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

Sommaire

Mars 1984

ADRESSE POSTALE : FACULTÉ DES SCIENCES DE LUMINY
MATHÉMATIQUE-INFORMATIQUE - L I T A M - BAT. TPR 2 - 9ème ETAGE
CASE 901 - 70, ROUTE LÉON LACHAMP - 13288 MARSEILLE CEDEX 9 - ☎ (91) 41.01.40 POSTE 3294 - 3110

Editorial

e. bianco

J'aurais pu ...

J'aurais pu intituler cet éditorial: INFORMATIQUE ET POUVOIR.Auquel cas j'aurais eu le plaisir amusé de récuser une qualification de fétichiste en précisant que le mot "pouvoir" n'a jamais mis personne en prison,et que le mot "informatique" n'a pas de poches.Ces remarques sont peut-être cocasses bien que banales,mais elles donnent le ton.Et sur ma lancée j'aurais pu remarquer que bien avant l'apparition du mot magique: "informatique" ,l'art de manipuler l'information,devenu science depuis,s'accordait fort bien avec un autre art,celui de manipuler les foules qui,lui,curieusement ne s'est jamais mué en science.Il est resté un art qui a plutôt évolué à la façon de celui que pratiquent les Clowns.J'aurais adressé mille excuses aux clowns que j'admire beaucoup,en les priant de ne pas se vexer du rapprochement.En effet dans la logique d'un tel propos qu'un corps de métier se mette à en imiter un autre,et même s'il le fait de façon lamentable,ne permet pas de tirer de conséquences quant à celui qui se trouve imité.Reconnaissons-là une des propriétés,un des mystères de l'implication.

J'aurais pu également,montant d'un cran une certaine mauvaise foi,illustrer encore l'implication en disant des choses en apparence simples,mais en réalité chargées de venin.J'aurais pu

dire par exemple et incidemment,en comparant esprit naîf du chercheur - chacun sait qu'un bon chercheur est un naîf - et esprit retors du politicien - chacun sait qu'un bon politicien a l'esprit retors,j'aurais pu dire donc qu'un esprit retors,lui,voit venir les choses de loin,car il regarde sous la table,il détecte les connivences,il entre dans le jeu,il sait ce qu'il faut faire,il le fait,et se retrouve un jour ministre.Et là,tout heureux il ne me restait plus qu'à remarquer finement que cela n'implique pas forcément,même si c'est vrai,qu'un ministre a l'esprit retors.Un petit exemple historique à l'appui et le tour est joué: cela crève les yeux que tous les ministres ne sont pas des Talleyrand.

Pourquoi s'en prendre ainsi aux ministres ? C'est grand,un ministre,certes.Mais c'est faible également.La grandeur n'exige pas des valises de diplômes - et c'est fort heureux,mais la faiblesse provient surtout de ce que nous exigeons de lui.Et nous exigeons du ministre tout ce que nous ne pouvons plus faire ou tout ce que nous ne voulons pas faire.Pour gagner nos grâces il est bien obligé de nous faire des promesses,ce pauvre ministre.Et comment voulez-vous qu'il les tienne,toutes ces promesses,les plus implicites,celles que chacun de nous a ressenties parcequ'il nous a dit des paroles chaudes,notre bon ministre,pour nous réconforter.

Alors j'aurais pu parler de la naîveté du bon chercheur,la souligner perfidement pour établir un contraste avec la vilenie du mauvais chercheur qui fait semblant,mais de fait recherche un certain pouvoir en se consacrant à de l'organisation,chacun sait cela mais je l'aurais souligné,car cela mène à arpenter

les trottoirs parisiens où se glane la bonne aubaine.

J'aurais alors poussé le lecteur à faire la remarque dès lors évidente que ce mauvais chercheur profite,abuse d'une légère tendance un peu française,qui veut qu'on ne donne bien qu'aux gens qu'on connaît bien,plus exactement qu'on rencontre souvent.J'aurais pu alors ajouter de plus en plus perfidement que beaucoup de français ont un coeur de Président,et que,quant on ne peut être celui de la république,on peut toujours essayer de devenir celui de la commission qui va répartir les crédits, surtout quant un gros nuage lourd de promesses surgit à l'horizon.

J'aurais dit tout cela en masquant délibérément que le bon chercheur,comme le bon Samaritain n'éprouverait aucun plaisir à faire ces choses,et s'il ne les fait pas ce n'est pas l'expression d'un bon naturel mais celle de l'égoïsme le plus plat. Et d'un autre côté,peut-être ne serait-il pas capable d'éffectuer un tel travail qui n'est pas au fond dénué d'intérêt,disent certains,ni tellement facile; Et alors pourquoi se plaindrait-il ?

J'aurais pu,sur le même ton,m'étonner de certaines réponses de l'administration à des sollicitations de chercheurs,et citer bêtement tel commentaire:

"Le moins pour sortir de son isolement est de consentir à utiliser le langage de la communauté et de situer ses travaux par rapport à d'autres.Tant que ces deux conditions ne sont pas remplies,la commission ne pourra pas prendre en considération la demande d'association." 21/11/83

Et j'aurais eu beau jeu d'ironiser platement en exhibant quelques exemples historiques choisis:

Il suffisait d'imaginer que le berger Kepler ait eu,un jour, l'envie de faire une demande d'association au CNRS,voila donc ce qui aurait pu lui être répondu.Peut-être en faisant un petit effort aurait-il pu situer ses travaux par rapport à ceux de Ptolémée,voire ceux de Newton.Et puis,peut-être préférait-il ses cieux calmes et étoilés,et ne désirait-il pas sortir de son isolement ? Allez savoir.

J'aurais pu encor me gausser davantage en imaginant Galileo Galilei s'adressant à la "communauté" en parlant son langage.

Faisant cela j'aurais été conscient d'entrevoir les confins du mauvais goût et de la mauvaise foi.

Cédant à une facilité toujours plus outrancière,et visant plus haut j'aurais pu par le détour d'une parodie de Ron Cobb juxtaposer la sérénité d'une déclaration ministérielle avec la rugosité de la simple réalité.Comment résister alors au plaisir de citer la parole du ministre.

Dans les 10 mesures que prévoit Monsieur le Ministre de l' Industrie pour relancer la recherche il est dit:

"Il existe environ 150 écoles d'ingénieurs en france;or la majorité ne font pas de recherche et lorsqu'elle existe,cette recherche est souvent insuffisante en volume et en qualité. Une convention entre le Ministre de l'Industrie et de laRecherche et le Ministre de l'Education Nationale aura pour objet de développer l'éffort de recherche entrepris dans les écoles qui en font déja,mais aussi de <u>créer des unités de recherche dans les écoles d'ingénieurs</u> qui n'en font pas encore."

Voila donc un ministre qui agit.Que pourraient dire après

cela les dénigreurs systématiques ? La reconnaissance n'est pas de ce monde,il se dirait encore qu'au fond,il faut bien dix ans pour qu'un laboratoire commence à produire,et puis que peut-être il existe déja des laboratoires qui produisent et pourraient facilement faire mieux,bien aidés,peut-être même il y en a-t-il qui travaillent déja en collaboration avec l'industrie et ne peuvent grand-chose par manque de moyens,et peut-être aurait-il été bon de les aider un peu en attendant que les autres soient opérationnels,et peut-être que...et patati et patata.

Qui n'a les oreilles pleines de ces longues déclamations sur les retards accumulés par des majorités plus ou moins précédentes,sur les nécéssités plus qu'urgentes de les combler,sur la nécéssité d'exciter l'innovation et l'esprit d'entreprise.

Quelque petit troll narquois aurait pu me souffler des remarques du genre: tiens ? recherche ? innovation ? quel est ce glissement.Ou bien même aurais-je pu citer ce mot d'Aigrain:

"Les consultants doivent remplir tous les trois mois des formulaires extrêmement complèxes pour la TVA et cela les enquiquine !!! "

Cité dans Formation par la Recherche N°6 de mars 1984.

N'importe quel vil esprit aurait pu ricaner que la recherche fondamentale est tellement hors de notre portée qu'on va forcer sur l'innovation dans la technique de l'ouvre-boite-à-conserve-pour-exportation.Un domaine toutefois où notre maîtrise demeure totale: la conception du formulaire de TVA,qui parvient,réussite sublime,à dépasser les capacités intellectuelles

de nos savants.

Dans le même ordre d'idées,le Président de la République constate que les entreprises sont écrasées par les charges administratives -oui- et il déclare qu'il faut absolument réduire ces dernières.Aussitôt les mêmes mauvais esprits de prédire un déluge de paperasses supplémentaires; Voila de quoi rassurer l'esprit d'entreprise.

Tout cela j'aurais pu le dire et je m'en suis bien gardé car il ne s'agit là que de propos désabusés,qu'il est bon de désarmer par le mépris et qui ne pourraient que stigmatiser ceux qui les tiennent.

Quant au fond véritable de ma pensée il est tout dénué de complications: le spectacle de la Nature me ravit,et l'approche pourtant tardive du printemps recouvre tout de vert tendre,de fleurs et de jeunes filles en fleur,cette ambiance ne peut inspirer que des pensées empreintes de sérénité.

C'est en toute sérénité que m'apparaît la situation,et c'est en toute sérénité que je peux la résumer.

Il existe plus qu'on ne croit de gens qui aiment à faire un travail sérieux.Qu'être sérieux est incompatible ni avec la gaité ni avec le rire et la drôlerie.Mais il se trouve que ces gens-là apprécient souvent,pour ne pas dire toujours,un minimum de confiance.Ils n'aiment pas forcément dire les choses comme tout le monde,ou plutôt comme n'importe qui,ce qui signifie que les jargons leur cassent les pieds.

Vous voulez de l'innovation ? laissez réfléchir ceux qui ont envie de le faire,sans oublier pour autant qu'on réfléchit sou-

vent bien mieux à plusieurs que tout seul.Encore un petit détail,on ne réfléchit bien que dans un milieu de gens avec qui l'on s'entend bien.Cela dépasse le simple plan technique: un laboratoire qui marche bien n'est pas un amoncellement disparâte de gens récupérés au petit bonheur pour ne pas laisser perdre les postes qui passent - surtout si ces postes sont rares.

On pourrait se demander s'il n'en est pas de la recherche comme de la classe ouvrière,on l'évoque beaucoup mais on ne sait plus trop où ça se trouve.Construisons une métaphore hors du temps,telle que toute coïncidence avec des faits réels ne pourrait être que fortuite.Imaginons que la france soit un peu à la traîne du côté de l'informatique et décide de frapper un grand coup: on va en même temps lancer un modèle NF d'ordinateur et on va un peu forcer la main à des gens dont le métier est d'enseigner.Imaginons,toujours hypothèse gratuite,une économie sur le salaire du programmeur qui fait le logiciel,et sur celui de l'ingénieur qui rédige la documentation technique,on peut s'attendre à ce que le logiciel ne soit pas merveilleux et la notice technique pas tout à fait commode ni avenante.Peut-on essayer de comparer le gain dû à l'économie,et la perte dûe au gaspillage de temps et de matériel qui va en découler pour chaque utilisateur.On répond à cela que ceux qui gagnent dans cette opération ne sont pas les mêmes que ceux qui perdent.Oui mais il est possible également que le client potentiel se considérant pris pour ce qu'il estime ne pas être,se mette à prendre la marque NF pour ce qu'elle est vraiment.

Si telle était la situation,mais bien sûr cela ne saurait être le cas,cela voudrait dire qu'On a,entre autres,pas encore compris que le logiciel coûte très cher et qu'une économie sur

le logiciel,infiniment plus cher encore.

Une comparaison osée avec les antiquités et la brocante surgit d'elle-même: plus c'est vieux et plus ça ne tient qu'avec des bouts de ficelle,plus ça revient à des fortunes.

Je cherche visiblement à me faire peur,et je vais encore plus loin dans la recherche du frisson,avec un exemple encore plus impossible.Imaginons notre recherche guidée par incitations.Il y a eu l'époque compilation,c'est déja la préhistoire,puis l'époque système,c'est le moyen-âge,aujourd'hui on ne pousse plus que ceux qui font dans le capteur d'information,le tout dilué dans une filière électronique annoncée à grands renforts de tambours et qui s'éffiloche dans l'espace comme un cumulus de beau temps en fin d'après-midi.Comment feraient ceux qui voudraient encore approfondir la notion de système si l'on imaginait qu'il puisse exister des utilisateurs mécontents de ceux qui existent ? et de leurs compilateurs ?

Ne pénétrons pas trop loin dans l'horreur.Ces questions-là sont définitivement réglées,On le sait bien: tout les utilisateurs sont contents des systèmes qui gèrent leurs ordinateurs et leurs compilateurs sont des mécaniques parfaites,et quand il s'agit de rassembler en un seul des programmes venus d'horizons différents,on a affaire à un problème qui n'en est plus un.

Je n'évoquerai pas l'avance technologique de la france en matière de conception de processeurs et de hardware,car elle fait pâlir d'envie americains et japonais réunis,donc là encore pas besoin d'incitation.

Il n'est vraiment pas possible de se laisser aller à toutes ces opinions de mauvais aloi.Au contraire,je suis bien convaincu qu'il faut trouver à ce conglomérat de faits la véritable explication capable de montrer à quel point tout ceci est cohérent et bénéfique.

D'abord: la Pureté,il faut à tout prix que les têtes pensantes évitent de venir se rendre compte d'elles-même sur le chantier,cela leur gâterait les idées qu'ils doivent conserver vierges de toute atteinte pharisienne.En effet,comment expliquer ce soucis apparent de tout démolir qui commence un peu à fonctionner,pourquoi ce soucis permanent de financer au maximum les entreprises les moins rentables,pourquoi la régression de la recherche fondamentale au profit du petit moulin qui libère tout le monde.Pourquoi,en même temps cette vague de mise en repos.Le chômage en expansion montre qu'il est un repos disponible pour autre chose que la recherche vaine du travail.En même temps que s'amorce la réduction d'activités des irréductibles.

