N° D'ordre : 226

THÈSE

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR INGENIEUR

(traitement de l'information)

par

PATRICK VAN LAER

Ingénieur I.S.E.N.

ETUDE D'UNE ARCHITECTURE A FLUX SIMULTANÉS

Soutenue le 25 avril 1979 devant la Commission d'Examen

Membres du Jury :

MM.	C. CARREZ	Président
	L. CARREZ	Examinateur
	R. CHEVENCE	Examinateur
	G. WERNER	Examinateur
	V. CORDONNIER	Rapporteur

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE

01.02.1976

DOYENS HONORAIRES de l'Ancienne Faculté des Sciences

MM. R. DEFRETIN, H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES des Anciennes Facultés de Droit et Sciences Economiques, des Sciences et des Lettres

M. ARNOULT, Mme BEAUJEU, MM. BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, CORSIN, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, P. GERMAIN, HEIM DE BALSAC, HOCQUETTE, KAMPE DE FERIET, KOUGANOFF, LAMOTTE, LASSERRE, LELONG, Mme LELONG, MM. LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, NORMANT, PEREZ, ROIG, ROSEAU, ROUBINE, ROUELLE, SAVART, WATERLOT, WIEMAN, ZAMANSKI.

PRESIDENTS HONORAIRES DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. R. DEFRETIN, M. PARREAU.

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

M. J. LOMBARD.

PROFESSEURS TITULAIRES

M.	BACCHUS Pierre	Astronomie
M.	BEAUFILS Jean-Pierre	Chimie Physique
M.	BECART Maurice	Physique Atomique et Moléculaire
M.	BILLARD Jean	Physique du Solide
M.	BIAYS Pierre	Géographie
M.	BONNEMAN Pierre	Chimie Appliquée
M.	BONNOT Ernest	Biologie Végétale
M.	BONTE Antoine	Géologie Appliquée
M.	BOUGHON Pierre	Algèbre
M.	BOURIQUET Robert	Biologie Végétale
M.	CELET Paul	Géologie Générale
M.	CONSTANT Eugène	Electronique
M.	DECUYPER Marcel	Géométrie
M.	DELATTRE Charles	Géologie Générale
M.	DELHAYE Michel	Chimie Physique
M.	DERCOURT Michel	Géologie Générale
M.	DURCHON Maurice	Biologie Expérimentale
M.	FAURE Robert	Mécanique
M.	FOURET René	Physique du Solide
M.	GABILLARD Robert	Electronique
M.	GLACET Charles	Chimie Organique
M.	GONTIER Gérard	Mécanique
M.	GRUSON Laurent	Algèbre
M.	GUILLAUME Jean	Microbiologie
M.	HEUBEL Joseph	Chimie Minérale
M.	LABLACHE-COMBIER Alain	Chimie Organique
M.	LANSRAUX Guy	Physique Atomique et Moléculaire
M.	LAVEINE Jean-Pierre	Paléontologie
M.	LEBRUN André	Electronique
M.	LEHMANN Daniel	Géométrie

.../...

Mme	LENOBLE Jacqueline	Physique Atomique et Moléculaire
M.	LINDER Robert	Biologie et Physiologie Végétales
M.	LOMBARD Jacques	Sociologie
M.	LOUCHEUX Claude	Chimie Physique
M.	LUCQUIN Michel	Chimie Physique
M.	MAILLET Pierre	Sciences Economiques
M.	MONTARIOL Frédéric	Chimie Appliquée
M.	MONTREUIL Jean	Biochimie
M.	PARREAU Michel	Analyse
M.	POUZET Pierre	Analyse Numérique
M.	PROUVOST Jean	Minéralogie
M.	SALMER Georges	Electronique
M.	SCHILTZ René	Physique Atomique et Moléculaire
Mme	SCHWARTZ Marie-Hélène	Géométrie
M.	SEGUIER Guy	Electrotechnique
M.	TILLIEU Jacques	Physique Théorique
M.	TRIDOT Gabriel	Chimie Appliquée
M.	VIDAL Pierre	Automatique
M.	VIVIER Emile	Biologie Cellulaire
M.	WERTHEIMER Raymond	Physique Atomique et Moléculaire
M.	ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS SANS CHAIRE

