage des notions importantes par ailleurs mais qui sont faciles à prendre en compte par la machine universelle.De telle manière que,partant d'une sorte de langage minimum,suffisant pour décrire cette machine, on va pouvoir soumettre à la machine un langage plus complet et partant,plus souple.Tout programme apparaît alors comme un sim-

ple objet étalé dans la mémoire,et des objets d'ordre algorithmique,donc jusque là intouchables vont pouvoir être atteints et soumis au calcul.Alors prudence! pas n'importe quel calcul.Les premières personnes qui découvrirent -il y a déja quelques décennies- le "calcul d'instructions" ,toutes à leur joie se mirent à construire des programmes évolutifs,dont rapidement le développement devenait incontrôlable,à la grande joie,ce coup-ci,des témoins amusés.

La définition de la procèdure,objet doté d'une certaine indépendance,qui se manifeste pour le nom des paramètres,entraîne la définition de l'insertion de la procédure.La pratique pousse alors,économie oblige,à se demander s'il ne serait pas possible d'accroître cette indépendance dans le sens de la réduction à une seule,les occurences du corps de la procédure,c'est-à-dire du code qui représente son algorithme.Cela signifie qu'il ne sera plus inséré en bloc mais "représenté" dans le texte par ce que nous appellerons une instruction d'insertion de procédure.

Cette instruction n'existe pas en machine élémentaire,et il est difficile de la simuler.Mais la construction de la machine universelle montre que dès qu'on dispose non pas de l'instruction mais de son code,alors il est facile d'en simuler le jeu.

Une difficulté apparaît alors.J'imagine plusieurs procédures enchaînées les unes aux autres par l'insertion,le code de l'ensemble est introduit à la main en mémoire,et je prépare,également à la main le démarrage du calcul.Que va-t-il se passer quand la machine universelle tombera sur le code du <u>fin</u> de la procédure insérée ainsi à la main ? Il est visible qu'il existe deux sortes de <u>fin</u> ,selon que la procédure aura été insérée à la main ou bien insérée automatiquement par la machine universelle.Le problème ainsi soulevé est celui de l'enchaînement du dé-

roulement de deux ou plusieurs programmes consécutifs.Le déroulement se décompose en une série d'étapes: les insertions successives.La suite des insertions appartient au domaine du fini illimité,car très généralement le nombre d'insertions dépend de la sémantique du programme.Mais l'insertion la plus extérieure se trouve être de même niveau pour deux programmes différents.

L'automatisation de cette insertion concerne une partie de la notion de système,l'autre partie en est relative à l'automatisation de la mise en place en mémoire des codes programmes et des configurations.

LE LANGAGE.

Je rappelle rapidement la structure du langage de la procédure formelle élémentaire,qui procède directement de la Machine de Nolin (Machine à Cases Adressables).On observe une double transformation:

1) L'apport d'une case spécifique supplémentaire "I" dont le contenu est le nom de l'une des cases de la mémoire centrale et les valeurs possibles des contenus de I sont les noms de toutes les cases de la mémoire centrale.

2) La désignation des opérandes qui se réfère à la case I à l'aide de l'écriture définie plus bas,et selon un algorithme longuement explicité dans l'ouvrage [1].

J'admet que je sais construire les algorithmes qui représentent l'arithmétique modulaire des opérateurs: + , - , * , / qui me suffisent.Rien d'autre ne me sera nécessaire pour construire la machine formelle.

[1] De la Machine de Turing aux Ordinateurs Modernes
Birkhaüser Verlag 1979 E. Bianco.

MACHINE FORMELLE

MACHINE UNIVERSELLE DE LA PROCEDURE FORMELLE.

LE LANGAGE.

Je rappelle rapidement la forme du langage admis.Je désigne les opérandes par "x" et "y".

T1

début

fin

x := y

x := y A y

si y R y vers ei

vers ej

insérer NP (LPEF)

T2

x { $[_\alpha$; $[_{\alpha,\beta}$; $[_\omega$; $[_I$

y { x ; constante

A représente les opérateurs arithmétiques: + , - , * , /

R représente les opérateurs de relation: = , ≠ , > , ≥ , < et ≤ .

NP est l'identificateur du nom de la procédure insérée.

LPEF est la liste des paramètres effectifs.C'est une liste d'entiers représentant,chacun,le numéro d'une variable dans la configuration de la procédure insérante.

LE CODE.

Je choisis de mettre le code des instructions sous la forme suivante:

(op,n1,n2,n3) (A1) (A2) (A3) (A4) (A5) (A6)

La première parenthèse contient le descripteur de l'opération à réaliser,et les trois descripteurs d'opérandes.Les Ai sont les valeurs d'opérandes quand elles sont présentes,c'est-à-dire pour les deux premières formes de "x" et la deuxième de "y".

Les 4 paramètres de la première parenthèse sont ainsi codés:

T3

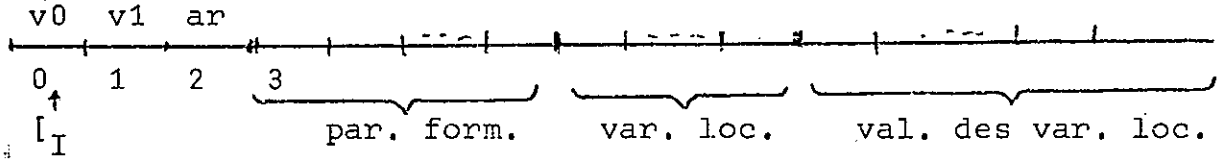
code op	opérateur
0	début
1	fin
2	:=+
3	:=-
4	:=*
5	:=/
6	:=
7	si=
8	si≠
9	si>
10	si≥
11	si<
12	si≤
13	vers
14	inserer

T4

n1,n2,n3	opérandes
0	$[_{\alpha}$
1	$[_{\alpha,\beta}$
2	$[_{\omega}$
3	$[_{I}$
4	Cons^{te}

STRUCTURE DE LA CONFIGURATION.

Pour systématiser le traitement des codes,d'une part et pour faciliter la prise en compte de l'insertion d'un autre côté,je suis obligé d'organiser la configuration.Voici donc le choix auquel je me suis arrêté:



En case 0 par rapport à l'origine,doit se trouver le volume de la configuration en cours de traitement (par opposition aux configurations qui en seront issues par insertion,par exemple)

En case 1 doit se trouver le volume de la configuration précédente,dans l'ordre des insertions de procédures.Je fais une hypothèse précise sur la façon dont les configurations se placent les unes par rapport aux autres au cours des insertions. Chaque fois que s'insère une nouvelle procédure,la configuration sur laquelle elle va travailler se place immédiatement à droite de la configuration sur laquelle travaille la procédure insérante.

On peut évidemment choisir d'autres lois de répartition des configurations,celle-ci me paraît la plus simple dans le cadre décrit,et de toute manière ce que je veux montrer ne dépend pas de cette loi.

En case 2 est placée l'adresse de retour dans la procédure insérante.

A partir de la case 3 et dans l'ordre,les cases sont considérées comme des paramètres formels,et leurs contenus: les paramètres effectifs,sont mis en place au moment de l'insertion par la machine formelle qui la déroule.

A la suite,sont alignées les variables locales.J'impose que ces dernières aient même structure que les paramètres: la case qui représente une variable locale contient une adresse qui renvoit à la valeur,laquelle se trouve dans une 4° partie de la configuration.De toute manière on ne peut pas faire autrement si la variable locale est un tableau.Reste le cas des variables locales simples qui pourrait être traité en adresse directe. Mais j'ai choisi l'uniformisation dans la représentation des variables locales et des paramètres afin de simplifier le traitement de l'insertion de procédure au moment où on charge les paramètres effectifs,en effet,ces derniers peuvent être puisés dans les deux sortes de variables.