La leçon devient évidente: le bond en avant,le sacrifice de la production au bénéfice de l'accroissement de production,le surarmement pour se sursécuriser,la surenchère à l'innovation pour surcompenser les importations,bref le sacrifice du présent pour le futur entraine de curieuses conséquences:

La surproduction crée des accroissements de charge qui exigent de nouvelles ressources lesquelles puisent dans une surproduction de la surproduction,et à force de surproduction,les cours s'effondrent - sans parler de la qualité.De son côté le surarmement force à la recherche d'une clientèle: que faire

des vieux stocks que les nouveaux modèles rendent caducs ? En même temps,vendre le dernier cri de la technique,outre la rentabilisation,permet de disposer d'un terrain d'expériences sans danger et peu coûteux.Par contre,le client ne tarde pas à être mieux armé que le marchand.

Face à cela on nous propose une réduction d'horaires,pendant que les esprits subalternes toujours à l'écoute de la moindre injonction organisent la présence systématique aux trente-neuf heures,ils pénalisent ainsi ceux qui travaillent bien plus,même si ce n'est pas sur le lieu du travail.Et c'est là que l'esprit médiocre,à son insu devient l'instrument de la grande idée car il ne se préoccupe que de détecter le fraudeur. L'être honnête ne pose pas problème et l'esprit sagace adore résoudre les problèmes.Cette coexistence obligatoire entre gens de nature si opposée aura au moins pour conséquence d'aboutir à la substitution d'une inactivité fébrile à une activité relative.

Voila la grande vérité: il faut freiner l'emballement de notre société technologique sans bloquer pour autant la suractivité.Pour cela il suffit d'organiser la neutralisation réciproque des éléments actifs.La véritable réflèxion pourra s'exercer dans un environnement d'actions raisonnables et mûrement pesées,lors d'un retour à une vie plus calme,lorsque le temps aura usé toutes les forces antagonistes.

Ce message aucun ministre ne peut l'émettre directement en sa vérité simple et nue,alors sont mis en place d'énormes efforts allusifs.Pour ma part,j'ai compris,et je m'en vais de ce pas,préparer d'excellentes vacances.

UTILISATION D'UN MICROPROCESSEUR SPECIALISE DANS LE CALCUL, LE MM 57109 (N.S.) , COMME PERIPHERIQUE D'UN MICROPROCESSEUR 16 BITS TMS 9980 (T.I.)

par Jean-Luc PAILLET

Résumé : Nous traitons ici d'une carte mise au point pour utiliser le microprocesseur spécialisé-calcul MM 57109 en processeur "esclave" d'un système à TMS 9980, ainsi que du programme nécessaire pour gérer la communication entre les deux microprocesseurs .

C.R. Subjects Classification Informatics B.7.1 C.1.2

UTILISATION D'UN MICROPROCESSEUR SPECIALISE DANS LE CALCUL , MM 57109 (N.S.) , COMME PERIPHERIQUE D'UN MICROPROCESSEUR 16 BITS TMS 9980 (T.I.)

Dans certaines applications de microprocesseur, comportant la nécessité de calculs complexes et précis, une solution intéressante est de confier les tâches de calcul à un microprocesseur spécialisé, associé au microprocesseur "principal" et "activé" par celui-ci. En effet, ceci évite la mise au point de sous-programmes de calcul compliqués, longs, et occupant beaucoup de place en mémoire.

Nous avons choisi pour étudier et réaliserune telle association :

* comme processeur de calcul : un microprocesseur "nomber-oriented" de National-Semiconducteur, le MM 57109, relativement bon marché, consommant peu (15 mA sous 9v), qui s'est révélé aisé à interfacer, et comportant des possibilités mathématiques intéressantes .

* comme processeur central : le TMS 9980 de Texas-Instrument, version économique (boitier 40 broches) à bus de données multiplexé 8 bits, du 9900 , dont il possède le même jeu d'instructions 16 bits, l'architecture interne 16 bits et la même organisation logique . Ses particularités nous ont permis d'obtenir un programme de communication assez court et d'insertion aisée dans un programme demandeur de calcul .

Nous disposions, pour faire cette étude, de la carte d'évaluation TM 990/189 (Carte "Université") du 9980. Elle comporte, en particulier, un port parallèle d'Entrée-Sortie TMS 9901 disponible pour les applications, et permet l'utilisation d'un Assembleur acceptant les références symboliques .

I. PARTICULARITES ESSENTIELLES DES PROCESSEURS CONCERNES .

1- La famille 9900 se caractérise essentiellement par les propriétés suivantes :

* Architecture "mémoire à mémoire" : accumulateurs banalisés

en Mémoire Centrale .

Possibilité, en n'importe quel endroit d'un programme, de pointer n'importe où en mémoire centrale un banc de 16 registres formels (de 16 bits chacun), permettant de définir un contexte de travail. Ces registres sont orthogonaux pour les différents modes d'adressage .

Par ailleurs, on peut "vectoriser" un changement de contexte provoqué par soft, avec passage automatique de paramètres entre les contextes (cf. plus loin : Insertion du programme de gestion calcul dans un programme appelant).

* Interruptions masquables hiérarchisées et vectorisées, avec préservation automatique du contexte interrompu (pointeur d' "espace de travail", registre d'état, adresse de retour) .

* Possibilité de communication Entrée/Sortie par voie réservée, au moyen d'un "registre de communication série" (le C.R.U.), avec adressage indexé du périphérique par un des registres formels (R 12) . On peut ainsi, par exemple, associer à des contextes différents prévus dans un programme global, des adresses périphériques différentes ; celles-ci sont affectées automatiquement lors des changements de contexte .

Plus précisément, soit un module M de programme, auquel est associé un vecteur d'interruption ou de "XOP" (instruction de changement de contexte, vectorisé, avec retour) ; dans M , le registre R 12 corres--pond à l'adresse A = WP + 2 * 12 (où WP = valeur du pointeur d'espace de travail, défini par le vecteur). Lors d'une instruction d'Entrée/ Sortie exécutée dans M , l'adresse-référence de base (CRUBASE) du péri--phérique est celle contenue en A . Pour un autre module, avec un autre vecteur, la CRUBASE peut être contenue en une autre adresse et donc être différente. Ces différentes CRUBASES peuvent être définies préala--blement, dans le début du programme global .

* L'interface programmable TMS 9901 sert de port parallèle adapté à la communication C.R.U.. Le 9901 sert aussi à gérer les demandes d'interruption: entrées masquables localement et individuelle--ment, et à priorités différentes ; émission d'un code de priorité . Il contient de plus un timer programmable .

2- Caractéristiques essentielles du MM57109 :

* Instructions de calcul des principales fonctions usuelles : + , * , 1/X , $\sqrt{X}$, X^2, fonctions trigonométriques et inverses, Y^X ,

e^X, log , Log , ...

Les données peuvent être entrées , digit par digit, indicateur d'exposant et signes, sous forme d'instructions "entrée de digit" .

* Format des données : nombre fractionnaire décimal, ou écriture à exposant et mantisse de longueur programmable (maximum = 8 chiffres pour la mantisse) ; exposant compris entre -99 et 99 .

* Manipulations de 5 registres de mantisse organisés en pile, plus une mémoire annexe de même format .

* Instructions de test, avec émission d'une impulsion sur une broche spéciale ($\overline{BR}$) si une certaine condition est vérifiée , ou par instruction inconditionnelle . Ces instructions comportent un 2^e mot, non signifiant pour le MM 57109 , mais utilisable pour définir une adresse de branchement dans le programme de calcul .

II. MODE DE COMMUNICATION UTILISE .

Nous faisons communiquer les deux processeurs par l'intermédiaire du port 9901 "utilisateur" de la carte TM 990/189 ;

Le programme de calcul, en codes MM 57109 est en mémoire centrale . Le programme de gestion calcul (GC) communique ce programme de calcul, instruction par instruction, au MM 57109, et récupère le résultat final .

Il y a donc deux phases , vues de l'Unité centrale 9980 :

- Emission d'instructions calcul à destination du MM 57109. Chaque instruction calcul est présentée directement au MM 57109 pour être exécutée immédiatement ; l'émission doit être synchronisée par le signal RDY (Ready) du MM 57109 .

En effet, le code instruction doit être présenté aux entrées "instructions" pendant que RDY = 1 et doit rester stable pendant l'exécution (tant que RDY = 0) . Pour cela, RDY est dérivé, inversé et appliqué à l'entrée $\overline{INT}$ 5 du 9901 , ce qui provoque une demande d'interruption (codée niveau 3 pour le 9980) ; le programme d'interruption associé transmet l'instruction calcul convenable au MM 57109 . RDY = 1 pendant 8 microcycles MM 57109 .

Pour réaliser des branchements inconditionnels ou conditionnels sur instruction test MM 57109 dans le programme de calcul , le signal $\overline{BR}$ est appliqué à l'entrée $\overline{INT}$ 4 . Une impulsion éventuelle de ce signal, avec front descendant avant le front montant du RDY qui suit l'exécution de l'instruction en cours, provoque une demande d'interruption (codée niveau 2 pour le 9980) prioritaire par rapport à une demande d'origine

RDY . Le programme associé calcule l'adresse de saut dans le programme de calcul et positionne le pointeur utilisé par le programme de gestion calcul, d'aprés la valeur relative (sur un octet) indiquée dans le deuxième mot du code ayant activé $\overline{BR}$. Une impulsion $\overline{BR}$ dure 4 microcycles .

- Réception du résultat final du calcul .

Il faut au préalable émettre l'instruction OUT , qui met le registre X (du MM 57109) en sortie .

Les chiffres apparaissent un par un en BCD sur les lignes DO 1 à DO 4 , dans un ordre bien défini, ainsi qu'un code signifiant les signes de mantisse et d'exposant, et un code pour la position du point décimal. En même temps apparaissent, sur les lignes DA 1 à DA 4, les "numéros de sortie" de ces chiffres .

Ceci est synchronisé par le signal $R/\overline{W}$ du MM 57109 , à raison d'une impulsion basse par couple chiffre-numéro émis .

Ce signal $R/\overline{W}$ est appliqué à l'entrée $\overline{INT}$ 3 , provoquant une damande d'interruption (codée niveau 1 pour le 9980) . Le programme associé lit et stocke chaque couple chiffre-numéro en mémoire centrale .

Une impulsion $R/\overline{W}$ dure 2 microcycles .

III. REALISATION DE LA CARTE PROCESSEUR CALCUL MM 57109

Le schéma de principe (fig. 1) résume les connections entre le MM 57109 et le port E/S TMS 9901 de la carte "Université" .

* Les entrées $\overline{INT}$ 7 à $\overline{INT}$ 14 du 9901 sont utilisées pour recueillir, en lecture, les couples chiffre-numéro de la séquence de sortie des résultats. Ces entrées sont évidemment masquées pour les interruptions .

* Les sorties P0 à P6 du 9901 transmettent les codes instruc--tions calcul au MM 57109 .

* Les entrées $\overline{INT}$ 3 , $\overline{INT}$ 4 et $\overline{INT}$ 5 reçoivent, en mode interruption les signaux de contrôle de séquencement .

Transcodage.

La liste d'instructions du MM 57109 est donnée en Octal, et ces instructions doivent être écrites en mémoire centrale avec des chiffres hexadécimaux. Cependant, il est agréable de conserver dans l'écriture hexadécimale les chiffres donnés en Octal , et d'utiliser, par contre, des chiffres décimaux pour l'écriture des données (instructions "entrée

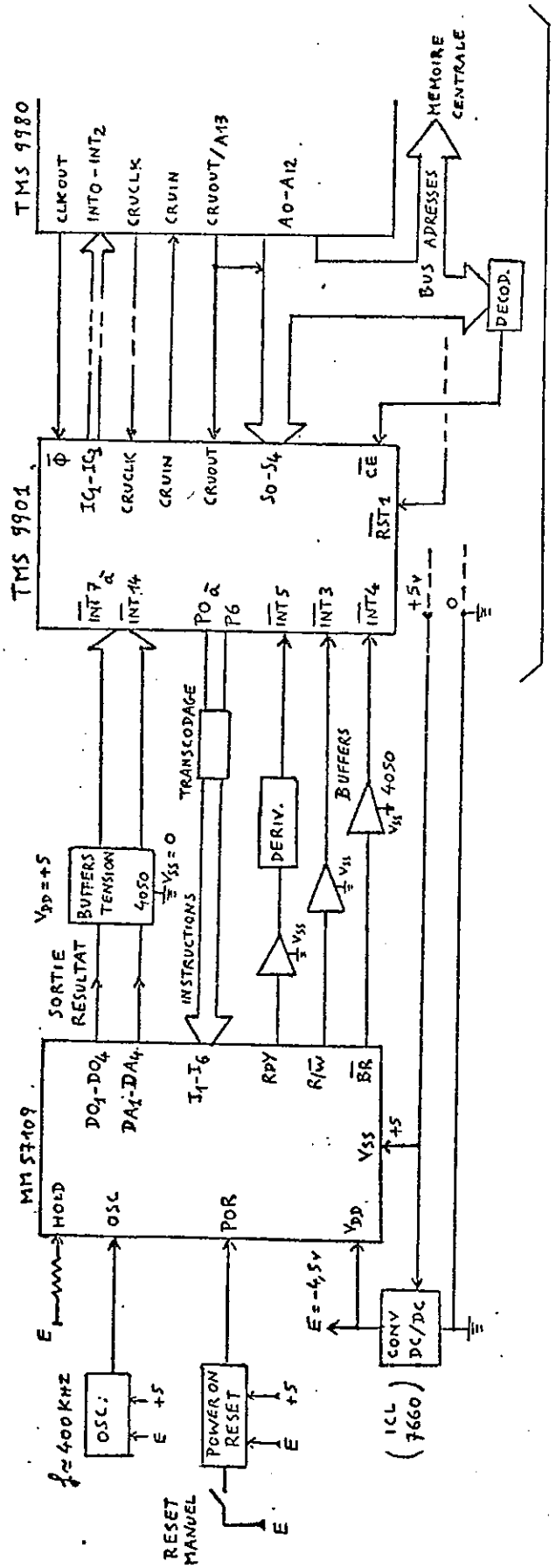


FIG.1 SCHEMA DE PRINCIPE

digit"). Pour cela, les lignes P0 à P6 sont connectées aux entrées I1 à I6 du MM 57109 de la manière suivante :
P0 sur I1, P1 sur I2, P2 sur I3 , le "OU" logique de P3 et P4 sur I4, P5 sur I5, P6 sur I6 .