M.	BELLET Jean	Physique Atomique et Moléculaire
M.	BODARD Marcel	Biologie Végétale
M.	BOILLET Pierre	Physique Atomique et Moléculaire
M.	BOILLY Bénoni	Biologie Animale
M.	BRIDOUX Michel	Chimie Physique
M.	CAPURON Alfred	Biologie Animale
M.	CORTOIS Jean	Physique Nucléaire et Corpusculaire
M.	DEBOURSE Jean-Pierre	Gestion des entreprises
M.	DEPREZ Gilbert	Physique Théorique
M.	DEVRAINNE Pierre	Chimie Minérale
M.	GOUDMAND Pierre	Chimie Physique
M.	GUILBAULT Pierre	Physiologie Animale
M.	LACOSTE Louis	Biologie Végétale
Mme	LEHMANN Josiane	Analyse
M.	LENTACKER Firmin	Géographie
M.	LOUAGE Francis	Electronique
Mle	MARQUET Simone	Probabilités
M.	MIGEON Michel	Chimie Physique
M.	MONTEL Marc	Physique du Solide
M.	PANET Marius	Electrotechnique
M.	RACZY Ladislas	Electronique
M.	ROUSSEAU Jean-Paul	Physiologie Animale
M.	SLIWA Henri	Chimie Organique

MAITRES DE CONFERENCES (et chargés d'Enseignement)

M.	ADAM Michel	Sciences Economiques
M.	ANTOINE Philippe	Analyse
M.	BART André	Biologie Animale
M.	BEGUIN Paul	Mécanique
M.	BKOUCHE Rudolphe	Algèbre
M.	BONNELLE Jean-Pierre	Chimie
M.	BONNEMAIN Jean-Louis	Biologie Végétale
M.	BOSCQ Denis	Probabilités
M.	BREZINSKI Claude	Analyse Numérique
M.	BRUYELLE Pierre	Géographie

.../...

M. CARREZ Christian	Informatique
M. CORDONNIER Vincent	Informatique
M. COQUERY Jean-Marie	Psycho-Physiologie
Mle DACHARRY Monique	Géographie
M. DEBENEST Jean	Sciences Economiques
M. DEBRABANT Pierre	Géologie Appliquée
M. DE PARIS Jean-Claude	Mathématiques
M. DHAINAUT André	Biologie Animale
M. DELAUNAY Jean-Claude	Sciences Economiques
M. DERIEUX Jean-Claude	Microbiologie
M. DOUKHAN Jean-Claude	Physique du Solide
M. DUBOIS Henri	Physique
M. DYMENT Arthur	Mécanique
M. ESCAIG Bertrand	Physique du Solide
Me EVRARD Micheline	Chimie Appliquée
M. FONTAINE Jacques-Marie	Electronique
M. FOURNET Bernard	Biochimie
M. FORELICH Daniel	Chimie Physique
M. GAMBLIN André	Géographie
M. GOBLOT Rémi	Algèbre
M. GOSSELIN Gabriel	Sociologie
M. GRANELLE Jean-Jacques	Sciences Economiques
M. GUILLAUME Henri	Sciences Economiques
M. HECTOR Joseph	Géométrie
M. JACOB Gérard	Informatique
M. JOURNEL Gérard	Physique Atomique et Moléculaire
Mle KOSMAN Yvette	Géométrie
M. KREMBEL Jean	Biochimie
M. LAURENT François	Automatique
Mle LEGRAND Denise	Algèbre
Mle LEGRAND Solange	Algèbre
M. LEROY Jean-Marie	Chimie Appliquée
M. LEROY Yves	Electronique
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique Théorique
M. LOUCHET Pierre	Sciences de l'Education
M. MACKE Bruno	Physique
M. MAHIEU Jean-Marie	Physique Atomique et Moléculaire
Me N'GUYEN VAN CHI Régine	Géographie
M. MAIZIERES Christian	Automatique
M. MALAUSSENA Jean-Louis	Sciences Economiques
M. MESSELYN Jean	Physique Atomique et Moléculaire
M. MONTUELLE Bernard	Biologique Appliquée
M. NICOLE Jacques	Chimie Appliquée
M. PAQUET Jacques	Géologie Générale
M. PARSY Fernand	Mécanique
M. PECQUE Marcel	Chimie Physique
M. PERROT Pierre	Chimie Appliquée
M. PERTUZON Emile	Physiologie Animale
M. PONSOLLE Louis	Chimie Physique
M. POVY Lucien	Automatique
M. RICHARD Alain	Biologie
M. ROGALSKI Marc	Analyse
M. ROY Jean-Claude	Psycho-Physiologie
M. SIMON Michel	Sociologie
M. SOMME Jean	Géographie
Mle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. STEEN Jean-Pierre	Informatique

.../...