STRUCTURE DU CODE DE L'INSTRUCTION.(Implications sur la mémoire)

Dans la première parenthèse du code,j'ai placé quatre valeurs. Les trois dernières caractérisent les opérandes,et le tableau T2 montre qu'il en existe 5 différents de forme.Les tableaux T3 et T4 montrent les maxima que peuvent prendre les "ni" et "op".La tentation est grande alors de mettre ces 4 valeurs dans une même case.Ceci introduit une contrainte qui s'exprime en nombre de bits minimum que doit contenir une case:

Il faut 3 bits pour chacun des ni,et 4 suffisent pour op. La case doit contenir au moins 13 bits.Je suppose que tel est le cas pour la machine que je vais décrire.Toute autre valeur supérieure ne changerait rien de fondamental à ce qui va suivre. Il faut tenir compte également des valeurs des Aj.Or,il peut s'agir soit d'indices soit de constantes.Les indices étant censés être toujours positifs,j'observerai,de plus qu'une case doit pouvoir contenir une adresse.Mais il est évident qu'il y aura des adresses négatives.Dans une représentation en complément à deux que j'adopte,cela signifie qu'en module on ne peut ainsi recouvrir que la moitié de la mémoire.Les valeurs négatives permettant d'atteindre l'autre moitié.Il est donc nécessaire de distinguer le traitement de l'indice de l'opérande,toujours positif,de celui des constantes et des adresses en représentation complément à deux.

Avant de donner quelques exemples je remarque que quand l'opérande s'adresse à une case unique comme "I" ou"Ø" il n'est pas nécessaire de lui faire correspondre un Ai car la valeur de ni (cf. T4) suffit à le caractériser.

Je vais donner l'image de codes d'instructions,mais sans aller jusqu'à les traduire en binaire,ce qui ne présente aucune difficulté.

instruction	codes
$[_3 := [_{5,10}$	(6,0,1,0) (3) (5) (10)
$[_i := [_\omega + 1$	(2,0,2,4) (1)
$[_{1,0} := [_{3,2} * [_{6,1}$	(4,1,1,1) (1) (0) (3) (2) (6) (1)

T5

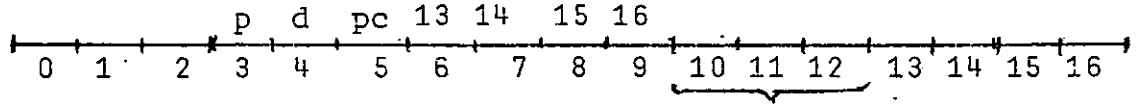
ALGORITHME DE LA MACHINE UNIVERSELLE.

Cette machine universelle,je vais la décrire avec la Procédure Formelle comme je l'ai annoncé plus haut.Ceci implique que je construis une procédure qui va s'appliquer sur une configuration de structure standard.Quand je parlerai d'origine sans autre précision ce sera de l'origine de cette configuration qu'il s'agira.

Je suppose que les codes des diverses instructions qui composent un algorithme sont alignés en mémoire les uns à la suite des autres,à partir d'une case repérée par une distance "p" de l'origine (celle que je viens de définir).Je définis de la même façon une distance "d" qui permet d'atteindre la configuration sur laquelle s'applique l'algorithme codé.

J'aurai besoin d'une adresse "pc" initialisée avec "p" mais utile pour l'exploration du code d'une même instruction.

Les cases 10,11 et 12 me servent à calculer les adresses par rapport à l'origine des opérandes de l'algorithme codé,lesquels font référence,bien entendu à l'origine de la configuration du programme objet."d" exprimant la relation qui existe entre les deux origines.



L'AIGUILLAGE DE LA MACHINE UNIVERSELLE.

On commence par analyser le code de l'opération,"p" désigne la première case du code d'une instruction,or cette case contient quatre informations,dont "op".C'est cette valeur qu'analyse l'aiguillage.

```
entrée : [6,0 := [3,0 / 2^9
         si [6,0 = 'début' vers edeb ;
         si [6,0 = 'fin'   vers efin ;
         si [6,0 = ':=+'   vers edpea ;
                   ':=-'   vers edpea
                   ':=*'   vers edpea ;
                   ':=/'   vers edpea ;
                   ':='    vers edpea ;
                   'si='   vers econd ;
                   'si≠'   vers econd ;
                   'si>'   vers econd ;
                   'si≥'   vers econd ;
                   'si<'   vers econd ;
                   'si≤'   vers econd ;
                   'vers'  vers ev ;
         si [6,0 = 'inser' vers eins ;
```

A1

A chacune des étiquettes auxquelles renvoit A1,se traite complètement le code d'une instruction.Après avoir achevé ce calcul,"p" aura évolué de façon à désigner la première case du code de l'instruction suivante et on se renvoit à "entrée".

Ce fonctionnement cyclique est en lui-même un système élémentaire qui assure le déroulement de tout ce qui peut être dé-

roulé.Mais quand tout a été déroulé,il ne peut de lui-même trouver la suite.

Contrôler les échanges avec l'extérieur exige un système logiciel un peu complèxe,qui dépasse les capacités de ce petit système purement machine.Mais nous verrons qu'il faudra encore un peu perfectionner ce système machine pour pouvoir réaliser la système logiciel.

INSTRUCTIONS DEBUT,FIN,INSERER.

Ces trois instructions sont liées par le jeu de l'insertion. C'est quand on rencontre un code d'instruction insertion qu'on va pouvoir créer la configuration de la procédure qui va être insérée.En effet à ce moment on connaît la liste des paramètres effectifs,donc on peut calculer les paramètres de la procédure insérée.Pour aller les mettre en place il faut connaître l'accès à sa configuration,dans le schéma ci-dessous,c'est la valeur v0,qui mesure le volume de la configuration acturlle.

Il devient évident que ce volume doit être mis en place au tout début du déroulement de la procédure puisque:

1) la configuration ne préèxiste pas,elle est créée à l'insertion,

2) c'est une information qui est propre à la procédure,elle est connue quand la procédure est construite.

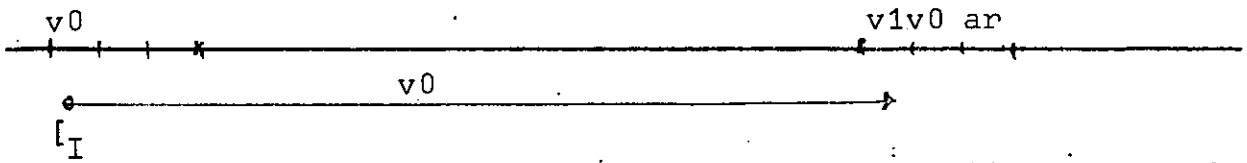
Il est donc raisonnable de loger cette information dans le code de début.On pourait aussi demander au programmeur de mettre lui-même cette information en place par une instruction:

$$\lceil_0 := v0$$

De toute manière on verra plus loin que la prise en compte des déclarations simplifie le problème.

Enfin c'est pendant l'insertion qu'on peut calculer l'adres-

se du retour en procédure insérante.Et c'est là qu'on va la mettre dans la troisième case de la configuration construite.Et c'est là qu'on ira la chercher,sur la rencontre du code de fin pour réaliser le retour.



La machine formelle qui rencontre le code de l'insertion réalise les opérations suivantes:

1) Mettre en place les paramètres effectifs dans la configuration en création.Le numéro du PE permet d'atteindre à l'adresse,qui,corrigée de la variation d'origine,est envoyée en v0+3, puis en v0+4,etc jusqu'à épuisement de la liste des PE.

2) Réserve le volume de la configuration actuelle: v0,en case 1 de la configuration créée,soit en v0+1.

3) Noter l'adresse de retour: ar,dans le code de la procédure insérante."ar" est placée en v0+2,troisième case de la configuration créée.

4) Faire évoluer "d" pour que la nouvelle origine soit sur la configuration créée.C'est la variation d'origine pour le programme traité.Cette variation est égale à v0.

5) Réaliser la commutation vers la procédure insérée.Le code contient par construction une distance à l'instruction début du code procédure insérée,calculée comme pour un vers.

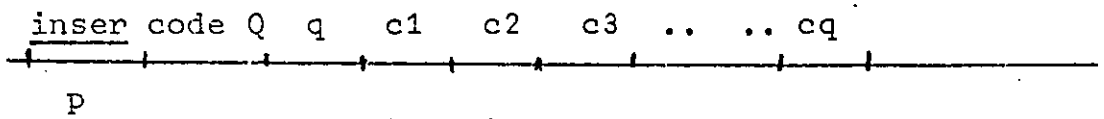
Sur la rencontre du code de fin,on revient à la procédure insérante par deux opérations:

1) Ramener l'origine sur la configuration de la procédure insérante en calculant "d" avec le contenu de la case1.