Dérivation du signal RDY .

Elle est réalisée par un réseau R - C , associé à un transistor (il n'y avait plus de place pour un boitier circuit intégré !) .

Compatibilité électrique .

Le MM 57109 nécessite une tension d'alimentation $V_{SS} - V_{DD}$ de 8 à 9,5 v . Cependant, avec V_{SS} au +5v (de la carte TM 990/189) et V_{DD} à une tension E = - 4,5 v , les entrées I1,..., I6 sont TTL-compatibles: elles acceptent 1v (max.) comme niveau logique 0 . Les niveaux logiques des sorties P0,..., P6 du 9901 sont donc convenables en tension pour I1,..., I6 .

Par ailleurs, la sortance de P0,..., P6 est suffisante pour driver directement, sans buffer, les lignes I1,..., I6 . Notons que, d'aprés la notice, I6 doit être tiré au + 5v par une résistance pull-up de 5 à 20 kΩ .

Notons aussi que I1,..., I6 sont, en l'abscence de connection, à l'état haut, ce qui correspond au code d'instruction NOP ; si par ailleurs, le 9901 est configuré avec toutes ses broches E/S en entrées, celles-ci sont au niveau haut, ce qui a pour effet d'appliquer un NOP au MM 57109 .

On suppose que le 9901 a cette configuration initiale, avant d'entrer dans le programme de gestion calcul ; on peut l'obtenir par un $\overline{\text{RST}}$ 2 .

Les sorties du MM 57109 sont actives au niveau haut, et sinon sont en haute impédance . On les adapte aux entrées du 9901 par des buffers non inverseurs, en technologie C-MOS, avec une tension d'alimen-tation 0v et 5v et avec l'entrée tirée au 0v par une résistance pull-down de 15 kΩ . Nous utilisons pour cela des circuits 4050 .

Un certain nombre de sorties du MM 57109 ne figurent pas sur le schéma de principe, mais peuvent être utilisées pour des contrôles : flags F1 et F2 , visualisés par des LED's ; flag ERROR, bufférisé sur carte et disponible pour être connecté au 9901 . Par ailleurs, la sortie $R/\overline{W}$ (impulsions trés courtes) est fortement amplifiée en courant pour allumer une LED (par décharges d'un condensateur, pour augmenter l'efficacité lumineuse des impulsions) : céci est utilisé pour contrôle,

en particulier dans l'utilisation d'une carte d'affichage direct des résultats, avec instruction OUT (MM 57109) maintenue .

La tension négative E est générée sur carte au moyen d'un circuit intégré convertisseur, sans utilisation de bobinage, le circuit "miroir" ICL 7660 (Intersil). C'est suffisant pour le courant consommé par l'ensemble de la carte sur la tension E : 15 mA environ . Le convertis--seur consomme à peu prés 30 mA .

Horloge .

Le processeur de calcul a besoin d'une fréquence d'horloge d'environ 400 kHz . Nous la générons par un oscillateur à portes inver--seuses et R - C .

Cette fréquence est divisée par 4 par le processeur ; la sortie SYNC émet une impulsion par microcycle , soit toutes les 10 µs .

IV. LE PROGRAMME DE COMMUNICATION ENTRE LES PROCESSEURS ; INSERTION DANS UN PROGRAMME APPELANT .

C'est le rôle du programme nommé "Gestion calcul" (GC), de procéder à la communication entre les processeurs (cf. II) .

Nous donnons l'organigramme de principe en fig. 2 , ainsi qu'un listing (à la main!) en assembleur et en codes hexadécimaux , dans l'Annexe 3 .

Un programme demandeur de calcul doit faire appel à GC , et lui communiquer l'adresse du programme de calcul (liste de codes MM 57109) et l'adresse où stocker le résultat .

La manière la plus intéressante à notre avis, vu les spécificités de la famille 9900, est de faire appel à GC par une instruction XOP S,n : en effet, celle-ci provoque un appel à sous-programme défini par le vecteur d' XOP numéro n , avec préservation du contexte appelant et passage de l'adresse effective définie par S , dans le registre R 11 du nouvel espace de travail . Le programme appelant peut donc communiquer automatiquement à GC les paramètres dont il a besoin . On peut, par ailleurs, faire en sorte que le programme GC définisse les espaces de travail de ses sous-programmes d'interruption (par les vecteurs associés) de manière relative à son propre espace de travail, et seul le programme appelant fixe l'emplacement absolu de cet ensemble d'espaces de travail, en fixant la valeur du pointeur d'espace de travail de la partie princi--pale de GC (par le vecteur d' XOP) .

Plus précisément : nous utilisons à cet effet, dés le début de

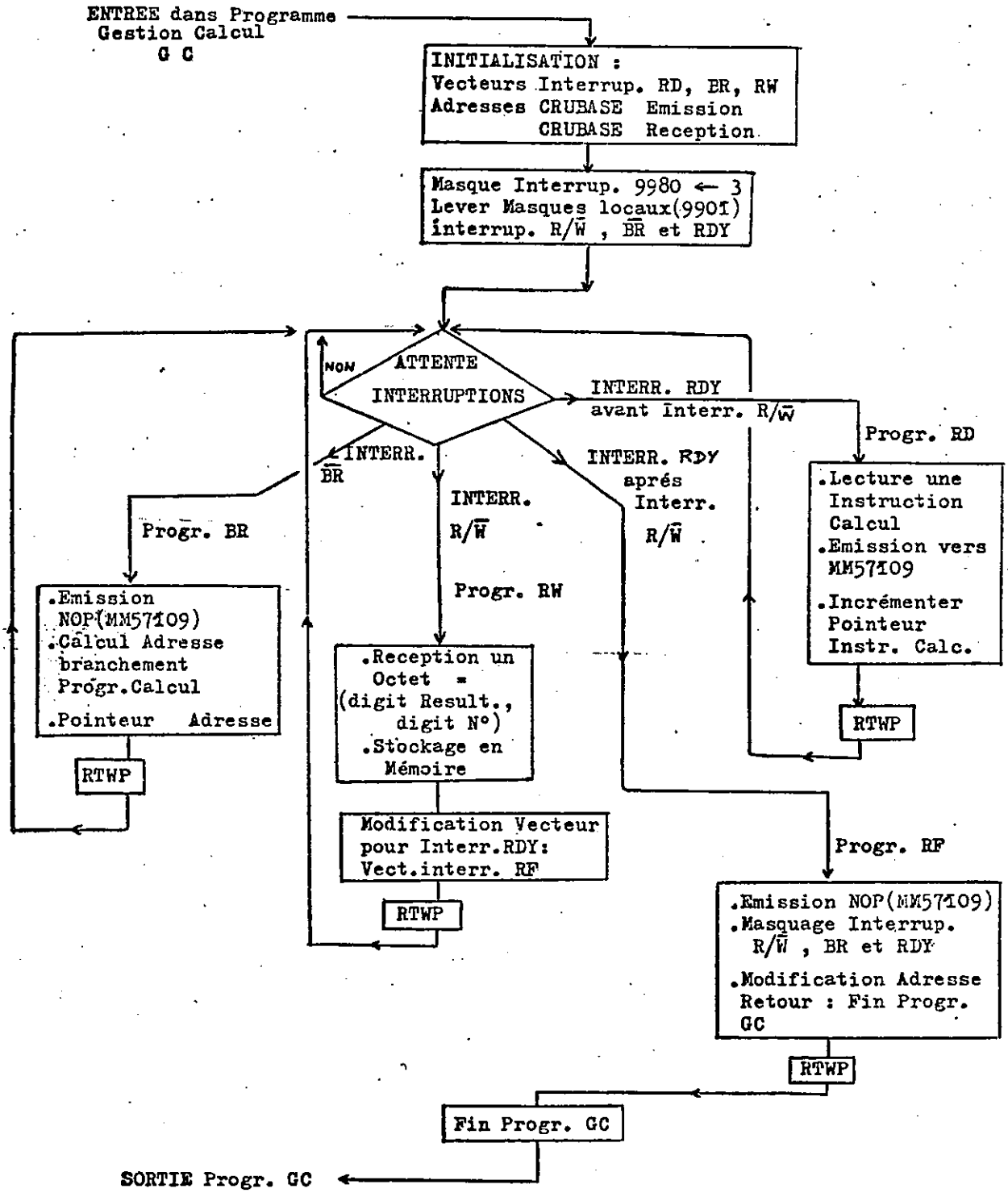


Fig. 2 ORGANIGRAMME DE PRINCIPE

Note: RTWP (Langage 9980) = Retour avec restauration de contexte .

GC, l'instruction STWP qui permet de charger dans un registre la valeur présente du pointeur d'espace de travail, puis nous calculons par diverses additions les valeurs des autres pointeurs (à charger dans les vecteurs d'interruption) ainsi que les adresses de certains registres associés (entre autres, pour définir préalablement les différentes CRUBASES utilisées par tous les modules) .

Précisons le passage des paramètres utiles à GC .

Notons CL l'adresse du programme de calcul (codes MM 57109)
RS l'adresse où stocker le résultat du calcul .

La communication de ces adresses à GC par le programme appelant, peut s'effectuer de la manière suivante , par exemple :

- Dans le programme appelant , CL est chargé à une adresse M1 , RS à l'adresse suivante , soit M2 ; puis on fait appel à GC par XOP M1,n (aprés chargement du vecteur d'XOP correspondant) .

- Dans le programme GC , les adresses CL et RS sont récupérées au moyen du registre R 11 , qui contient automatiquement l'adresse M1 :

MOV *R11+, Ri charge CL dans Ri , puis incrémente R11 , qui pointe alors sur l'adresse M2 ,

à la suite, MOV *R11, Rj charge RS dans Rj .

Noter que, dans le programme appelant, M1 peut être défini par un registre Rk de l'espace de travail de ce programme ; dans ce cas M2 correspond au registre R k+1 .

Remarque .

On pourrait aussi, en utilisant le même procédé, écrire GC totalement translatable en définissant les différentes valeurs de PC début des modules d'interruptions de GC de manière relative (par addi--tions) à son propre PC de départ , puisque la valeur de celui-ci peut être passé à GC par la XOP d'appel .

Positions relatives des différents registres de travail des modules composant GC .

Nous notons PP la "partie principale" de GC (le module "extérieur") , RD , RF , BR et RW , les programmes d'interruptions (cf. organigramme , fig.2) . WP(P) = la valeur du pointeur d'espace de travail associé à un programme P .

On fait alors WP(RD)=WP(RF)=WP(BR)=WP(PP)+ 4*2 et WP(RW)= WP(PP)+ 8*2 (le 2 est dû à l'organisation de la mémoire en octets) .

Si on note Ri(P) l'adresse du registre Ri associé à P , on a alors par exemple RO(RD)=R4(PP) et RO(RW)=R8(PP)=R4(RD) .
Les recouvrements partiels qui en résultent, permettent de passer automatiquement des paramètres (par ex. CL , RS) de PP aux autres modules, mais il faut veiller à ne pas perturber certains registres :
par exemple , la suite (R13,R14,R15)(PP) contient le contexte de retour de GC au programme appelant , cette suite correspond à (R9,R10,R11)(RD) et à (R5,R6,R7)(RW) , dans lesquelles il ne faut pas écrire .

Notons aussi qu'il est prudent de faire

$$(W(PP) \cup W(RD) \cup W(RW)) \cap W(PA) = \emptyset ,$$

où PA désigne le programme , et W l'espace de travail associé à un programme .

V. UNE CARTE D'AFFICHAGE DIRECT .

Nous avons réalisé une carte supplémentaire qui permet d'afficher directement les résultats issus des lignes DO1... DO4 , sur une rampe d'afficheurs LED 7 segments plus point décimal multiplexés ; plus exactement 8 afficheurs pour la mantisse, précédés d'une LED indépendante indicatrice du signe, et 2 afficheurs pour l'exposant, précédés aussi d'une LED de signe .

Il faut, pour utiliser cette carte, remplacer dans le programme RF l'émission de NOP (MM 57109) par OUT (MM 57109) , i.e. dans le programme (donné en annexe) remplacer en 036E l'instruction LDCR R1,7 par LDCR R2,7 . La présence continue de OUT sur les entrées du MM 57109 a pour effet l'émission cyclique du contenu du registre de sortie du MM 57109 en rafales séparées par la phase de lecture et exécution de OUT.

Les lignes DO1... DO4 sont connectées à un circuit CMOS latch décodeur driver d'anodes LED , le 4511 .

Les lignes DA1... DA4 sont utilisées pour démultiplexer les données, au moyen d'un circuit démultiplexeur latch 4514 .

Le signal $R/\overline{W}$ issu du MM 57109 est utilisé comme signal strobe sur ces 2 latch ; il sert aussi à engendrer un signal qui, sur l'entrée Enable du 4514, assure l'extinction des LED's entre deux rafales consé-cutives (car des signaux indésirables sont présents sur les lignes DO1... DO4) .

Le traitement du point décimal et des signes est un peu plus complexe. L'ensemble des deux signes est émis sous forme codée, lorsque

le chiffre 2 est présent sur les lignes DA1... DA4 : le bit sur DO1 représente le signe de l'exposant (éventuel), celui sur DO4 le signe de la mantisse .

Lorsque DA1... DA4 émettent le chiffre 3, les lignes DO1...DO4 émettent le chiffre DPP codant la position du point décimal :

DPP = 12 - DM , où DM est le numéro (de 1 à 8) du digit de mantisse à la droite duquel est placé le point décimal .