M.	THERY Pierre	Electronique
M.	TOULOTTE Jean-Marc	Automatique
M.	TREANTON Jean-René	Sociologie
M.	VANDORPE Bernard	Chimie Minérale
M.	VILLETTE Michel	Mécanique
M.	WALLART Francis	Chimie
M.	WERNIER Georges	Informatique
M.	WATERLOT Michel	Géologie Générale
Mme	ZINN-JUSTIN Nicole	Algèbre

Je remercie

Monsieur C. Carrez, qui a accepté de présider le jury de cette thèse,

Messieurs L. Carrez, R. Chevence et G. Werner qui se sont intéressé à ce travail et ont accepté de participer au jury de cette thèse,

Monsieur le Professeur V. Cordonnier, qui m'a proposé ce travail et m'a guidé de ses conseils,

Les chercheurs et les Techniciens du Laboratoire, en particulier Monsieur B. Toursel, pour les nombreuses discussions que nous avons eues et Monsieur P. Behague pour les travaux de programmation qu'il a bien voulu réaliser,

Françoise et Bénédicte, qui ont assuré avec compétence et gentillesse la dactylographie de cette thèse,

Monsieur et Madame Debock, qui ont assuré la réalisation matérielle de ce document dans les meilleures conditions.

A Anne

A Elise

TABLE DES MATIERES

CHAPITRE I

LES MÉMOIRES CIRCULANTES

INTRODUCTION

L'apparition de nouvelles technologies de mémoires, que nous regrouperons sous le terme de mémoires circulantes, permet d'envisager des améliorations, ou même des inovations importantes dans l'architecture des ordinateurs. Ces mémoires se situent, aux points de vues prix et performances, entre les mémoires magnétiques (disques et bandes) et les mémoires électroniques à accès aléatoire. Elles ont comme les premières un accès séquentiel, mais leur technologie et leurs performances les rapprochent des secondes. Elles comblent le vide considérable caractérisé par un facteur d'échelle de l'ordre de 10^4 entre les vitesse électroniques (10^{-6}) et les vitesses mécaniques (10^{-2}).

Dans une première partie, on présentera les technologies de ces mémoires : on étudiera ensuite leurs applications présentées et l'état des recherches les concernant.

I.1 TECHNOLOGIE DES MEMOIRES CIRCULANTES

I.1.1

Les mémoires CCD

Les mémoires CCD (Charge Coupled Device, Dispositifs à Transfert de Charge) sont des circuits électroniques qui transférent de l'information sous forme d'une charge électrique le long d'une rangée de condensateurs MOS [BOR 77].