2) Utiliser "ar" pour remplacer "p".

La rencontre du code début fournit à la machine formelle la valeur: v0 à mettre en place dans la case 0 de la configuration qui vient tout juste d'être créée par l'insertion.

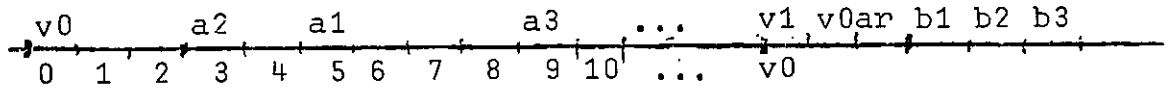
```
edeb :  [4,0 := [3,1                 v0 en case 0 de la conf.
        [3 := [3 + 2 ;
        vers entrée ;
efin :  [3 := [4,2 ;                 ar à la place de p
        [4 := [4 - [4,1 ;            d = d - v1
        vers entrée ;
eins :  [5 := [3 + 3 ;               calcul de pc
        [7,0 := [3,2 ;               récupération de q
        [9 := [4,0 + 3 ;
        [9 := [9 + [4 ;              v0+d+3 adresse des par. for.
eins2 : si [7,0 = 0 vers eins1 ;     q=0 ?
        [8 := [4 + [5,0 ;            d+ci
        [9,0 := [8,0 - [4,0 ;        bi=ai-v0 valeur du par. eff.
        [5 := [5 + 1 ;               pc+1
        [7,0 := [7,0 - 1 ;           q-1
        [9 := [9 + 1 ;
        vers eins2 ;
eins1 : [9 := [4 + [4,0 ;            d+v0
        [9,1 := [4,0 ;               v0 en case 1
        [9,2 := [5 ;                 ar en case 2
        [4 := [4 + [4,0 ;            d=d+v0
        [3 := [3 + [3,1 ;            p+codeQ
        vers entrée ;
```



Code d'une insertion.

Exemple:

insérer Q (5 , 3 , 9) ;

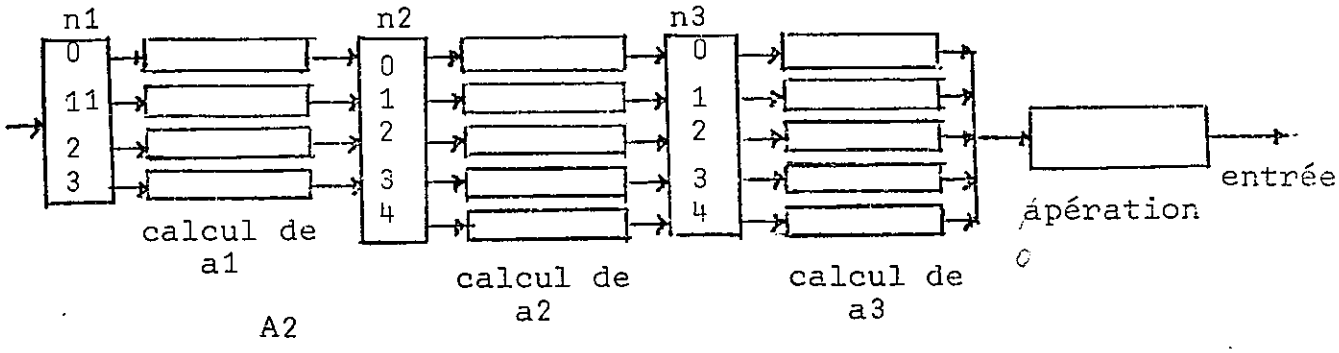


Les adresses: a1,a2 et a3 sont transmises en b1,b2 et b3 avec b1=a1-v0 , b2=a2-v0 et b3=a3-v0 .

ARITHMETIQUE ET AFFECTATION.

Après avoir détecté un opérateur d'affectation suivi ou non d'une opération arithmétique,on va se préoccuper du type de chacun des opérandes.On explore successivement chacune des valeurs: n1,n2,n3 (sauf pour l'affectation simple où on ignore n3) et on utilise les valeurs correspondantes Ai,quand elles existent,pour calculer l'adresse de l'opérande par rapport à l'origine de la configuration de la machine formelle.Les cases 10,11 et 12 servent à stocker les trois résultats de ces calculs en vue de leur exploitation par une instruction d'affectation et arithmétique de la machine formelle.

Le schéma de traitement est le suivant:



edpea : $[_{7,0} := [_{3,0} * 2^4$;

$[_{7,0} := [_{7,0} / 2^{10}$;

$[_5 := [_3 + 1$;

si $[_{7,0} = 0$ vers edpea0 ;

```
          si [7,0 = 1 vers edpea1 ;
                  = 2      edpea2 ;
          si [7,0 = 3      edpea3 ;
```

Erreur 1

```
edpea0 : [10 := [4 + [5,0 ;
         [5 := [5 + 1 ;
         vers edpea05 ;
edpea1 : [10 := [4 + [5,0 ;
         [10 := [10,0 + [5,1 ;
         [10 := [10 + [4 ;
         [5 := [5 + 2 ;
         vers edpea05 ;
edpea2 : [10 := [13 ;
         vers edpea05 ;
edpea3 : [10 := [4 ;
         vers edpea05 ;
edpea05: [7,0 := [3,0 *: 2^7 ;
         [7,0 := [7,0 / 2^10 ;
         si [7,0 = 0 vers edpea00 ;
                 = 1      edpea01 ;
                 = 2      edpea02 ;
                 = 3      edpea03 ;
         si [7,0 = 4 vers edpea04 ;
```

Erreur 2

```
edpea00: [11 := [4 + [5,0 ;
         [5 := [5 + 1 ;
         vers edpea55 ;
```

```
edpea01: [11 := [4 + [5,0 ;
         [11 := [11,0 + [5,1 ;
         [11 := [11 + [4 ;
         [5 := [5 + 2 ;
         vers edpea55 ;
edpea02: [11 := [13 ;
         vers edpea55;
edpea03: [11 := [4 ;
         vers edpea55 ;
edpea04: [11 := [5 ;
         [5 := [5 + 1 ;
         vers edpea55 ;

edpea55: si [6,0 = ':=' vers edpea56 ;
         [7,0 := [3,0 * 2^10 ;
         [7,0 := [7,0 / 2^10 ;
         si [7,0 = 0 vers edp10 ;
                 = 1      edp11 ;
                 = 2      edp12 ;
                 = 3      edp13 ;
         si [7,0 = 4 vers edp14 ;
```

Erreur 3

```
edp10 :  [12 := [4 + [5,0 ;
         [5 := [5 + 1 ;
         vers edp555 ;
edp11 :  [12 := [4 + [5,0 ;
         [12 := [12,0 + [5,1 ;
         [12 := [12 + [4 ;
         [5 := [5 + 2 ;
```

```
        vers edp555 ;
edp12 : [12 := [13 ;
        vers edp555 ;
edp13 : [12 := [4 ;
        vers edp555 ;
edp14 : [12 := [5 ;
        [5 := [5 + 1 ;

edp555 : si [6,0 = ':=+' vers edp555+ ;
                   ':=-'      edp555- ;
                   ':=*'      edp555* ;
         si [6,0   ':=/' vers edp555/ ;


edpea56: [10,0 := [11,0 ;
         [3 := [5 ;
         vers entrée ;


edp555+: [10,0 := [11,0 + [12,0 ;
         vers edpfin ;
edp555-: [10,0 := [11,0 - [12,0 ;
         vers edpfin ;
edp555*: [10,0 := [11,0 * [12,0 ;
         vers edpfin ;
edp555/: [10,0 := [11,0 / [12,0 ;
         vers edpfin ;
edpfin : [3 := [5 ;
         vers entrée ;
```

Les emplacements marqués: Erreur 1,Erreur 2 et Erreur 3 ainsi que la fin de l'algorithme de l'aiguillage (A1),correspondent à une valeur de "op" ou de "ni" inadéquate.En fait,les valeurs: 15 pour op ou 5,6 et 7 pour les ni.Cela pourrait correspondre soit à une erreur de codage,soit à un mauvais cadrage de "p",l'adresse d'accès au code programme,sur le code de l'instruction.La prise en compte de tels cas peut se faire dans l'ordinateur formel,comme on le verra plus loin.