Un circuit quadruple latch D , le 4042, sert à saisir la valeur du code signe quand celui-ci se présente, puis ensuite à saisir et mémoriser (pendant tout le temps restant de la rafale) le code DPP . Le code signes est envoyé à des LED's dont la cathode commune est selectionnée lorsque <DA4,...,DA1> = la représentation binaire de 2 , par une des sorties du 4514 . Le code DPP est présenté aux lignes "sélection" d'un multiplexeur 8 entrées , le 4512 . Les entrées de celui-ci sont connectées aux sorties du 4514 (sélection des cathodes afficheurs) de manière à ce que la sortie du 4512 (alimentant les points décimaux) soit active lorsque son entrée sélectionnée et active correspond à l'afficheur où doit s'allumer le point décimal . Pour cela, si on note O4,...,O11 les sorties du 4514 , dans l'ordre des chiffres de la mantisse, et I0,...,I7 les entrées du 4512, on connecte Ii à O(11-i) , pour i = 1 à 7 .

VI. REMARQUES .

- La carte processeur calcul MM 57109 est issue, aprés modification minime d'une carte que nous avions mise au point pour associer à un 6802 (Motorola). Le programme gestion calcul est par contre trés différent .

- Le 9980 est relativement rallenti par le multiplexage de son bus données. Il est donc intéressant de remarquer que la carte calcul est facilement adaptable à d'autres membres de la famille 9900 plus performants, par exemple le processeur trés rapide 9995 .

- La logique des signaux de contrôle est assez "naturelle", et on peut penser que le programme GC peut s'adapter facilement à d'autres processeurs de calcul ayant une logique de contrôle analogue .

Dans le cas présent, la communication entre les deux unités, totalement asynchrones, obéit à un protocole demi-handshake initié par le processeur le moins rapide , ici le MM 57109 . Cependant, pour un processeur de calcul plus rapide (beaucoup plus cher!), il peut s'avérer

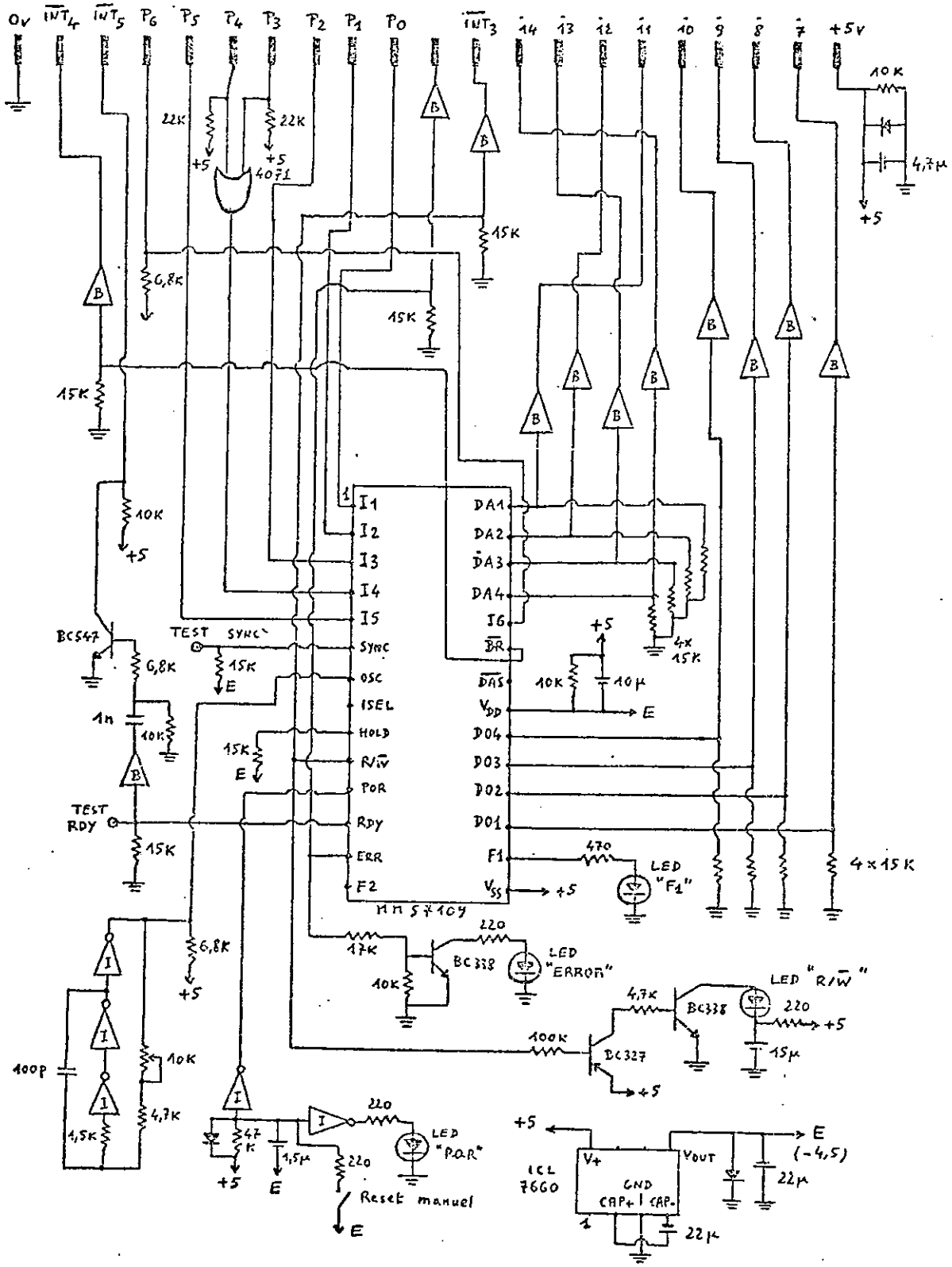
nécessaire de recourir à un handshake complet ; notons que dans le cas du MM 57109 , ceci pourrait être réalisé en utilisant en plus la ligne HOLD (MM 57109) .

ANNEXE 1 . Schéma électronique complet de la carte calcul.

ANNEXE 2 . Schéma électronique de la carte d'affichage direct.

ANNEXE 3 . Listing du programme GC , supposé appelé par une XOP . Il est suivi d'un exemple d'appel à GC , avec communication des paramètres (adresses) CL et RS (cf. IV) .

Note. Cet article a fait l'objet d'une publication, sous forme résumée, dans Micro-Bulletin n°9, Avril 83, pp.109-114, CNRS , Saint Martin d'Hères.

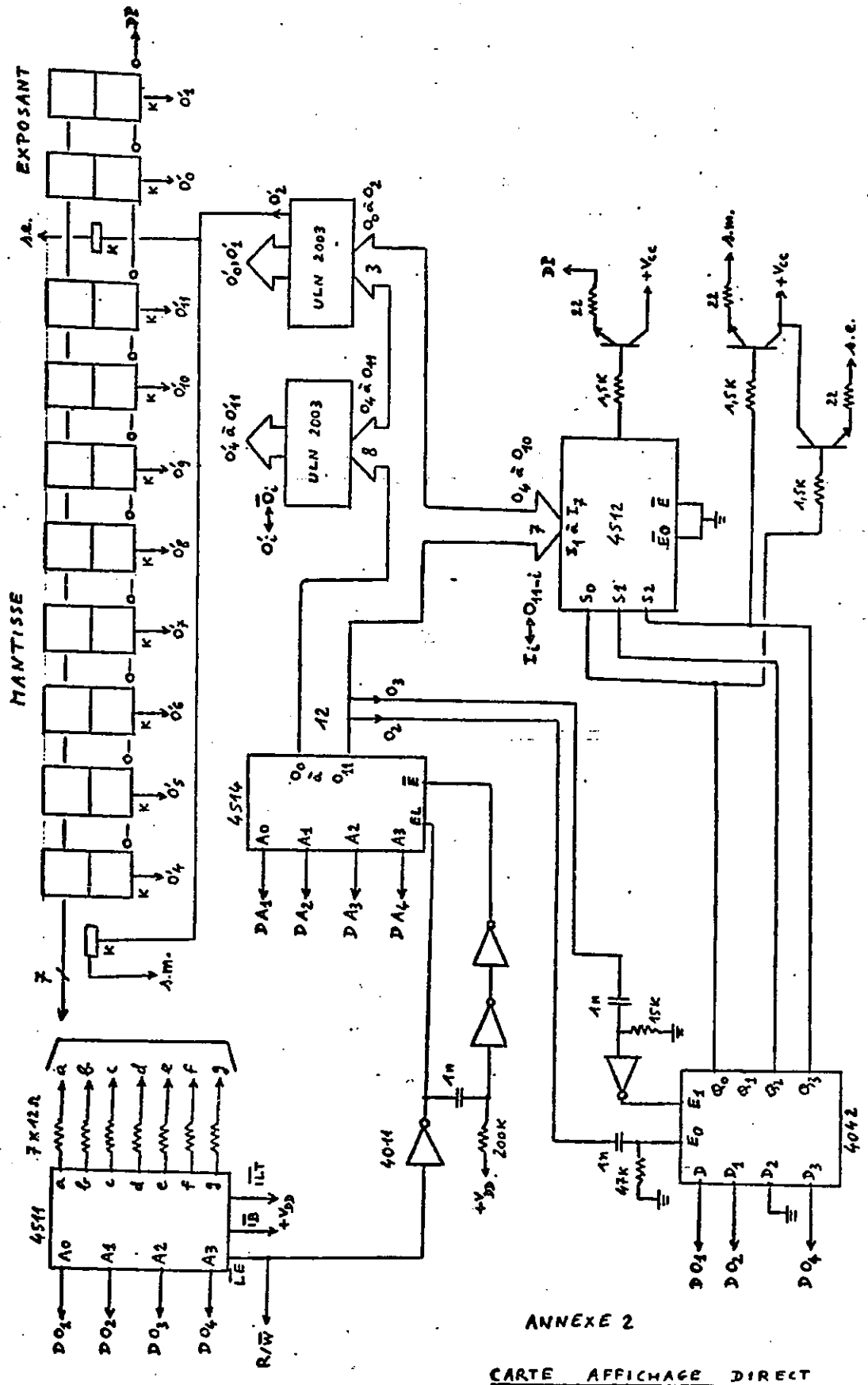


I : 4069 UB
V_{DD} = +5
V_{SS} = E

B : HC 14050B
V_{DD} = +5
V_{SS} = 0

ANNEXE 1

CARTE CALCUL



ANNEXE 2

CARTE AFFICHAGE DIRECT

Adresse	Code Objet	Etiquette	Mnémonique	
0300	02A0	PC	STWP R0	Début PP
0302	0201 0008		LI R1,8	
0306	A001		A R1,R0	
0308	C800 000C		MOV R0, @>C	
030C	C800 0008		MOV R0, @ 8	
0310	A001		A R1,R0	
0312	C800 0004		MOV R0, @ 4	
0314	A001		A R1,R0	
0318	0202 0020		LI R2,>20	
031C	C402		MOV R2, *R0	
031E	A001		A R1,R0	
0320	C402		MOV R2, *R0	
0322	A001		A R1,R0	
0324	0202 000E		LI R2, >E	
0328	C402		MOV R2, *R0	
032A	0200 0362		LI R0,RD	
032E	C800 000E		MOV R0, @>E	
0332	0200 037A		LI R0,BR	
0336	C800 000A		MOV R0, @>A	
033A	0200 0366		LI R0,RW	
033E	C800 0006		MOV R0, @ 6	
0342	0209 036E		LI R9,RF	
0346	C13B		MOV *R11+,R4	
0348	C21B		MOV *R11,R8	
034A	0205 7700		LI R5,>7700	
034E	0206 2600		LI R6,>2600	
0352	1EF0		SBZ -16	
0354	1DF3		SBO -13	
0356	1DF4		SBO -12	
0358	1DF5		SBO -11	
035A	0300 0003		LIMI 3	
035E	10FF		JMP $	
0360	0380		RTWP	Fin PP, Retour

.../ ...

ANNEXE 3 . PROGRAMME GC

```
Adresse    Code Objet    Etiquette    mnémonique
0362       31F0          RD           LDCR *RO+,7      Début RD
0364       0380                       RTWP             Fin RD,Ret.à PP
0366       3630          RW           STCR *RO+,8      Début RW
0368       C801  000E                 MOV R1, @>E
036C       380                        RTWP
036E       31C1          RF           LDCR R1,7        Début RF
0370       1EF3                       SBZ -13
0372       1EF4                       SBZ -12
0374       1EF5                       SBZ -11
0376       05CE                       INCT R14         Modific.Ret.
0378       0380                       RTWP             Fin RF,Ret.à PP
037A       31C1          BR           LDCR R1,7        Début BR
037C       0600                       DEC RO
037E       D0D0                       MOVB *RO,R3
0380       0883                       SRA R3,8
0382       A003                       A R3,RO
0384       0380                       RTWP             Fin BR, Ret.à PP
```

Exemple d'Appel à GC par XOP :

```
:          :                          :
0224       0200  0300                 LI RO,PC
0228       C800  0042                 MOV RO, @>42
022C       0200  0500                 LI RO,WP
0230       C800  0040                 MOV RO, @>40
0234       0200  0392                 LI RO,CL
0238       0201  0386                 LI R1,RS
023C       2C00                       XOP RO,0
:          :                          :
:          :                          :
```

ANNEXE 3 (Suite) . PROGRAMME GC

APPEL à GC

UNE PREUVE DIRECTE DE L'EQUIVALENCE ENTRE MACHINE DE TURING ET MACHINE A CASES ADRESSABLES.

REMARQUES SUR LA NOTION DE COMMUTATION

J.Ph.Lehmann

(suite)

C.R.:Subject classification informatics
4.20 5.20 5.26 5.29 6.30 6.32

Résumé:

A l'époque de Turing, l'intuition du calculable se manifestait de diverses façons. La préoccupation essentielle était de construire une ou plusieurs bonnes définitions d'une notion imprécise.
Aujourd'hui l'intuition se développe à travers une pratique, celle de l'utilisation des ordinateurs, de plus en plus répandue.
Dans cet article, on établit un lien direct, entre un ordinateur très simple, mais qui mérite néammoins son nom, et la machine de Turing. On ne passe par l'intermédiaire d'aucune autre définition du calculable.