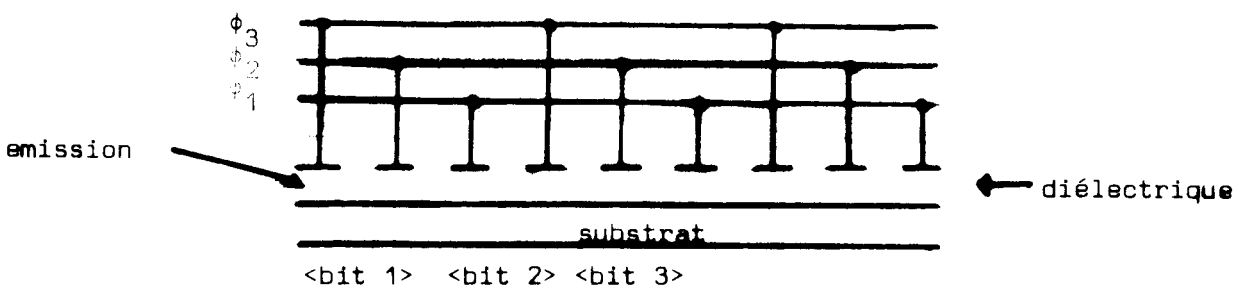


Fig. I.1 Transfert de la charge dans une mémoire CCD

Les bits '1' ou '0' sont définis par la présence ou l'absence d'une charge créée au point d'émission. Cette charge se déplace en faisant varier convenablement les potentiels des phases (exemple à 3 phases fig I.1). La réception se fait en détectant la charge (amplification). La longueur de la ligne est limitée par la dégradation du signal par perte de la charge (500 à 1000 bits) : pour la même raison une vitesse minimale de transmission est nécessaire (fonctionnement dynamique) : la fréquence maximale est imposée par la vitesse de déplacement des charges en surface et par la puissance dissipée qui augmente avec la vitesse.

...

En général, l'extrémité de la ligne est rebouclée sur l'entrée ; l'accès se fait donc en ce point, de manière séquentielle. On l'appelle souvent la fenêtre d'accès à la mémoire. Comme, bien entendu, cette fenêtre est fixe, il n'est plus question de définir un contrôle de l'accès par sélection géométrique de la fenêtre.

Cette technologie est du même type que les autres mémoires MOS. Elle se distingue des mémoires RAM dynamique, où l'information est elle aussi stockée sur un condensateur (condensateur parasite d'un transistor MOS) par l'absence d'accès aléatoire, donc économie de surface (rapport 1:2 actuellement) et de consommation (rapport 1:3).

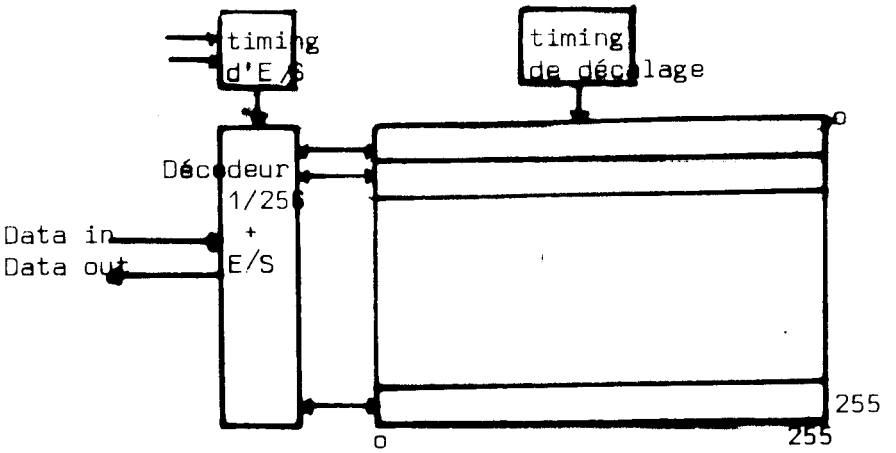
On appellera 'piste' l'ensemble des positions binaires de stockage formant une boucle et disposant d'une fenêtre. Du fait de la limitation de la longueur d'une piste, et de la vitesse maximale de circulation, on est conduit a grouper plusieurs pistes sur un même chip [ALT 78] [ARM 77]. Une des solutions extrêmes consiste à considérer le chip comme un tambour (fig I.2).

Les pistes sont courtes, et la vitesse de rotation est relativement lente ; on accède à une des pistes aléatoirement (ex : Intel 2464, vitesse : 2.5 MHZ, temps moyen d'attente : 125 µs). Cette organisation est orientée vers l'utilisation de la mémoire en temps que produit de remplacement économique des RAM.