Il est bon ici,de placer trois remarques qui me paraissent importantes.L'une porte sur la dissociation du champ linguistique et du champ métalinguistique,l'autre sur les conséquences de l'organisation de la configuration,la troisième sur le rôle des "registres".

Remarque 1.

Origine des configurations: programme-objet,machine universelle.

Il faut bien observer que si,dans le texte,je désigne l'origine de l'une ou l'autre de ces configurations par l'écriture: $[_I$,quand je me place du point de vue de la rédaction de la machine universelle,seul,le $[_I$ de la configuration de cette machine existe,l'autre est simulé.

Le programme objet,donc: codé,est construit pour travailler normalement sur sa propre configuration,donc ses opérandes sont normalement codés par rapport à l'origine de cette configuration.C'est pourquoi la machine universelle doit calculer ces opérandes-là en les reportant à sa propre origine.C'est "d"qui permet la translation,et les résultats sont placés dans les cases 10,11 et 12 pour leur utilisation au sein de la machine universelle.

Remarque 2.

Gestion des insertions.

Il est bien connu que l'insertion de procédure oblige à gérer la mémoire de façon particulière.L'adresse de retour dans l'algorithme se traite bien avec une pile.Un traitement similaire pourrait s'appliquer à l'empilement des configurations,mais il faut tenir compte là,qu'il s'agit de groupes d'éléments et non plus d'éléments simples.

C'est pourquoi le choix de structure de la configuration de Procédure Formelle est important,quand il rassemble dans des cases spécialisées (les trois premières) l'information suffisante pour assurer la gestion de l'insertion.Une seule organisation "stack-like" suffit alors.

Remarque 3.

Registres du "processeur" machine Formelle.

Dans les processeurs habituels on reconnait deux sortes de "registres" : ceux dont on sait qu'ils existent,mais qu'on ne peut programmer directement,et ceux qu'on peut atteindre par programme.Dans ce deuxième genre,il y a ceux qui ne servent qu'à la programmation et ceux qui peuvent à la fois être programmés de l'extérieur et traités de l'intérieur par le processeur.Comme exemple de ce deuxième genre on a le registre ordinal,sur lequel on peut,selon les modèles intervenir et qui sert au processeur pour marquer le code programme.

En procedure formelle,les registres du processeur,c'est-à-dire de la machine formelle sont ceux qui sont définis dans sa configuration.Quatorze cases m'ont été utiles pour la réalisation décrite plus haut.Bien sûr,si je change mes hypothèses de construction,le nombre de cases risque également de changer.J'aurais

pu aussi en calculant au plus juste en économiser peut-être quelques uns.L'essentiel est de constater la correspondance entre structure du langage et hypothèses de mise en place et le rôle des cases de la configuration spécifique de la machine formelle.

Quand au programme objet on constate que sa configuration comporte Trois cases qui ne servent qu'au "processeur",à laquelle il faut ajouter I,ce sont des registres auxquels il ne doit pas toucher.Mais il dispose d'une liste illimitée de cases d'usage commun,ce sont les paramètres formels,au moins pour la procédure la plus extérieure.Par contre ces cases ont un rôle parfaitement imposé.

DECLARATIONS EN PROCEDURE FORMELLE.

Je vais distinguer trois sortes de variables,de par le rôle particulier qu'elles jouent dans une configuration:

1) Les paramètres formels.
2) Les index.
3) Les variables locales.

Paramètres formels et variables locales sont la matérialisation informatique du nom des variables sur lesquelles s'applique l'algorithme.Mais l'existence de tableaux exige aussi bien sur l'un que sur l'autre type de variable,des calculs particuliers sur le nom pour désigner l'élément,ce sont les index qui servent à ces calculs locaux.

On porte dans l'ordre: les déclarations des paramètres formels,des index et des variables locales.

En paramètres et index,on précise le nombre de variables,en variables locales,il faut préciser le nombre de variables et le

volume de chaque variable.

Exemples:

```
début
paramètres formels 3 ;
index 2 ;
variables locales 3 :1 , 1 , 1 ;
etc...

début
paramètres formels 2 ;
index 4 ;
variables locales 5 : 10 , 5 , 1 , 1 , 25 ;
etc...
```

Je vais choisir les codes suivants pour représenter ces deux déclarations:

```
decl 3  2  3  1  1  1
decl 2  4  5  10 5  1  1  25
```

Soit le code opération,suivi du nombre de paramètres,du nombre d'index,du nombre de variables locales et suivi de la liste des volumes de ces dernières.

MACHINE UNIVERSELLE ET DECLARATIONS.

L'algorithme qui prend en compte ces déclarations va pouvoir mettre en place dans la case N° 0,le volume de la configuration et également calculer et mettre en place les adresses des variables locales.Tous ces actes ne sont donc plus à la charge du programmeur,et on modifie aussi l'instruction début qui n'a plus besoin de contenir le volume de la configuration.A la fin

du tableau T3 il faut rajouter une ligne:

15 décl

Dans l'aiguillage il faut rajouter une instruction:

si $[_{6,0}$ = 'decl' vers edecl ;

```
edecl  : [5 := [3 + 1 ;
         [10 := [5,0 + 3 ;
         [10 := [10 + [5,1 ;
         [11 := [10 ;          Variable locale encase11.
         [10 := [10 + [4 ;     adresse de la var. loc.,
         [9,0 := [5,0 ;        dans la configuration.
         [5 := [5 + 2 ;
edecl2 : si [9,0 = 0 vers edecl1 ;
         [11 := [11 + [5,0 ;   Chargement des adresses
         [10,0 := [11 ;        des variables locales,dans
         [9,0 := [9,0 - 1 ;    la configuration.
         [10 := [10 + 1 ;
         [5 := [5 + 1 ;
         vers edecl2 ;
edecl1 : [4,0 := [11 + [5,0 ;  Volume de la configuration
         [3 := [5 + 1 ;        placé en case 0 de cette
         vers entrée ;         configuration.
```

Le petit effort qui consiste à déclarer les variables de la configuration,est bénéfique,somme toute sur un autre plan,puisqu'il oblige un peu à classer ses variables,économise les efforts plus complèxes de calcul et de mise en place des adresses locales et du volume de la configuration.

La mise en place du volume de la configuration est une con-

séquence de l'instruction d'insertion.Mais pour les adresses des variables locales,on paye le prix de l'uniformisation de leur traitement avec celui des paramètres: faisant ainsi on simplifie le traitement de l'insertion,au passage des paramètres effectifs,mais on doit alors placer une fois pour toutes en tête du programme les adresses locales dans la configuration.

La suite de cet article montrera comment construire un "ordinateur formel" autour de ce processeur qu'est la "machine formelle".Là va se matérialiser la notion de système,que je compliquerai afin de lui faire rendre des services de plus en plus importants.

A suivre.

NOUVELLE STRUCTURE DE MEMOIRES ADRESSABLES PAR UNE FENETRE A DEPLACEMENT CONTINU

par

Jean-Philippe LEHMANN* et Patrick ISOARDI**

RÉSUMÉ. — Les programmeurs qui travaillent au niveau machine, sont confrontés en permanence aux difficultés entraînés par l'absence de souplesse que détermine la structure cloisonnée des mémoires actuelles.

Les auteurs proposent d'atteindre l'espace mémoire par le moyen d'une fenêtre qui peut être instantanément située, n'importe où, dans cet espace. Il ne s'agit plus d'adresser des cases, ni même des ensembles de cases mais de se localiser dans un espace sans aucune autre restriction que celle de la dimension de la fenêtre par laquelle l'espace peut être observé.

Cet article présente un procédé de structuration de blocs mémoires répondant à la question, ainsi que l'assemblage de tels blocs.

Actuellement, tous les blocs mémoires utilisés dans les techniques de l'informatique se présentent chacun comme un ensemble d'emplacements disjoints, adressables individuellement. L'information circule sur un bus dont la largeur est égale à celle de l'emplacement.

Un emplacement est une cellule mémoire élémentaire ou un groupement de telles cellules qui sont alors sélectionnées simultanément. Même dans ce dernier cas, il est impossible d'observer, de façon simultanée sur le bus, une information dont une partie proviendrait des dernières cellules mémoires d'un emplacement et l'autre partie des premières cellules de l'emplacement suivant. A fortiori, il est impossible d'opérer un déplacement continu (cellule après cellule) de tout l'espace mémoire, à travers le bus. La structure actuelle des blocs fait de ses emplacements des éléments complètement séparés les uns des autres.