Au passage, on examine dans le détail, les caractéristiques d'un langage de programmation et de sa sémantique, du point de vue structurel et logiciel.

La notion d'état dans une machine de Turing, est éclairée à travers la notion de commutation intimement liée aux langages de programmation.
On transpose certains résultats de Shannon sur les machines universelles de Turing à 2 états, dans le langage de la machine à cases adressables, et on donne une forme générale des programmes dans ce langage.

2ème PARTIE

- A -

INTRODUCTION

- B -

L'EQUIVALENCE

- C -

REMARQUES

Plan de l'exposé(suite)

2ème PARTIE

- A -

INTRODUCTION

Dans la première partie de cet exposé (i), le point de départ était fourni par le langage de la machine à cases adressables, que nous désignons par abréviation, M.A.C..
Cette machine, définie par L.Nolin (ii), a été longuement décrite, ainsi que le langage qui lui est associé.

Nous avons appliqué sur cet automate, un procédé de réduction à la fois structurel et logiciel. Le but de la méthode mise en oeuvre, était d'aboutir à une machine à la fois plus simple, du point de vue de sa structure, et la plus dépouillée possible, du point de vue du langage. Le tout, de façon que la forme la plus réduite de la machine, permette de réaliser tout ce qu'il était possible de faire, alors qu'on disposait de la forme et du langage les plus évolués.

Une des caractéristiques de la méthode suivie, consistait, à chaque pas du processus de réduction, à ne définir que des sous-machines et des sous-langages des précédents obtenus; ainsi une structure d'ordre, pouvait être installée sur l'ensemble des machines construites, et la question du caractère minimal de l'une de ces formes pouvait être examinée.

Successivement, les langages suivants ont été définis, dont nous donnons à nouveau les formes:

(i):publiée dans le précédent numéro de ce bulletin.
(ii):cf. Ière partie et bibliographie.

- Formes du langage M.A.C.

$Ca_i := Ca_j$	(1)
$Ca_i := \bar{0}$	(2)
$Ca_i := Ca_i + 1$ (k)	(3)
Si $Ca_i = \bar{0}$ vers e_i	(4)
Si Cu = $\bar{1}$ vers e_i	(5)
vers e_i	(6)
$Ca_i := Cf_n$	(7)
$Cf_n := Ca_i$	(8)
$Cf_n :=$ ""	(9)
Av f_n	(10)
Ar f_n	(11)
Si Cf_n = "" vers e_i	(12)

La forme F_o

$Ca_i := Ca_i + 1$ (k)	
Si $Ca_i = \bar{0}$ vers e_i	
$Ca_i := Cf$	
$Cf := Ca_i$	
Av	
Ar	

La forme F_1

$\bar{a}$	application de la permutation ($\bar{0}$ $\bar{1}$) ⟶ ($\bar{1}$ $\bar{0}$) au contenu de a, unique case de la M.C.
Si Ca=$\bar{0}$ vers e_i	
Ca := Cf	
Cf := Ca	
Av	
Ar	

La forme F_2

Si $\bar{0}$ vers e_i	(1") Test sur la seule case de la M.C.
af	(2") Transfert de la file vers la M.C.
fa	(3") Transfert de la M.C. vers la file
Av	(4") Avant sur la file
Ar	(5") Arrière sur la file

Il n'y a plus que deux formes $\bar{0}$ et $\bar{1}$.

Forme F_3

Cette forme impose que $\bar{0}$ et $\bar{1}$ soient déposés, au préalable sur la file. Au départ la case où pointe l'index est précédée du $\bar{1}$ et du $\bar{0}$.

Les deux automates que nous allons maintenant comparer, machine de Turing, M.T., et machine à cases adressables, ne peuvent évidemment pas l'être dans le cadre de la relation d'ordre à laquelle il a été fait allusion plus haut.

Sur le plan logiciel, aucune d'elles n'est extension de l'autre, et d'autre part elles sont structurellement différentes. Il s'agira donc, de représenter dans un langage, toutes les descriptions permises par l'autre langage et réciproquement.

De façon générale, ce type de problèmes appelle la remarque suivante: plus le langage dans lequel on cherche à représenter les processus décrits dans une autre langue, est évolué, c'est à dire assorti de commodités, plus l'opération est rendue aisée. A cet égard, la forme F_2 ou F_3 du langage M.A.C., à cause justement de leur caractère extrêmement réduit, devraient être écartées, lorsqu'on a en vue de traduire en M.A.C., la langue M.T.; néammoins nous utiliserons la forme F_2.

En effet, bien que la forme F_o soit évidemment plus commode, il se trouve que, dans le cas particulier, la traduction n'est pas spécialement compliquée dans la forme F_2; mais surtout, il est raisonnable d'espérer, que, dans cette forme très dépouillée, voisine par certains aspects de la M.T., la nature des liens existant entre les deux automates apparaisse de manière plus directe.

Quant à l'opération inverse, consistant à traduire dans le langage M.T. des programmes quelconques écrits en M.A.C., il est clair qu'alors, la forme F_3 constituera un avantage, du fait même de son caractère minimal.

La M.T. que nous utiliserons sera une machine binaire, à laquelle on sait ramener toute M.T. dont l'alphabet comprend un nombre fini quelconque de signes.

Les schémas de ces machines seront donnés selon les conventions suivantes:

x, q_i ------ x', q_j, X

avec x et x' égaux à 0 ou 1, et X égal à G ou D, pour Gauche et Droite, les q_i désignant les états. Lorsque, dans le schème, le signe résultat ou le nouvel état défini, ne seront pas changés, on n'indiquera rien dans la partie droite; de même, lorsqu'il

n'y aura pas de déplacement.

Ainsi, par exemple $\bar{0}, q_1 \longrightarrow \bar{1}, q_1, D$

sera noté: $\bar{0}, q_1 \longrightarrow \bar{1}, D$

de même $\bar{1}, q_1 \longrightarrow \bar{1}, q_2$

sera noté: $\bar{1}, q_1 \longrightarrow q_2$

On utilisera également la présentation sous forme de tableaux à double entrée.

- B -

L'EQUIVALENCE

I COMPILATION MACHINE DE TURING EN M.A.C.

1) CONSTRUCTION D'UN ALGORITHME UNIVERSEL

Considérons un schème de Turing quelconque. A un moment donné du processus de calcul, l'automate est dans un certain état q_i, et l'information présente sur la file est soit 0, soit 1; le schème nous indique alors quelles sont les actions qu'il faut exercer: écrire tel signe, se déplacer à gauche ou à droite, éventuellement rester sur place, enfin prendre l'état indiqué, pour passer à la suite de l'exécution du calcul.
En d'autres termes, le schème nous indique, d'une part ce qu'il faut écrire et quel déplacement opérer, d'autre part où aller chercher la suite des instructions; le premier type de renseignement nous permettra de définir des instructions, le second, la commutation dans le programme, pour l'essentiel.

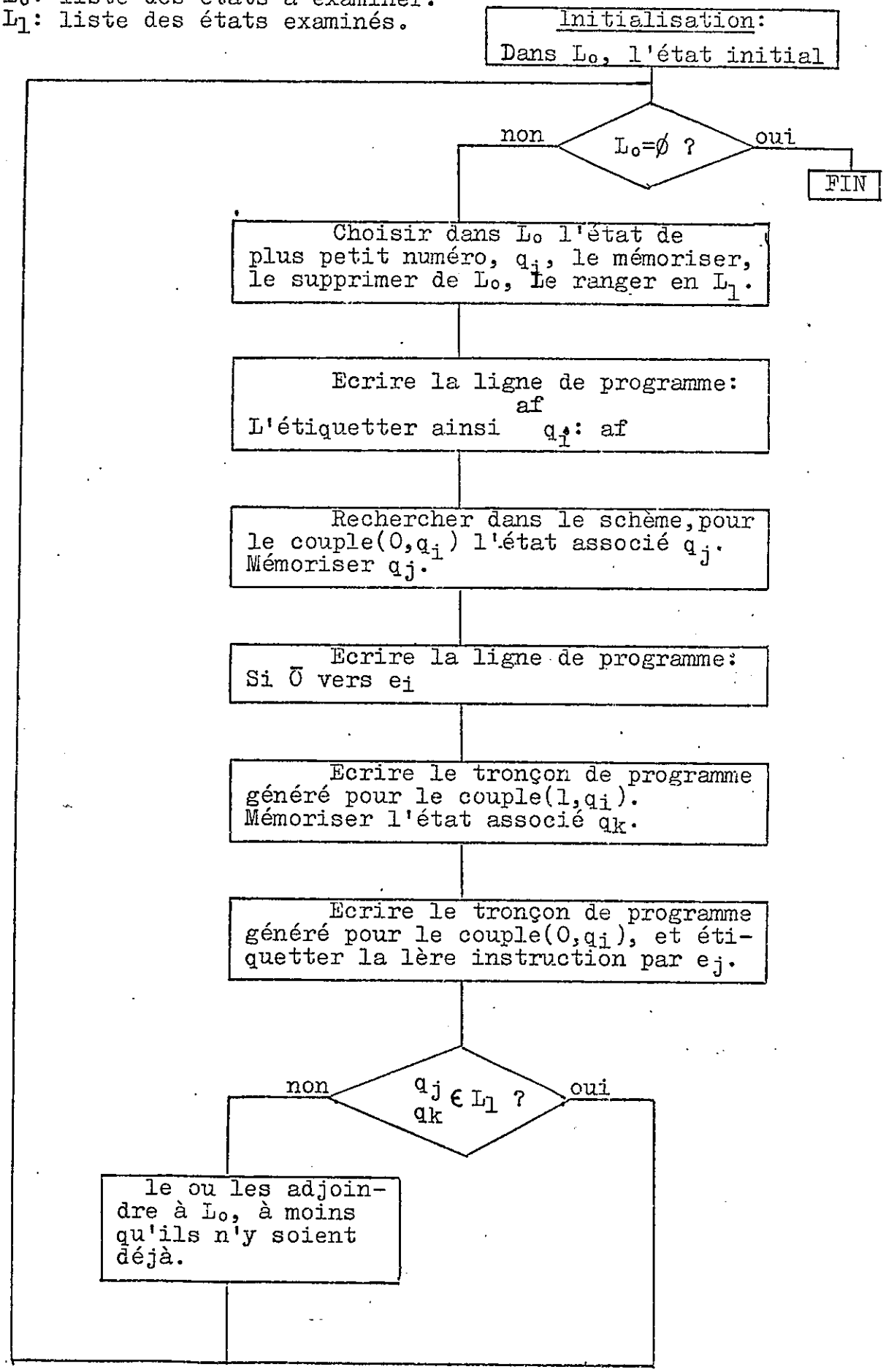
Nous commencerons par donner les tronçons de programme M.A.C. qui devront être générés, pour tous les couples (signe, état) définis par le schème.
Ensuite on indiquera, comment construire automatiquement la totalité du programme, en enchainant ces tronçons entre eux, en précisant les conditions d'initialisation, et en donnant les conditions d'arrêt du processus de construction.

- Génération des tronçons:

Schème de Turing	Correspondant M.A.C.
$\overline{1}, q_i$ —— $\overline{0}, q_k$	$\bar{a}$ fa Si $\overline{0}$ vers q_k
$\overline{1}, q_i$ —— $\overline{1}, q_k$	$\bar{a}$ Si $\overline{0}$ vers q_k
$\overline{1}, q_i$ —— $\overline{0}, q_k, G$	$\bar{a}$ fa Ar Si $\overline{0}$ vers q_k
$\overline{1}, q_i$ —— $\overline{0}, q_k, D$	$\bar{a}$ fa Av Si $\overline{0}$ vers q_k
$\overline{1}, q_i$ —— $\overline{1}, q_k, G$	Ar $\bar{a}$ Si $\overline{0}$ vers q_k
$\overline{1}, q_i$ —— $\overline{1}, q_k, D$	Av $\bar{a}$ Si $\overline{0}$ vers q_k
$\overline{0}, q_i$ —— $\overline{0}, q_k$	Si $\overline{0}$ vers q_k
$\overline{0}, q_i$ —— $\overline{1}, q_k$	$\bar{a}$ fa $\bar{a}$ Si $\overline{0}$ vers q_k
$\overline{0}, q_i$ —— $\overline{0}, q_k, G$	Ar Si $\overline{0}$ vers q_k
$\overline{0}, q_i$ —— $\overline{0}, q_k, D$	Av Si $\overline{0}$ vers q_k
$\overline{0}, q_i$ —— $\overline{1}, q_k, G$	$\bar{a}$ fa Ar $\bar{a}$ Si $\overline{0}$ vers q_k
$0, q_i$ —— $1, q_k, D$	$\bar{a}$ fa Av $\bar{a}$ Si $\overline{0}$ vers q_k

- Construction du programme

L_o: liste des états à examiner.
L_1: liste des états examinés.