De très nombreux procédés d'assemblage de blocs permettent de sélectionner simultanément un emplacement dans chaque bloc de l'assemblage. L'information ainsi regroupée, des dispositifs variés de masquage et de décalage d'une fenêtre autorisent celle-ci à chevaucher deux ou plusieurs emplacements. Si un tel regroupement peut se faire à partir de n'importe quel emplacement de l'espace mémoire, le déplacement continu de la fenêtre sur l'espace est alors possible. Mais quelle que

* Faculté des Sciences de Luminy, Département de Mathématique-Informatique, 70, route Léon-Lachamp, Case 901, 13288 Marseille Cedex 9.
** Université Louis-Pasteur, Centre de Calcul de l'Esplanade, 7, rue René-Descartes, 67084 Strasbourg Cedex.

Ce texte a été composé au Laboratoire de Typographie informatique de l'Université Louis-Pasteur au moyen du logiciel TEX.

soit la méthode utilisée, l'information doit être nécessairement traitée et répartie dans l'ensemble des blocs constitutifs de l'espace mémoire; en aucun cas elle ne peut être enregistrée dans plusieurs cellules consécutives d'un seul et unique bloc mémoires.

Visant à supprimer cette contrainte, on expose ici une nouvelle manière de structurer des blocs mémoires, puis d'assembler de tels blocs entre eux. Les blocs ainsi conçus comporteront un ensemble de cellules mémoires élémentaires. L'adressage permettra la sélection simultanée de plusieurs cellules successives à partir de n'importe quelle cellule du bloc. En reprenant les termes utilisés ci-dessus pour décrire l'état de la technique antérieure, ce procédé de structuration autorisera en particulier, le déplacement continu d'une fenêtre sur tout l'espace mémoire du bloc, avec chevauchement de ses emplacements.

Exposé de la méthode

1. Le problème. — On considère comme élément minimum une cellule mémoire à p bits. Lorsqu'elle est sélectionnée, elle délivre ou reçoit simultanément les p bits constitutifs de l'information élémentaire.

On suppose qu'on dispose de $2N$ éléments minimum. Le fait de choisir $2N$ de ces éléments va permettre d'indiquer :

- d'une part, comment se constitue un bloc de N cellules mémoires,
- d'autre part, comment de tels blocs sont assemblés les uns aux autres.

On note B et B^+ deux blocs consécutifs de N éléments minimum et

$$M_0, M_1, \ldots, M_{N-1} \ , \ M_0^+, M_1^+, \ldots, M_{N-1}^+$$

la suite des éléments minimum de l'espace mémoire ainsi constitué.

Si on considère qu'un emplacement du bloc est un groupement de k cellules consécutives, le bus données support de la circulation de l'information relative à l'ensemble des emplacements, est ainsi constitué : c'est un groupement de k canaux; chacun d'eux est un groupement de p canaux binaires. On nomme ces k canaux :

$$D_0, D_1, \ldots, D_{k-1}$$

Dans une architecture classique N est un multiple de k. Lorsqu'une adresse est donnée depuis l'extérieur, pour une lecture par exemple, il circule alors sur le bus données

$$D_0 D_1 \ldots D_{k-1}$$

une information émise par un groupement du type :

$$M_i M_{i+1} \ldots M_{i+k-1}.$$

Si on passe à l'adresse suivante, c'est alors l'information émise par le groupement :

$$M_{i+k}M_{i+k+1}\dots M_{i+2k-1}$$

qui circule dans les mêmes conditions. Et ainsi de suite.

Le problème qu'on pose est le suivant (il importe peu qu'il s'agisse de lecture ou d'écriture) : on suppose que l'adresse X corresponde à la circulation sur le bus données de l'information relative à

$$M_0M_1\dots M_{k-1}$$

et on veut que la succession des adresses à partir de X, mette immédiatement en relation le bus données

$$D_0D_1\dots D_{k-1}$$

avec les emplacements indiqués dans le tableau suivant :

ADRESSE	D_0	D_1	...	D_{k-2}	D_{k-1}
...			...		
X	M_0	M_1	...	M_{k-2}	M_{k-1}
$X+1$	M_1	M_2	...	M_{k-1}	M_k
$X+2$	M_2	M_3	...	M_k	M_{k+1}
...	...	...	...	...	...
$X+i$	M_i	M_{i+1}	...	M_{i+k-2}	M_{i+k-1}
...	...	...	...	...	...
$X+N-k$	M_{N-k}	M_{N-k+1}	...	M_{N-2}	M_{N-1}
$X+N-k+1$	M_{N-k+1}	M_{N-k+2}	...	M_{N-1}	M_0^+
$X+N-k+2$	M_{N-k+2}	M_{N-k+3}	...	M_0^+	M_1^+
...	...	...	...	...	...
$X+N-1$	M_{N-1}	M_0^+	...	M_{k-3}^+	M_{k-2}^+
$X+N$	M_0^+	M_1^+	...	M_{k-2}^+	M_{k-1}^+
et ainsi de suite dans tout l'espace mémoire					

TABLEAU T1

Le fonctionnement du bloc mémoires B à structurer se définit ainsi de façon précise. Il est séléctionné par $N+k-1$ adresses allant de $X-k+1$ à $X+N-1$.

- Chacune des $N-k+1$ adresses

$$X, X+1, \ldots, X+N-k$$

sélectionne simultanément k cellules mémoires consécutives du bloc B suivant la relation définie dans le tableau T1.

- Chacune des $k-1$ adresses

$$X+N-k+1, \ldots, X+N-1$$

met en relation le bus données avec un emplacement regroupant les dernières cellules du bloc B et les premières cellules du bloc suivant B^+. Ce qui est mis en évidence dans le tableau ci-dessus.

- De la même manière, chacune des $k-1$ adresses $X-k+1, \ldots, X-2, X-1$ sélectionnera simultanément k cellules mémoires prises parmi les premières du bloc B et les dernières du bloc précédent.

Remarque 1. — Si ce bloc est le dernier de tout l'espace mémoire, une adresse de la suite $X+N-k+1, \ldots, X+N-1$ sélectionnera uniquement les dernières cellules du bloc, sauf si l'espace mémoire est circulaire.

Remarque 2. — Lorsque le bloc B est le premier de l'espace mémoire ($X=0$), les adresses $X-k+1, \ldots, X-1$ n'ont pas d'existence physique. Conformément au contenu du tableau T1, la première adresse (qui est 0) sélectionne immédiatement l'emplacement $M_0 M_1 \ldots M_{N-1}$.

Ces remarques ne modifieront en rien le procédé général de structuration du bloc puisque les adresses lui sont données de l'extérieur. Un bloc ainsi structuré pourra donc être placé en n'importe quel endroit de tout l'espace mémoire.

2. La solution. — Un bloc mémoires répondant à la question va comprendre :

- N cellules mémoires $M_0, M_1, \ldots, M_{N-1}$ dont les fonctions décrites précédemment sont déjà matérialisées de diverses façons dans diverses technologies;
- un bus données qui correspond à la description précédente;
- des dispositifs supplémentaires dont on donnera, au fil de l'exposé, une description fonctionnelle, étant entendu qu'ils sont matérialisables. Les moyens pour les réaliser font partie de l'état de la technique. On indiquera à l'occasion un cablage possible.

Le procédé de structuration du bloc sera entièrement défini par le cablage interne de ces éléments. Nous l'exposons ci-dessous.

STRUCTURE DE MÉMOIRES

- En premier lieu, chaque élément minimum doit être relié aux k canaux du bus données et non pas seulement à l'un d'entre eux, comme cela se fait dans les architectures classiques. Ainsi, le bus données des cellules mémoires M_i sera connecté à $D_0, D_1, \ldots, D_{k-1}$ et ceci pour $0 \leq i \leq N-1$. Chacune de ces liaisons devra être sous le contrôle d'une commande dont la fonction est d'autoriser ou non le flux des informations le long de cette liaison. Si on note C_j^i la commande qui contrôle la liaison de la cellule mémoire M_i avec le canal D_j du bus données, il existe les nk commandes :

$$C_0^0, C_1^0, \ldots, C_{k-1}^0 \ , \ C_0^1, C_1^1, \ldots, C_{k-1}^1 \ , \ \ldots \ , \ C_0^{N-1}, C_1^{N-1}, \ldots, C_{k-1}^{N-1}.$$

Pour définir le sens de circulation des informations (lecture ou écriture), il suffit de combiner la commande $R/\bar{W}$ provenant de l'extérieur avec chaque C_j^i dans un dispositif R, selon des méthodes classiques et parfaitement connues. Un exemple en est donné dans la description détaillée d'un mode de réalisation ci-après. A noter que cette question ne se pose pas dans le cas d'une ROM.

- En second lieu, un système d'adressage AD comporte des entrées suffisantes pour recevoir depuis l'extérieur les informations représentatives des adresses qui sélectionnent le bloc. Il possède $N + k - 1$ commandes de sortie :

$$E_{N-k+1} \ , \ E_{N-k+2} \ , \ \ldots \ , \ E_{N-1} \ , \ S_0 \ , \ S_1 \ , \ \ldots \ , \ S_{N-k} \ , \ \ldots \ , \ S_{N-1},$$

qui seront respectivement actives pour les valeurs de l'adresse

$$X-k+1 \ , \ X-k+2 \ , \ \ldots \ , \ X-1 \ , \ X \ , \ X+1 \ , \ \ldots \ , \ X+N-k \ , \ \ldots \ , \ X+N-1.$$

Ces commandes vont contribuer à la mise en œuvre des fonctions logiques du cablage interne du bloc mémoires en question. On s'exprime ici dans le cadre de la logique positive.

- Si on note CS_1 la commande de sélection de la cellule mémoire M_i, le tableau T1 indique de toute évidence que CS_i devra être actif pour les k adresses

$$X+i-k+1, \ldots, X+i-1, X+i.$$

Les fonctions qui régissent les valeurs logiques de ces commandes s'écrivent simplement en remplaçant les adresses ci-dessus par les sorties correspondantes du système d'adressage AD. Elles sont ainsi définies :

$$\begin{aligned} CS_i &= S_{i-k+1} + S_{i-k+2} + \cdots + S_i \quad \text{pour} \quad k-1 \leq i \leq N-1, \\ CS_i &= E_{N-k+i+1} + \cdots + E_{N-1} + S_0 + S_1 + \cdots + S_i \\ &\qquad\qquad \text{pour} \quad i < k-1. \end{aligned} \tag{I}$$

- Le tableau T1 montre encore que C_j^i devra être actif pour l'adresse $X+i-j$. En procédant de la même façon que précédemment, les commandes de contrôle des liaisons cellule mémoire-bus données sont ainsi cablées :

$$\text{pour tout } i \leq N-1 \quad \begin{array}{ll} C_j^i = S_{i-j} & \text{pour } j \leq i, \\ C_j^i = E_{N+i-j} & \text{pour } j > i. \end{array} \qquad (II)$$

- Enfin, il reste à indiquer comment différents blocs structurés selon le procédé décrit ci-dessus peuvent être assemblés entre eux de manière à généraliser le problème sur un espace mémoire plus important. Là encore, c'est essentiellement l'adresse qui sélectionne le ou les blocs. Si B est le bloc en question, B^- son précédent et B^+ son suivant, il suffit d'assembler ces blocs mémoires en respectant la correspondance définie dans le tableau suivant :

Adresses	Blocs sélectionnés
...	B^-
$X-k$	B^-
$X-k+1$	B^- et B
...	...
$X-1$	B^- et B
X	B
...	...
$X+N-k$	B
$X+N-k+1$	B et B^+
...	...
$X+N-1$	B et B^+
$X+N$	B^+
...	...

TABLEAU T2

STRUCTURE DE MÉMOIRES

Exposé d'un mode de réalisation

Dans la solution générale du problème, on a dégagé les liaisons à réaliser et comment doivent être activées les commandes de sélection des mémoires ainsi que les commandes de contrôle des liaisons cellules-bus à partir des sorties du système d'adressage AD. Il reste à indiquer comment ses commandes de sortie de AD sont elles-mêmes activées à partir de l'adresse qui est fournie à l'entrée du système. Un tel cablage est fonction du codage des adresses.

Dans l'état actuel de la technique, N est une puissance de 2 et X est en général un multiple de N. Si on pose $N = 2^n$ et $X = uN$, une adresse s'écrit :

$$uN + i \qquad \text{pour} \qquad 0 \leq i \leq N - 1.$$

Le bloc est sélectionné par u et la mémoire dans le bloc par i. On code u et i en binaire sur deux canaux consécutifs qui constituent le bus adresses.

Si on conserve pour notre réalisation un tel système de codage, un bloc sera sélectionné pour des informations représentatives des adresses :

$$(u-1)N + N - k + 1\ ,\ \ldots\ ,\ (u-1)N + N - 1\ ,\ uN\ ,\ uN + 1\ ,\ \ldots\ ,\ uN + N - 1.$$

Les entrées du système d'adressage AD sont alors constitutées :

- d'un canal de n fils $A_0 A_1 \ldots A_{n-1}$, suffisant pour porter une valeur binaire allant de 0 à $N - 1$;
- d'une commande CS activée pour la valeur u;
- d'une commande ER activée pour la valeur $(u - 1)$ et l'une des valeurs

$$N - k + 1, N - k + 2, \ldots, N - 1$$

portée par le canal ci-dessus.

Si ADR désigne la valeur du canal $A_0 A_1 \ldots A_{n-1}$, les fonctions qui régissent les valeurs des commandes sortie de AD se formulent ainsi :

$$\begin{aligned} S_i &= CS.(ADR = i) \quad \text{pour} \quad 0 \leq i \leq N - 1, \\ E_i &= ER.(ADR = i) \quad \text{pour} \quad N - k + 1 \leq i \leq N - 1. \end{aligned} \qquad (III)$$

Le système d'adressage AD peut être réalisé par le cablage suivant :

- Le canal $A_0 A_1 \ldots A_{n-1}$ entre dans un décodeur D. Sa commande de sélection est CS. Ses sorties sont $S_0, S_1, \ldots, S_{N-1}$;
- On considère les t fils de poids faible de ce canal : $A_0, A_1, \ldots, A_{t-1}$, avec t défini par $2^{t-1} < k - 1 \leq 2^t$.

Ces fils entrent dans un second décodeur D' qui est sélectionné par ER. Ce sont les $k-1$ commandes de sorties de D', correspondant aux adresses les plus hautes que nous désignons, dans l'ordre décroissant de ces adresses : $E_{N-1}, \ldots, E_{N-k+1}$. Les autres sorties éventuelles qui correspondent aux adresses les plus basses sont inutilisées. Un tel dispositif est représenté dans le dessin 2.

Le dispositif R qui spécifie le sens du flux de l'information sur chaque liaison cellule-bus a ses commandes internes définies par les fonctions logiques suivantes :

$$C_L^{ij} = C_j^i . R/\bar{W} \qquad \text{et} \qquad C_E^{ij} = C_j^i . (\text{NON}(R/\bar{W})) \qquad (IV)$$

Ceci est mis en oeuvre de façon évidente par un jeu de portes **ET** et **NON** et de buffers 3-états comme l'indique le dessin 3.

Il reste à définir le moyen d'assembler plusieurs blocs entre eux. Là encore, on conserve le codage binaire actuellement employé. Dans ce cas, le tableau T2 montre qu'un bloc est sélectionné en même temps que le précédent, pour une valeur i comprise entre $N-k+1$ et $N-1$, circulant sur le canal $A_0 A_1 \ldots A_{n-1}$. Une telle valeur active la commande S_i du système d'adressage AD du bloc ainsi que la sortie E_i du système d'adressage du bloc suivant. Si le canal $A_0 A_1 \ldots A_{n-1}$ est commun aux deux blocs, pour respecter l'équation (III-2), il suffit que la commande ER d'un bloc soit activée par l'un des S_i du bloc précédent, pour $N-k+1 \leq i \leq N-1$. Cette dernière condition est cablée à l'intérieur du bloc par la présence d'une commande de sortie supplémentaire RE qui vérifie l'équation :

$$RE = S_{N-k+1} + S_{N-k+2} + \cdots + S_{N-1} \qquad (V)$$

Ce qui est mis en œuvre par une porte **OU** dans le dessin 1.