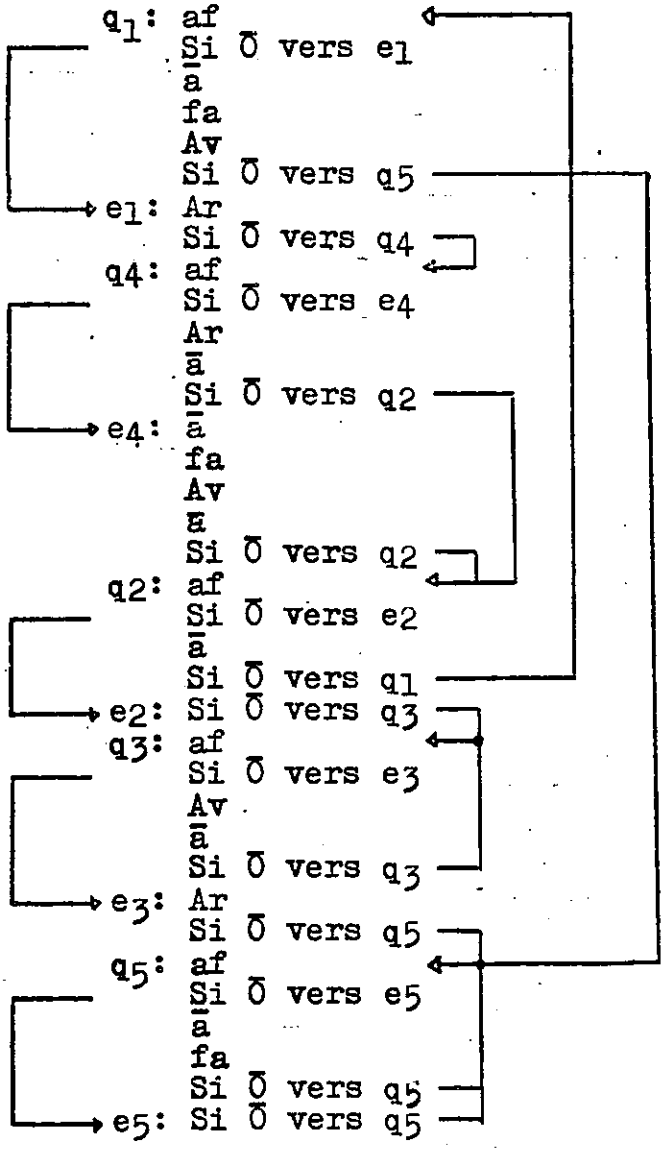


2) UN EXEMPLE

Soit le schème:

	q_1	q_2	q_3	q_4	q_5
0	$00q_4G$	$0q_3$	$0q_5G$	$1q_2D$	$0q_5$
1	$0q_5D$	$1q_1$	$1q_3D$	$1q_2G$	$0q_5$

L'algorithme que nous venons d'exposer au paragraphe précédent, génerera le programme M.A.C. suivant:



On a indiqué à droite les branchements obligés, à gauche ceux qui sont fonction du calcul.

II COMPILATION M.A.C. EN MACHINE DE TURING

1) ANALYSE DU PROBLEME

Dans le paragraphe précédent, les actions commandées dans un certain état étaient représentées par des tronçons de programmes spécifiques, qui pouvaient tous être prédéfinis. Le passage d'un état à un autre s'obtenait par une organisation convenable de la commutation.
En définitive, on aboutissait assez simplement au résultat, en utilisant le langage M.A.C. sous la forme F2, à cause de la formulation explicite des transitions d'état qui caractérise les schèmes de Turing.
Comme on va le voir, le problème inverse est différent, dans la mesure où, la commutation est à la fois implicite et explicite; de plus la structure de la M.A.C. comprend une mémoire centrale, si réduite soit-elle, et son absence dans la M.T. pose forcément le problème de sa représentation.

- Forme générale des programmes

Le procédé qu'il faut imaginer, doit pouvoir s'appliquer sur un programme quelconque. Bien entendu, aucune considération de cohérence, si ce n'est d'ordre syntaxique, ne doit intervenir; il faut donc définir ce qu'est un programme quelconque, sans se préoccuper de ce qu'il fait ou de ce qu'il peut représenter, dès lors qu'il est correctement formé.
Pour tout ce qui suit, nous utiliserons la forme F3 du langage M.A.C., dont on a exposé le caractère minimal.

Définition: un programme de F3 est une suite finie quelconque d'instructions.

une instruction est soit Av, soit Ar, soit af, soit fa, soit encore une spécification du schéma infini Si $\bar{0}$ vers e_i.
Les instructions peuvent être étiquettées.

un programme est dit correctement formé, si, pour chaque instruction du type Si $\bar{0}$ vers e_i, il existe dans le programme une instruction exactement, étiquettée e_i.

il existe une instruction spéciale fin qui impose l'arrêt de l'exécution.

Remarque: la définition retenue pourrait sembler trop large, dans la mesure où de nombreuses simplifications paraissent possibles, lorsqu'on se donne

une suite quelconque d'instructions.
Par exemple on pourrait remplacer toutes les suites AvAv.....(m fois)..AvArAr..(n fois)..Ar par AvAv...(m-n fois)...Av quand $m \geqslant n$. En fait ceci n'est possible que si aucune des instructions n'est étiquettée.
On s'aperçoit rapidement, qu'à l'exception de quelques rares séquences-types, répétition de af ou de fa consécutifs, les simplifications ne sont possibles que dans des cas de figure très particuliers.
Il faut donc abandonner toute idée visant à ramener les programmes à des formes standard, et considérer le problème dans sa généralité, du moins à ce stade de l'analyse.

Remarque: Plus haut, on a fait apparaître une instruction spéciale: fin. Bien que jusqu'ici, on ne l'ait jamais mentionnée , celle-ci jouait son rôle de façon implicite: un programme était terminé, parce que, sur la feuille, à partir d'un certain endroit, plus rien n'était écrit. De la même manière, la question du début d'un programme se poserait, mais la façon dont nous initialisons les processus nous permet d'éluder la question. Néanmoins, à cause du formalisme plus précis qui sera requis, il deviendra indispensable de marquer explicitement l'arrêt de l'exécution, grâce à l'instruction fin.

- Traduction des actions-instructions et des actions-commutations

Les instructions Av, Ar, af, fa, seront dites du type A.
Les instructions provenant du schéma Si $\bar{0}$ vers e_i, seront dites du type B.
Deux aspects semblent devoir être considérés:

D'une part, il faut traduire les actions décrites par les instructions du type A,

d'autre part, il faut exprimer la progression dans le programme, qui se fait, soit de façon implicite en passant en général de chaque instruction à la suivante, soit de manière explicite, à la rencontre des instructions du type B.

En ce qui concerne le premier aspect, les actions définies par Av et Ar, trouvent à l'évidence leurs correspondantes en D et G.

Pour réaliser af un premier type de problème se pose: il existe bien dans le M.T. une tête de lecture, mais le

transfert du signe lu dans la file, vers une case de mémoire centrale, est structurellement hors d'atteinte. Une telle opération vise en fait à mémoriser l'information, pour éventuellement l'utiliser plus tard; le moyen de réaliser l'équivalent s'obtiendra par la création d'un état, dont la fonction sera précisément d'indiquer, quelle est l'information à mémoriser.

Dès lors, la réalisation de fa pourra être menée à bien, puisque l'état de l'automate, indiquera lui-même quel est le signe qu'il faut transférer sur la file; quant à l'opération de transfert elle-même, les moyens existent déjà dans l'automate de Turing pour y pourvoir.

Ainsi, on aperçoit déjà, une partie des fonctions qui devront être dévolues aux états de la M.T. que l'on souhaite définir: mémoriser la dernière information lue, pour permettre la réalisation éventuelle, soit de fa, soit d'une instruction du type B

En ce qui concerne le second aspect, c'est à dire la progression dans les programmes, les choses sont un peu plus délicates: non seulement le type d'instructions B pose un problème, mais encore le fait "naturel" de passer d'une ligne de programme à la suivante, contribue à masquer une partie essentielle de la question.

Pour éclairer les choses, considérons la séquence suivante, dont les instructions peuvent être traduites immédiatement, de manière que notre attention ne se fixe que sur la progression dans la séquence:

Av
Av
Av
Av

Comment déterminer le schème de Turing correspondant?
Il n'y a pas de test à opérer, ni de transfert en lecture ou en écriture; il ne s'agit que de passer d'une instruction à la suivante, en effectuant chaque fois un déplacement à droite; l'essentiel se limite donc à exercer, un certain nombre de fois, la même action; en d'autres termes, contrôler la progression, c'est compter les instructions qu'on dérou-

le, pour savoir exactement, où on en est dans l'exécution du programme.

Une autre manière de considérer la question est la suivante: soit un programme M.A.C. se déroulant; stoppons son exécution à un instant quelconque, non trivial(ni le début ni la fin), et considérons l'état du système dans sa totalité:

La file a subi un certain nombre de modifications tant par les déplacements qui se sont exercés, que par les opérations d'écriture qui ont pu avoir lieu. La mémorisation de l'ensemble de ces évènements ne pose aucun problème, puisque justement, celle-ci est entièrement inscrite dans la structure courante de la file (cette structure englobant notamment la position courante de l'index).

Voyons l'état du reste du système: celui-ci est défini par deux faits exactement, la connaissance de l'instruction courante du programme et la connaissance de l'état interne de la machine, totalement déterminé par le contenu de sa mémoire unique.

De sorte que, si l'on souhaite "reprendre la main", il nous suffit d'être assuré de 3 circonstances:

Un état de la file garanti comme celui de l'instant où le calcul a été stoppé, index compris.

L'indication du contenu de a, au même instant.

L'indication de la ligne de programme active, au même instant.

La première de ces circonstances est entièrement extérieure à la structure du programme.

La deuxième a déjà été étudiée plus haut.

Il ne reste donc plus qu'à créer le moyen d'indiquer, dans la suite des instructions du programme, laquelle est active; une méthode simple permettant de désigner, dans une liste ordonnée, un élément précis, consistera à donner le numéro de la ligne correspondante.

Compte tenu de l'ensemble des remarques précédentes, il est clair que le schème de Turing recherché, devra, à travers sa structure, donner une image de la liste M.A.C.

correspondante, c'est à dire représenter, grâce à certains états, à la fois, les numéros des lignes de programme actives, et les numéros de celles qui devront suivre. La réalisation des instructions du type B se fera alors ainsi: les caractéristiques des états nous indiqueront d'une part, le contenu courant de a, ce qui permettra de décider si le test est positif ou non, d'autre part, selon le résultat du test, quel est le numéro de ligne suivant, c'est à dire le nouvel état à atteindre.

- Problèmes d'initialisation

la position de l'index, et la configuration de la file, au départ du processus ne posent pas de problème: celles-ci seront reprises dans le schème de Turing, exactement comme elles auront été spécifiées dans la M.A.C.
Ce qu'il faut préciser ici, c'est le contenu initial de a; en effet, dans un programme M.A.C., le caractère déterministe du processus n'est assuré (sauf cas particulier), que dans la mesure où ce contenu est fixé.
Les programmes qu'il nous faudra compiler, seront donc assortis de cette indication, et nous verrons comment représenter cette condition initiale particulière.

2) CONSTRUCTION D'UN ALGORITHME UNIVERSEL

Pour des raisons de commodité, on remplacera le programme M.A.C. qu'il faut compiler, par un programme isomorphe, dans lequel les étiquettes e_i auront été changées en e_{j_i}, j_i étant le numéro de la ligne dont l'étiquette était e_i. Ainsi le nom de l'étiquettage correspondra au numéro de ligne.

Dans un premier temps nous donnons la partie essentielle de l'algorithme, c'est à dire l'aiguillage qui permettra de générer automatiquement, pour chaque instruction M.A.C. rencontrée, le schème de Turing correspondant.

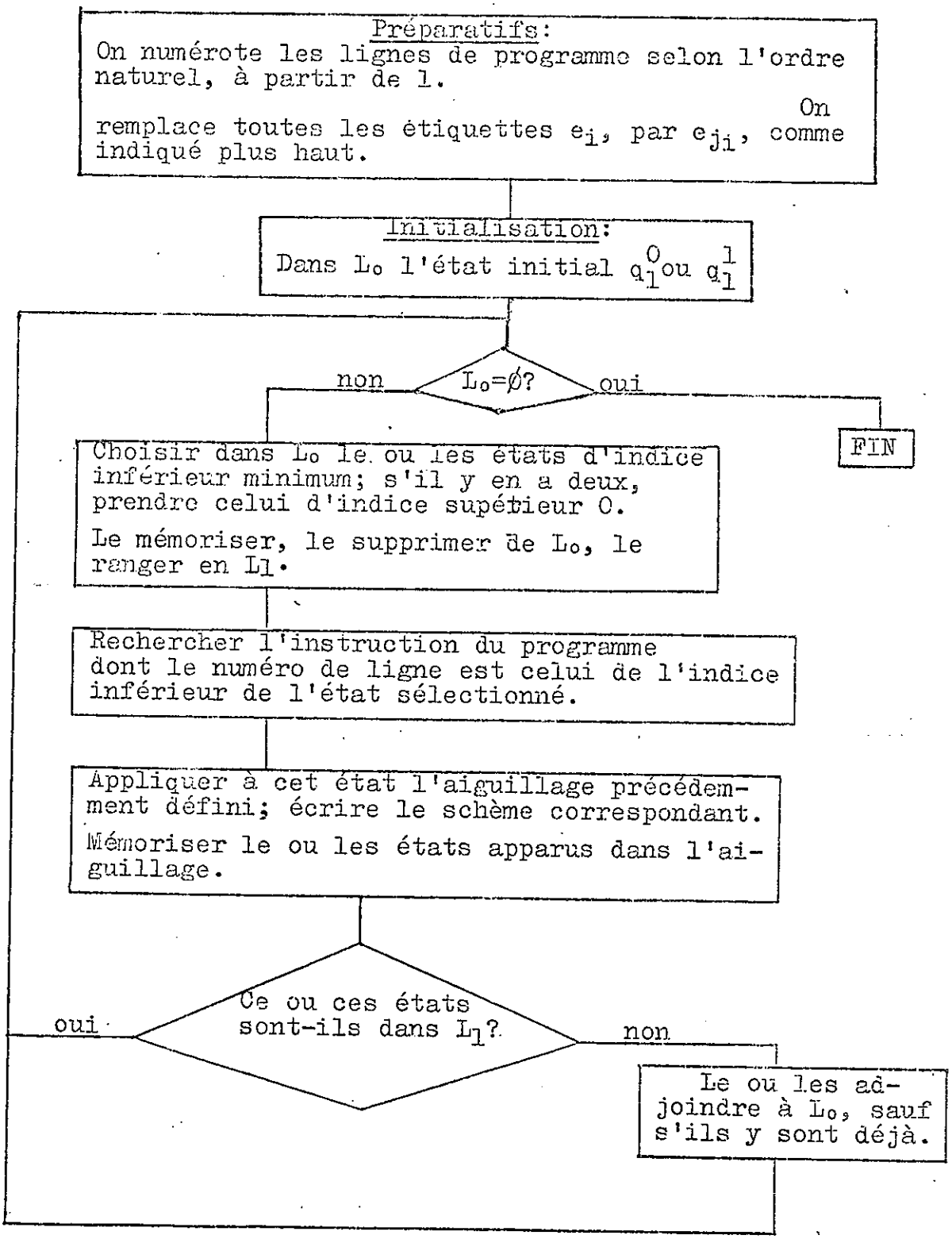
Puis nous donnerons, la structure complète de l'algorithme, avec ses enchainements.

- L'aiguillage

Ligne de programme M.A.C.	Correspondant M.T.
Av	$\bar{0},q_i^x \longrightarrow \bar{0},q_{i+1}^x,D$ $\bar{1},q_i^x \longrightarrow \bar{1},q_{i+1}^x,D$
Ar	$\bar{0},q_i^x \longrightarrow \bar{0},q_{i+1}^x,G$ $\bar{1},q_i^x \longrightarrow \bar{1},q_{i+1}^x,G$
af	$\bar{0},q_i^x \longrightarrow \bar{0},q_{i+1}^0$ $\bar{1},q_i^x \longrightarrow \bar{1},q_{i+1}^1$
fa	$\bar{0},q_i^x \longrightarrow x,q_{i+1}^x$ $\bar{1},q_i^x \longrightarrow x,q_{i+1}^x$
Si $\bar{0}$ vers e_j	$\bar{0},q_i^x \longrightarrow \bar{0},q_{x(i+1)+(1-x)j}^x$ $\bar{1},q_i^x \longrightarrow \bar{1},q_{x(i+1)+(1-x)j}^x$
<u>fin</u>	$\bar{0},q_i^x \longrightarrow \bar{0},q_T^x$ $\bar{1},q_i^x \longrightarrow \bar{1},q_T^x$

L'état initial est q_1^x, x valant $\bar{0}$ ou $\bar{1}$ selon le contenu initial qui est défini pour a.
Les états q_T^x sont des états terminaux, qui indiquent l'arrêt du processus de calcul.

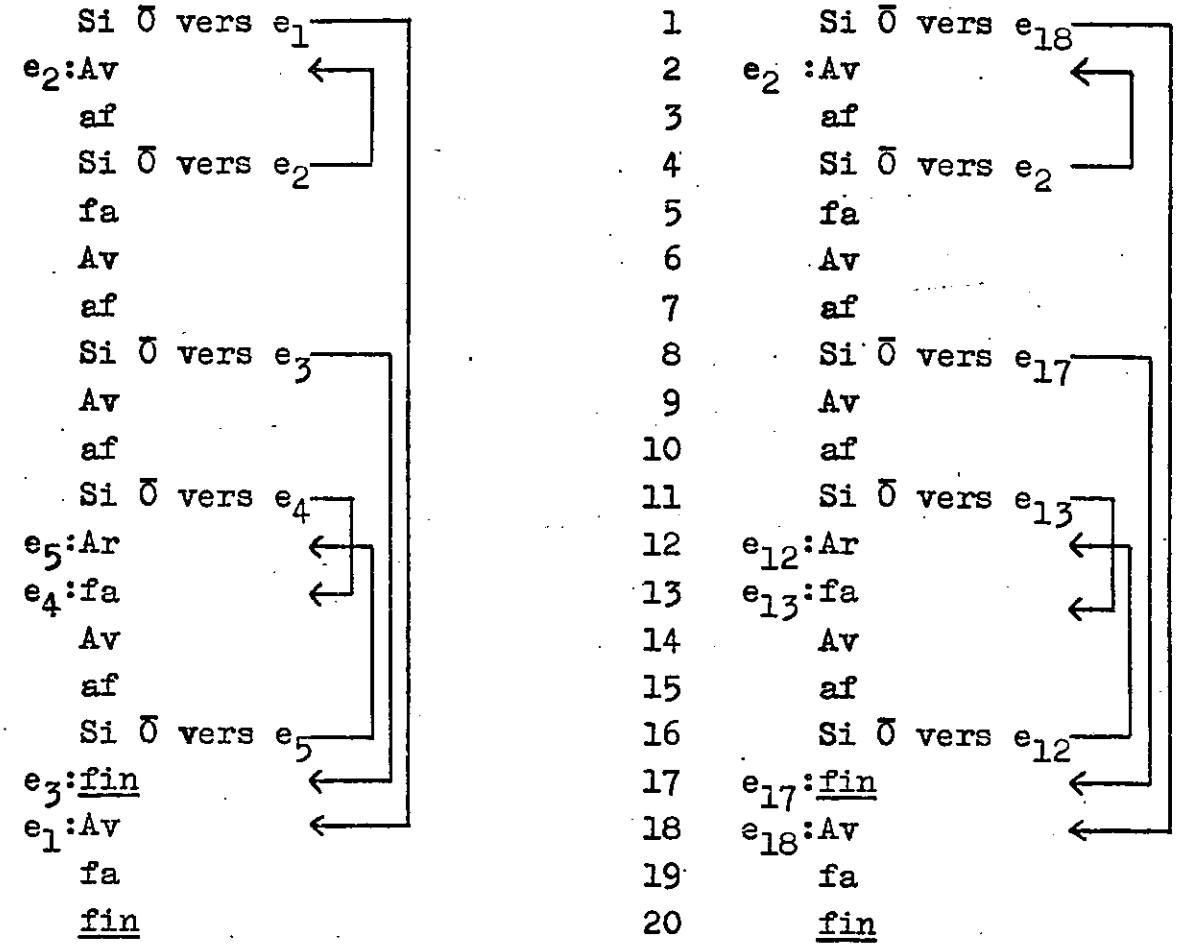
- Construction du schème de Turing



L_o est la liste des états à examiner.
L_1 est la liste des états examinés.

3) UN EXEMPLE

On se donne un programme quelconque, puis on le transforme en numérotant ses lignes, et en changeant les étiquettes comme il a été indiqué:



On génère alors le schème de Turing associé, en utilisant l'algorithme universel qui a été décrit plus haut; page suivante, on a ainsi construit le schème correspondant au programme ci-dessus.

- Si le contenu initial de a est $\bar{0}$

	q_1^0	q_{18}^0	q_{19}^0	q_{20}^0
$\bar{0}$	q_{18}^0	q_{19}^{0D}	$\bar{0}q_{20}^0$	q_T^0
$\bar{1}$	q_{18}^0	q_{19}^{0D}	$\bar{0}q_{20}^0$	q_T^0

- Si le contenu initial de a est $\bar{1}$

Les états sont écrits dans l'ordre dans lequel ils sont découverts; ils sont numérotés, en haut du tableau dans l'ordre dans lequel ils ont été examinés.

	1	2	3	4	7	5	6	8	9	10	11	12	26	13	14	15	16
	q_1^1	q_2^1	q_3^1	q_4^0	q_4^1	q_2^0	q_3^0	q_5^1	q_6^1	q_7^1	q_8^0	q_8^1	q_{17}^0	q_9^1	q_{10}^1	q_{11}^0	q_{11}^1
$\bar{0}$	q_2^1	q_3^{1D}	q_4^0	q_2^0	q_5^1	q_3^{0D}	q_4^0	$\bar{1}q_6^1$	q_7^{1D}	q_8^0	q_{17}^0	q_9^1	q_T^0	q_{10}^{1D}	q_{11}^0	q_{13}^0	q_{12}^1
$\bar{1}$	q_2^1	q_3^{1D}	q_4^1	q_2^0	q_5^1	q_3^{0D}	q_4^1	$\bar{1}q_6^1$	q_7^{1D}	q_8^1	q_{17}^0	q_9^1	q_T^0	q_{10}^{1D}	q_{11}^1	q_{13}^0	q_{12}^1

	18	17	19	20	21	22	23	24	26	25	27
	q_{13}^0	q_{12}^1	q_{13}^1	q_{14}^0	q_{14}^1	q_{15}^0	q_{15}^1	q_{16}^0	q_{16}^1	q_{12}^0	q_{17}^1
	$\bar{0}q_{14}^0$	q_{13}^{1G}	$\bar{1}q_{14}^1$	q_{15}^{0D}	q_{15}^{1D}	q_{16}^0	q_{16}^0	q_{12}^0	q_{17}^1	q_{13}^{0G}	q_T^1
	$\bar{0}q_{14}^0$	q_{13}^{1G}	$\bar{1}q_{14}^1$	q_{15}^{0D}	q_{15}^{1D}	q_{16}^1	q_{16}^1	q_{12}^0	q_{17}^1	q_{13}^{0G}	q_T^1

Bien entendu, certaines simplifications pourtant possibles, n'ont pas été faites; par exemple les états q_2^0 et q_3^0 auraient pu être supprimés, et d'autres encore, mais ce n'est pas le but recherché, qui d'ailleurs, ne pourrait être atteint que dans des cas particuliers.

III FAMILLES DE SCHEMES M.T. ET DE PROGRAMMES M.A.C.

Signalons que les deux types d'algorithmes que nous avons étudiés dans les paragraphes précédents, permettent en fait de définir, des classes de programmes et des classes de schèmes:

La donnée d'un schème de Turing, ne définit un algorithme précis que dans la mesure où l'on indique l'état initial; ainsi, pour un schème donné, il y a autant d'algorithmes que d'états; La méthode qui a été exposée construit les programmes associés à chacun d'eux; il suffit d'initialiser l'algorithme universel, en mettant dans L_o l'état qu'on spécifie initial. Le programme correspondant sera alors obtenu à partir de l'ancien, en définissant la première instruction comme celle dont l'étiquette est le nom du nouvel état initial.

Le problème inverse appelle les mêmes remarques: la donnée d'un programme M.A.C. est accompagnée d'informations implicites: le contenu initial de a, et le point d'entrée dans le programme; habituellement cette dernière condition est associée au fait qu'une instruction <u>début</u> est sous-entendue, juste au-dessus de la première ligne de programme écrite. Toutes les autres possibilités d'initialisation sont obtenues en définissant une valeur de a et en plaçant <u>début</u>, quelque part dans le programme juste avant une instruction. Si n est le nombre des lignes du programme, on a alors 2n initialisations possibles. L'algorithme universel qu'on a présenté, permet d'engendrer les schèmes de Turing correspondants: il suffit d'indiquer pour L_o, au départ l'état q_i^x, i étant le numéro de ligne suivant <u>début</u>, x étant la valeur du contenu de a.

- C -

REMARQUES

I LA COMMUTATION

Les méthodes que nous avons mises en oeuvre, pour compiler les langages M.T. en M.A.C. et réciproquement, ont permis de mettre en évidence le lien direct existant entre la notion de transition d'état à état (M.T.) et celle de commutation (M.A.C.). Nous avons pu rapprocher également la notion d'état actif (M.T.) à un instant donné de celle de ligne de programme active au même instant.
Observons toutefois qu'il n'est pas possible de confondre entièrement ces deux aspects: l'état actif (M.T.) indique à la fois, la ligne de programme (M.A.C.) active et la valeur courante de a; à l'inverse la ligne de programme ne peut à elle seule, carctériser l'état interne de l'automate, qui n'est parfaitement défini que lorsque le contenu de a est connu.

Du point de vue de l'architecture des machines le langage M.T. n'opère de distinction qu'entre la file illimitée et le reste de l'automate; dans l'état sont cristallisées toutes les indications relatives à la structure interne: actions à exercer, mémorisations, phase précise du séquencement.
A l'opposé le langage M.A.C. indique distinctement, par le contenu des lignes de programme les actions de traitement, par la disposition de ces lignes le séquencement; enfin l'expréssion de la mémorisation est traduite de façon explicite dans la structure même de la machine qui est munie d'une mémoire centrale.

II COMMUTATION ET CALCULABILITE

Dans un article connu(i), Shannon donne un procédé de construction d'une machine de Turing universelle à seulement 2 états; en fait le procédé mis en oeuvre, permet de ramener n'importe quelle machine de Turing (en particulier universelle), à une machine à 2 états au prix d'un certain accroissement de l'alphabet initial.
(i):cf bibliographie

Il est intéressant de considérer ces résultats transposés dans le langage M.A.C.. Au lecteur soucieux du détail, nous ne pouvons que suggérer le renvoi à l'article évoqué ci-dessus, qu'il n'est pas possible de reprendre ici.
Nous nous contenterons de montrer comment une forme standard peut être donnée à tout programme M.A.C.:

Partant d'un programme M.A.C. quelconque, nous appliquons le procédé de réduction qui a été exposé dans notre première partie;

puis nous compilons en langage M.T, suivant la méthode développée dans la seconde partie;

on applique alors le procédé de Shannon, qui ramène cette machine à une machine à 2 états;

enfin, on compile en langage M.A.C. le résultat obtenu.
Au total on obtient une forme des programmes qui est la suivante:

```
E: af
   Si Ca=0 ou 1 ou 2 ou 3 ou.....ou n
      Vers e0 ou e1 ou e2 ou.....ou en

e0:.....

e1:.....

.............

.............

ei:Ca:=j
   fa
   Av (ou Ar)
   Vers E (ou Vers E')

.............

.............

en:.....
```

```
E': af
    Si Ca=0 ou 1 ou 2 ou3 ou.....ou n
       Vers e'0 ou e'1 ou e'2 ou.....ou e'n
e'0:.....

e'1:.....

.....

.....

e'i,: Ca:=j'
    fa
    Av (ou Ar)
    Vers E (ou Vers E')

.....

.....

e'n:.....
```