L'assemblage de deux blocs est alors réalisé très simplement, selon le procédé suivant :

- le bus données $D_0 D_1 \ldots D_{k-1}$ et le bus adresses $A_0 A_1 \ldots A_{n-1}$ sont communs aux deux blocs;
- la commande de sélection CS respecte les méthodes classiques d'assemblage; elle est active pour la valeur u;
- si RE^- est la commande de sortie du bloc précédent B^-, il suffit de réaliser le cablage :

$$ER = RE^- . \qquad (VI)$$

Si l'espace mémoire est linéaire,

- dans le cas où ce bloc est le premier, ER est à la masse,
- dans le cas où ce bloc est le dernier, RE n'est pas connecté.

Un exemple de ce procédé est présenté dans le dessin 4.

Le dessin 1 présente la structure générale d'un bloc mémoires réalisé selon la nouvelle méthode de structuration que nous venons d'exposer, dans le cas particulier où $p = 1$, $k = 3$, $N = 4$ donc $n = 2$ et $t = 1$.

Conclusion

Il devient possible de gérer aisément et de façon optimale une suite d'informations de longueurs variables inférieures ou égales à la largeur de la fenêtre, en réduisant au minimum le nombre des accès mémoire ainsi que le temps de ces accès.

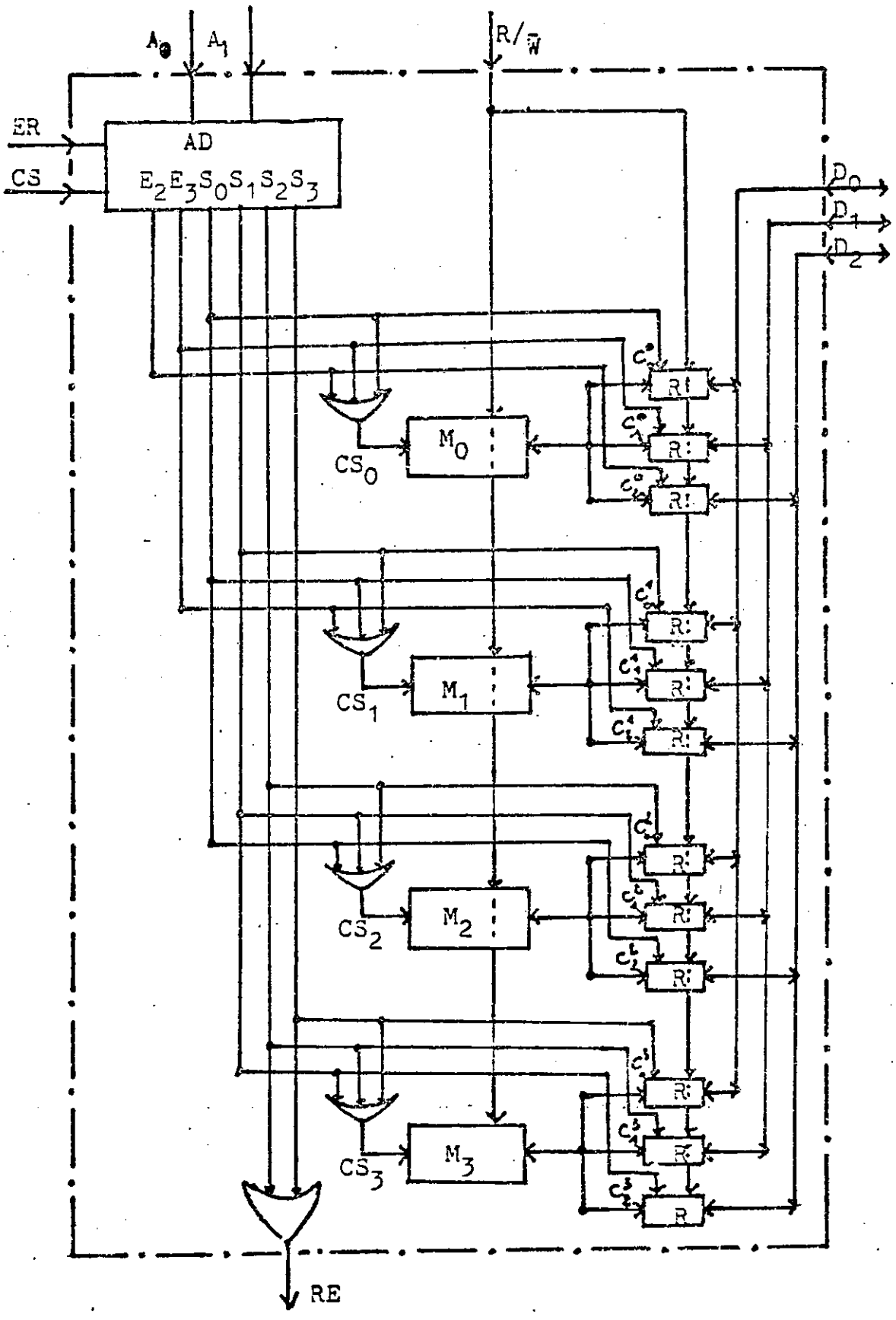
On sait qu'actuellement c'est à ce niveau que se rencontre une des plus grandes difficultés pour accroître la vitesse des ordinateurs. Il est raisonnable d'en déduire que la voie est ainsi ouverte aussi bien à l'intégration de nouveaux types de mémoires qu'à la conception d'architectures nouvelles de processeurs et de nouveaux langages machine, adaptés aux ressources ainsi offertes.

Bibliographie

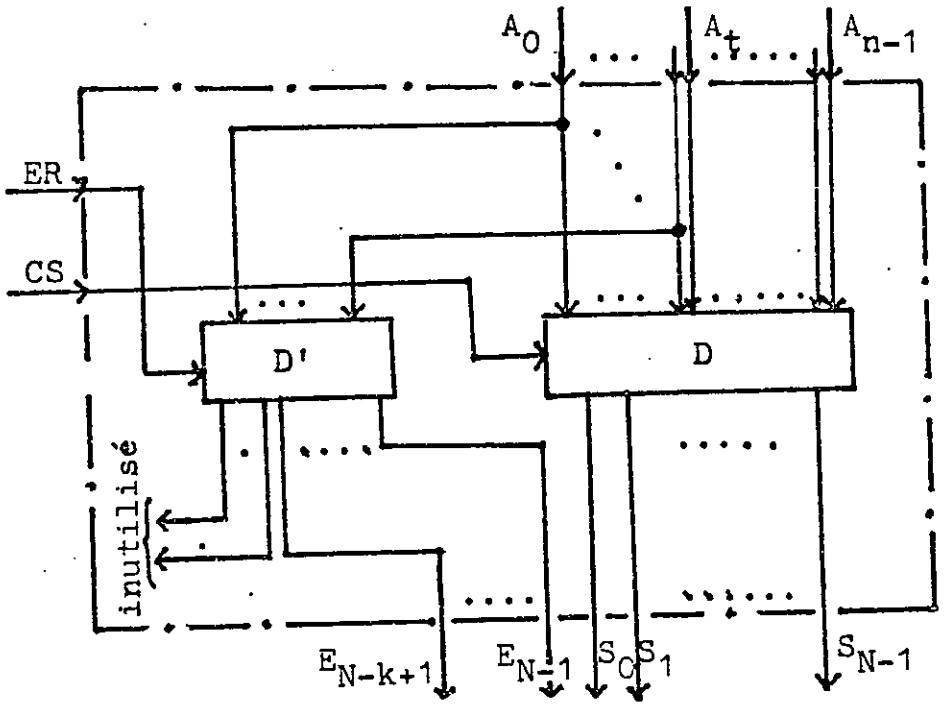
[1] GILOI (W.K.) and BERG (H.K.). — Data structure architectures. A major operational principle, *Proc. 5th Annual Symposium on computer architecture*, IEEE Publi. 78 CH 12849C.

[2] BACKUS (J.). — « Can programming be liberated from the von Neumann style?... », CACM, **21**, 1978.

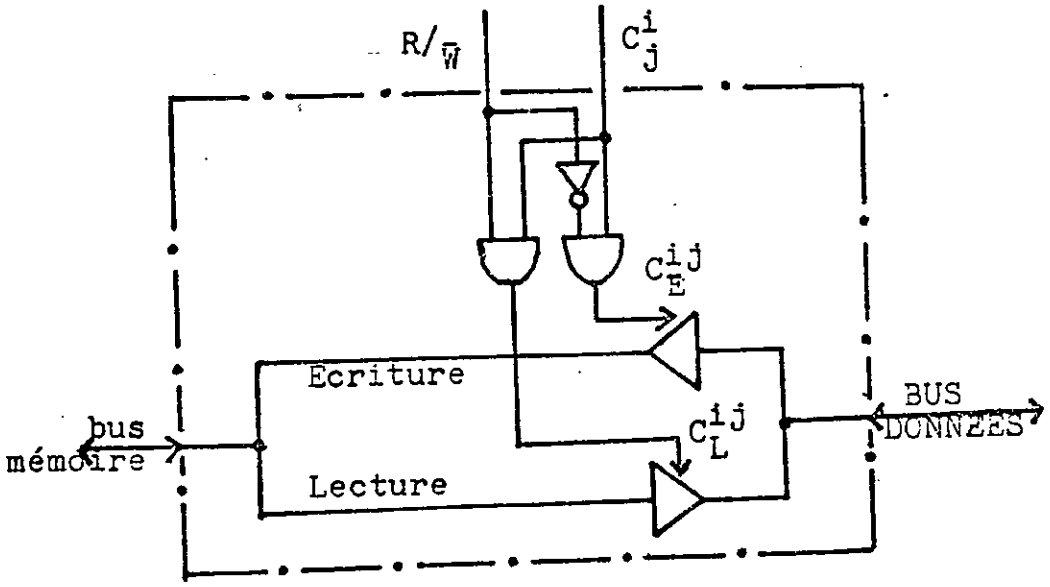
[3] LEHMANN (J.-P.). — Machines et calculabilité, *Thèse 3ème cycle, Luminy*, 1984.



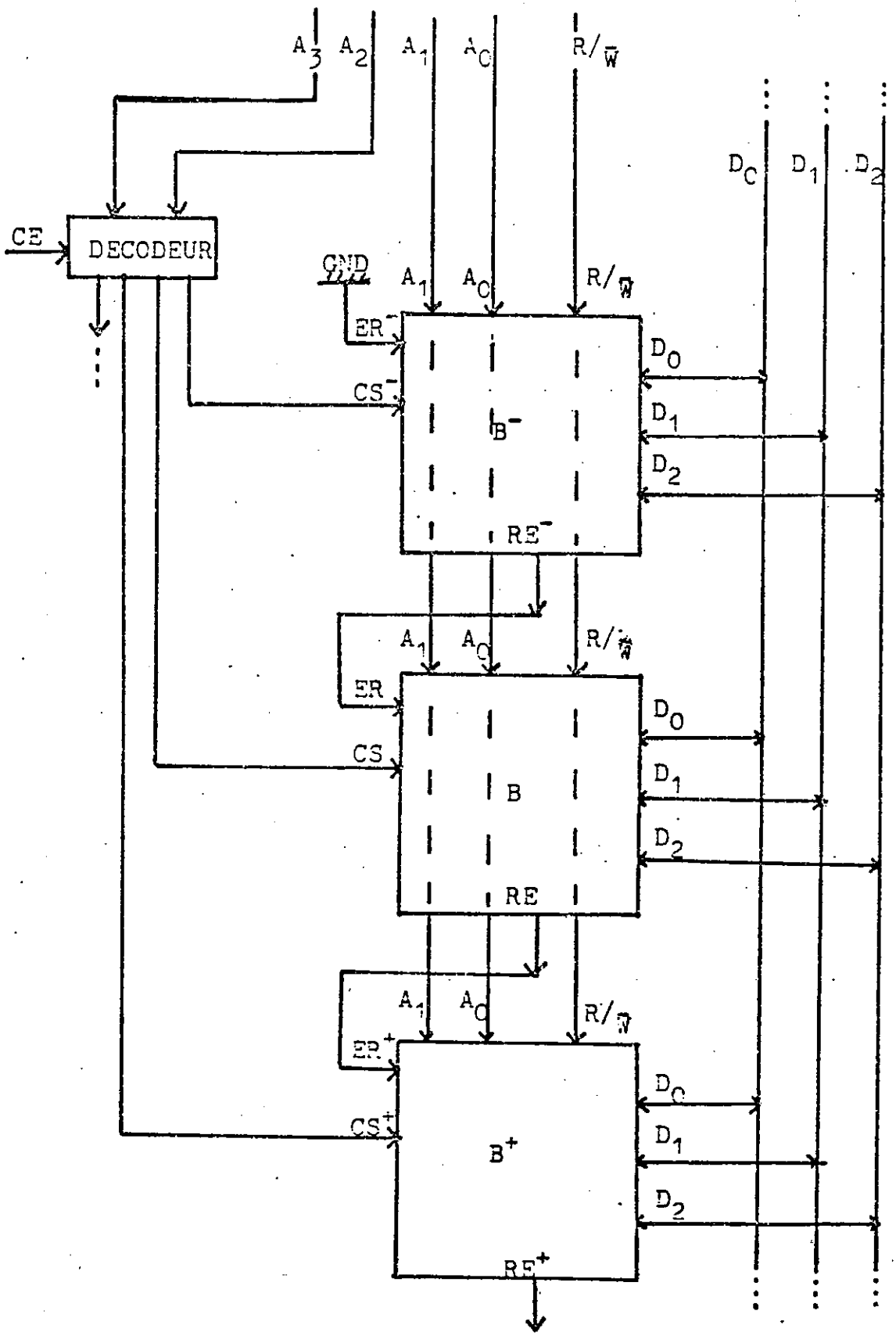
DESSIN 1 : Nouvelle structure de bloc mémoire



DESSIN 2 :Système d'adressage AD



DESSIN 3: Dispositif R contrôleur de liaison



DESSIN 4 : Assemblage de plusieurs blocs.

vouzzavedibisar

Devant le débordement de modernisme et de technicité,il est naturel de se demander si la vieux fonds de sagesse accumulé au cours des âges sous forme de contes,légendes,proverbes et fabliaux,ne se révèle pas inadapté devant notre agitation.Et cela au point qu'on ne puisse plus trouver de chutes qui aient un impact moderne.L'envie m'est donc venue de reprendre un vieux fabliau moyen-âgeux ,et de tenter la gageure d'une conclusion qui parlerait à nos consciences fatiguées.

Voici donc:

La légende de Petit-Lapin et de Petit-Renard.

Il était une fois Petit-Lapin qui se trouvait en difficulté dans l'immense forêt;Il était dangereusement pousuivi par des chasseurs obstinés.Hors d'haleine et profondément déprimé par la situation sans issue dans laquelle il se trouvait acculé: toutes ses ruses avaient été déjouées,il tombe sur Petit-Renard qui vaquait calmement à ses affaires.En quelques mots entrecoupés par une respiration difficile,Petit-Lapin désespéré,conte son aventure:

"Petit-Renard,Petit-Renard sauve-moi !"

Qu'à cela ne tienne lui dit Petit-Renard,monte sur ma queue et tiens toi bien ! sitôt dit,sitôt fait,Petit-Renard part d'un trait et Petit-Lapin est sauvé.

Un bienfait n'est jamais perdu.Le temps passa,beaucoup de pluie tomba sur la forêt,et un jour,les chasseurs pistèrent Petit-Renard,grand amateur devant l'eternel,de poulets élevés au grain.Et Petit-Renard hors d'haleine et les pattes douloureuses de n'avoir plus le moral au beau fixe tombe sur Petit-Lapin vaquant,tranquille,à ses affaires :

"Petit-Lapin,Petit-Lapin sauve-moi ! "
Petit-Lapin dit à Petit-Renard: suis-moi vite derrière ce gros buisson ! Et là une grosse automobile apparait aux yeux étonnés de Petit-Renard.Monte ! dit Petit-Lapin,et il démarre en trombe.Petit-Renard est sauvé.

Moralité: si vous avez une petite queue,mieux vaut que vous possédiez une grosse automobile.

J'ai donc essayé d'une chute plus moderne:
"Si vous avez peu d'idées ayez donc un gros ordinateur,au moins ça meublera votre salon Louis XV."

Mais vous voyez,ça tombe à plat,notre époque est une bien triste époque !

B