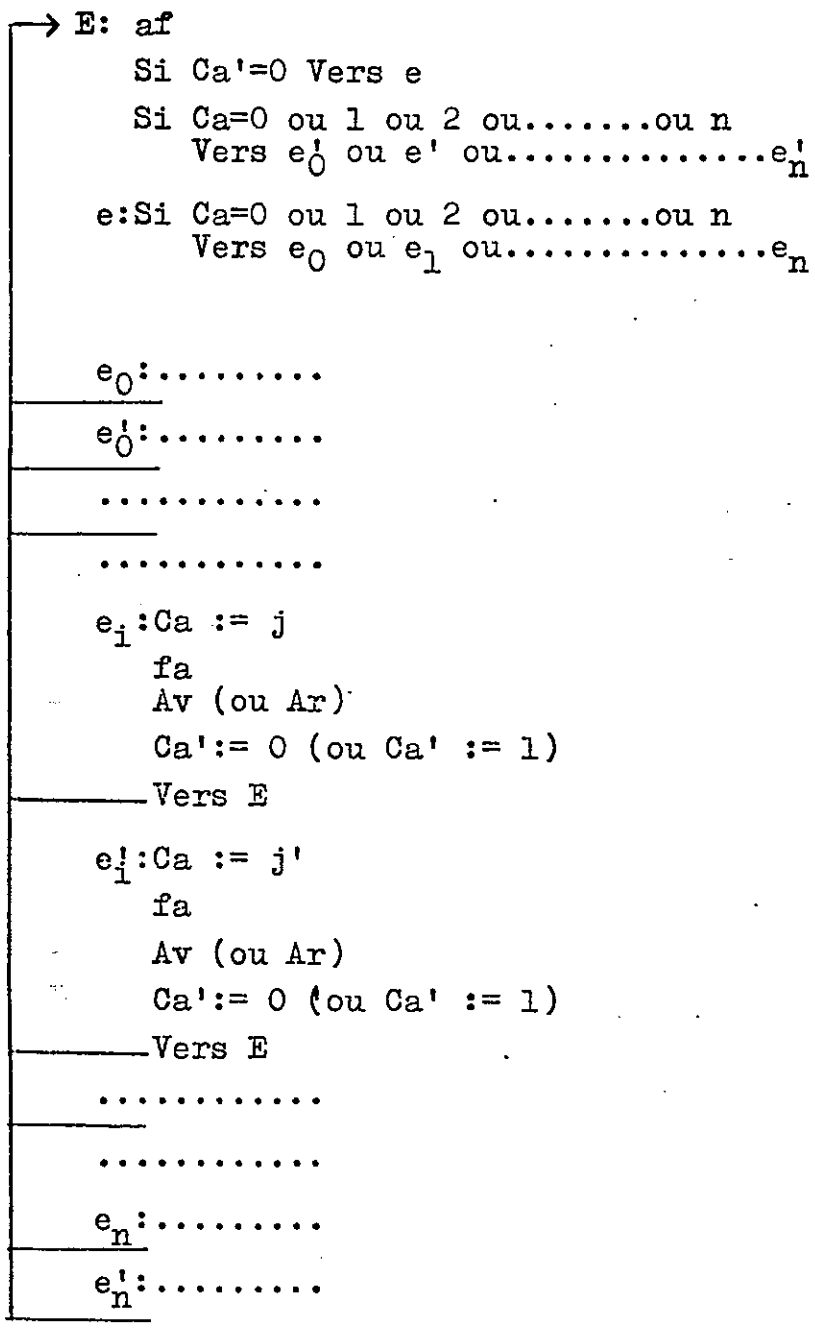
Dans la machine ainsi déterminée, 0 et 1 représentent par exemple les deux uniques symboles de la M.A.C. réduite sous la forme F_3, et les autres symboles 2,3,....,n sont ceux qu'il faut adjoindre à l'alphabet pour se ramener à 2 états avec la méthode de Shannon.
Si on considère alors cette forme standard des programmes, on constate qu'elle se caractérise par les éléments suivants:

- on n'entre dans une phase de calcul que par E ou E';
- La 1ère instruction d'une phase consiste à charger depuis la file la seule case de mémoire centrale;
- Cette case est alors examinée par le moyen d'un test multiple, qui détermine la suite du calcul par l'envoi à l'étiquette correspondante;
- A chacune de ces étiquettes est associée une phase particulière du calcul au cours de laquelle on écrit sur la file, on se déplace, et on entre dans une nouvelle phase par E ou par E'.

Observons dans un premier temps que la machine qu'il faut mettre en oeuvre est tout de même assez évoluée: structure de test multiple, commutations très nombreuses. Comme on l'avait déjà signalé plus haut, il n'est pas possible de confondre l'état actif (M.T.) et la ligne de programme active (M.A.C.), et de ce fait, il est sans espoir d'espérer définir une forme standard des programmes en M.A.C., n'utilisant que deux étiquettes.
Dans un second temps, on remarque que les étiquettes E et E' jouent un rôle particulier: lorsqu'on entre dans l'une ou l'autre de ces lignes de programme, l'état interne de la machine (c'est à dire le contenu de a) n'a aucune importance, puisque la première action qui s'exerce alors est précisément de charger a; de ce fait l'information antérieure est définitivement perdue et ne peut donc plus rien conditionner dans la suite du calcul.
Ceci met clairement en évidence le rôle de E et E' sur le plan de la mémorisation: puisqu'aucun état antérieur de la mémoire n'est conservé, la seule influence que les calculs précédents peuvent avoir sur le déroulement ultérieur, réside précisément dans l'alternative E ou E' qui termine chaque phase de calcul. Ainsi, ce type de commutation liée à la considération des données externes (le choix E ou E' est arrêté dès que l'action af a eu lieu), est associé à la mémorisation interne indispensable qui <u>sépare deux phases</u> de calcul.

De même qu'on établit l'impossibilité d'une M.T. universelle à 1 état, et plus généralement l'impossibilité de ramener une M.T. quelconque (ayant plus d'un état) à une M.T. à 1 état, on montre qu'une structure générale des programmes M.A.C. du genre précédent ne possédant qu'une étiquette du type E, est hors d'atteinte: une telle structure serait de fait sans mémoire inter-phases.
On peut noter au passage qu'une structure à une seule étiquette de type E, pourrait s'obtenir dès que la mémoire centrale contiendrait une case de plus, et qu'on adjoindrait un test supplémentaire; la forme générale des programmes serait alors la suivante:

a' est la mémoire supplémentaire; elle n'a besoin que de contenir 2 valeurs 0 ou 1.



Bibliographie: lère partie

(1) Minsky.M. Computation. Finite and infinite machines
Prentice-Hall. 1967.

(2) Nolin.L. Formalisation des notions de machine et de programme. Gauthiers-Villars. 1969.

Bianco.E. Informatique fondamentale.
Birkhauser. 1979.

2ème partie

- Les références ci-dessus
- Shannon.C. Automata Studies (p.157)
Princeton University Press.1956.

VOUZZAVEDIBISAR.

Citations.

Toujours est-il que c'est dans ce climat de dictature qu'il faut replacer l'évolution des mouvements littéraires et de la langue elle-même au XVII[e] siècle. Le « génie de Richelieu » est d'avoir compris qu'il lui fallait non seulement imposer ses lois par la force, mais également en se rendant maître des hommes qui pensent et écrivent. Car le vent de la subversion ne soufflait pas seulement chez les croquants du Limousin. Il était nécessaire de protéger les dévots contre les libertins, et, d'une façon générale, de « gagner les cœurs et les esprits » à l'idéologie nouvelle.

Du reste, dans cette glorieuse entreprise de « protection des arts », le cardinal avait des alliés de choix. En effet, Malherbe était venu !...

Malherbe est arrivé à Paris en 1605. Il avait cinquante ans. Cet homme, aigri par les revers, parvient enfin à force d'intrigues et de flatteries à s'introduire solidement auprès du roi, prêt à toutes les servitudes. Il aide Henri IV vieillissant à tourner des vers pour ses maîtresses et devient poète officiel inébranlable. Après avoir longtemps imité Ronsard — sans l'atteindre — ce rimeur froid et médiocre se charge avec une belle servilité de néophyte de l'éloge des grands et de leur politique. L'ordre du jour était-il à la remise en ordre d'un royaume déchiré ?... Malherbe se chargea de mettre de l'ordre aussi dans la langue française. Il avait mis du temps à s'installer à la cour, il entendait y rester.

Donc, à l'élan vital des poètes de la Renaissance qui évoque peut-être encore trop les troubles de la Réforme, il oppose une doctrine autoritaire qui s'appuie sur la pensée logique, la rigueur grammaticale, l'épuration. Juriste de formation, il formule le code, édicte les règles. En 1610, l'Académie de l'art poétique définit ce que devra être la langue du XVII[e] siècle. Malherbe décide l'exclusion des influences des langues régionales sur la langue française qu'il veut pure (ça rappelle des choses : il aurait pu dire aryenne tant qu'à faire). L'exclusion du parler du peuple et de tous les vocabulaires techniques qui ne sont pas ceux de la cour et de la haute société parisienne (tiens, tiens !...). Il érige en loi la langue des gens

Parler croquant

Alors ? Eh bien ! comme disait déjà Montaigne, « que le gascon y aille si le français n'y peut aller » !

Des noms ou des verbes ?

Cette caractéristique, qui peut paraître anodine à un lecteur non averti, est un fait extrêmement important en ce sens qu'elle illustre un trait fondamental du français qui tend à donner l'idée plutôt que les faits, ou encore à interpréter le réel. C'est le processus même de l'abstraction, qui rejoint la tendance à employer le mot-idée de préférence au mot-image, et c'est pourquoi l'on dit, à juste titre, que le français est une langue abstraite.

Cela, signifie-t-il, encore une fois, que cette « intellectualisation » du langage provient d'une tradition naturelle, commune à l'ensemble des Français, dont le cerveau porterait, en naissant, la « bosse de l'abstraction » ? C'est douteux. Je ne pense pas en tout cas que la masse de la population occitane naisse ainsi. En effet, l'occitan — bien que langue romane, influencé par le français au cours du dernier siècle — est encore bien plus comparable à l'anglais dans ce domaine. Je n'hésite pas à dire que le génie de l'occitan préfère le verbe, le réel, à sa transposition nominale. Il dira : « *As the troops marched by* » (« quand les soldats passeron »), et non pas « au passage des troupes ».

Si un raisonnement se poursuit trop longtemps sur le plan théorique et n'est pas rattaché à la réalité de la vie par des mots de tous les jours, des images solides qui parlent d'elles-mêmes, l'Anglais a l'impression qu'on le promène dans des sphères trompeuses et qu'on cherche à lui faire croire n'importe quoi. Il a une réaction de bon sens paysan : il se méfie beaucoup des notions incontrôlables et se lasse vite à demeurer toujours sur le plan de l'entendement. D'où la nécessité pour un auteur d'animer son récit, de l'amarrer par un style quelquefois à ras de terre, surtout s'il s'adresse en principe à un vaste public. Car la crainte d'être ridicule en anglais, c'est la crainte d'être pédant. Il faut convenir que c'est souvent l'inverse en français : on craint d'être ridicule en utilisant un langage trop clair qui est compris de tout le monde. Plus on est abstrait, pédant, et donc plus on s'adresse à une élite, plus on a de chances d'être pris au sérieux. C'est là l'influence durable de l'évolution historique de nos deux pays — une langue profondément démocratique d'une part, une langue profondément aristocratique de l'autre. Ou bien est-ce que je devrais employer le mot « génie » ? C'est ça ! Le génie du français, c'est de se croire encore à la cour de France !

Le français, dit-on, préfère le substantif (c'est-à-dire le nom) au verbe. Par exemple — et là encore je suis l'excellente analyse de Vinay-Darbelnet — le français n'emploie pas « dès qu'il arrivera », mais « dès son arrivée » — là où l'anglais utilise seulement la tournure verbale « *as soon as he arrives.* » De même, on ne dira pas : « Les gens ont applaudi lorsque les troupes ont passé », mais plutôt : «... ont applaudi sur le passage des troupes » — pour « *people cheered as the troops marched by* ». Or, traduire une action — arriver, passer — par un nom, une tournure nominale — son arrivée, le passage — au lieu d'employer un verbe, est une attitude de l'esprit qui tend à « présenter les événements comme des substances ». Cela signifie qu'on ne décrit pas le réel tel qu'il est donné — « les troupes ont passé » — mais qu'on le fixe, qu'on l'analyse pour en communiquer l'essentiel, le passage. A. Chevrillon, cité par Vinay-Darbelnet, indique : « Le français traduit surtout les formes, états arrêtés, les coupures imposées au réel par l'analyse. »

L'argot, c'est le défoulement d'un peuple étouffé, brimé dans son expression d'une façon incroyable. C'est la joie de parler.

Un néo-patois ?

A l'autre pôle, se développe depuis de nombreuses années, parmi les intellectuels, un langage qui reste, lui, totalement incompréhensible du commun des Français, et qui se caractérise par l'usage systématique de mots savants. Il convient d'être prudent, car les critiques adressées au jargon des philosophes émanent souvent de milieux réactionnaires qui, sous couvert de défendre le beau langage, attaquent le contenu, et pour être amusants défendent en fait leurs valeurs bourgeoises menacées. Pourtant, il n'est pas douteux que certains intellectuels se défoulent dans le jargon de façon nettement outrancière. Il est sûrement nécessaire d'inventer des mots nouveaux pour traduire des concepts nouveaux, et ces vocables ainsi créés ne sauraient être autre chose qu'abstraits. Mais est-il également nécessaire de traduire en mots savants des notions coutumières qui ont déjà une expression dans le langage courant ? Exemples : les occulter (pour cacher), situation conflictuelle (pour tension), excentré (pour éloigné), etc., sont-ils de la plus haute urgence linguistique ? Tous les belligérances, irrationnalités, immédiatéités, historicismes qui aboutissent à cette langue que R. Beauvais a appelé « l'hexagonal » ?

« Redonner une vie aux régions avant toute chose, c'est rendre aux habitants la confiance en eux. »
devient :

Grand serait notre plaisir d'en citer davantage, mais manque de place oblige et plaisir de découvrir pour le lecteur également.

Nous avons entra-perçu,lors de notre dernière rubrique combien le Novlangue,qui ne nous était pourtant pas totalement étranger ,seyait à l'Angsoc.On pourrait bien découvrir cette fois-ci combien le Français sied bien au Fransoc ou même au Franlib,pour ceux qui excellent en nuances.

Je propose un concours,comme la dernière fois,il faut deviner le titre et l'auteur de l'ouvrage cité.Le plaisir de lire sera une ample récompense à la sagacité des trouveurs.

La Parole c'est comme les idées: ça ne fait pas couler le Pétrôle.