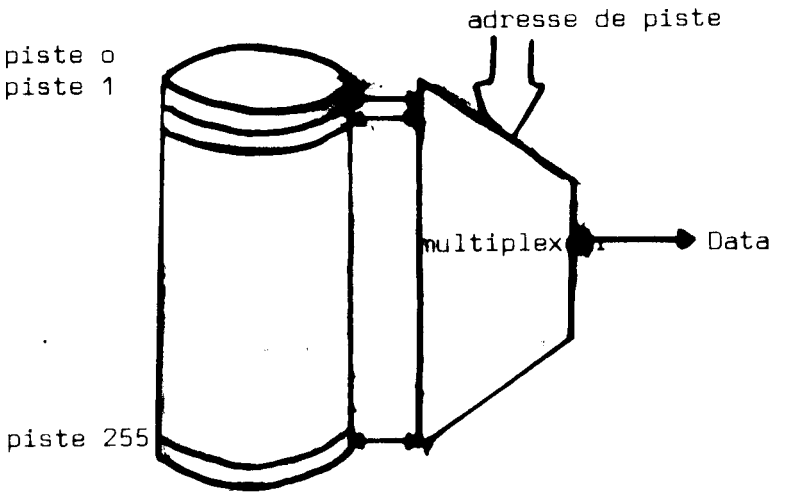
A l'autre extrême, on conserve l'aspect séquentiel de la mémoire, en cherchant à augmenter le débit grâce à l'organisation série-parallèle-série (fig I.3.a)) [THR 6/78]. C'est la formule adoptée par TI (TMS 3064) et par Fairchild (F464).

Les registres à décalage A et B sont très rapides ils permettent de faire fonctionner l'ensemble à 5 MHZ alors que les 64 registres de 64 bits ne fonctionnent qu'à 156 KHZ ;Vu de l'extérieur, il s'agit donc d'un seul registre de 4 K bits plus rapide.

...

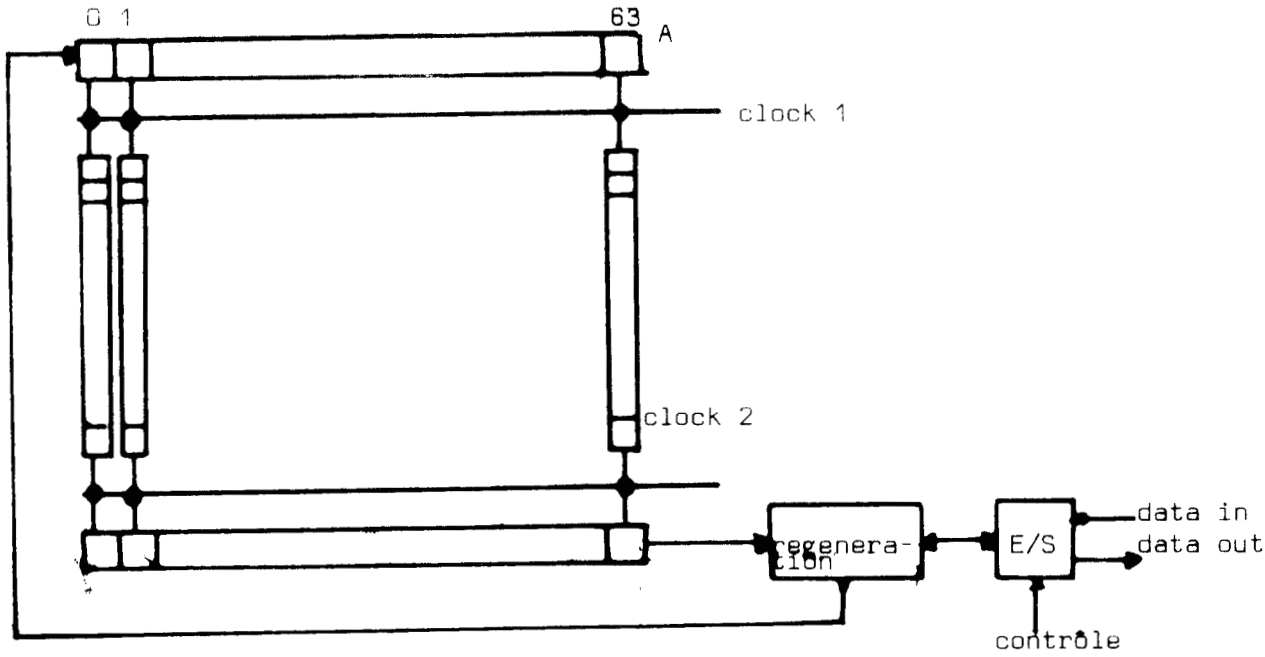


Configuration de la mémoire

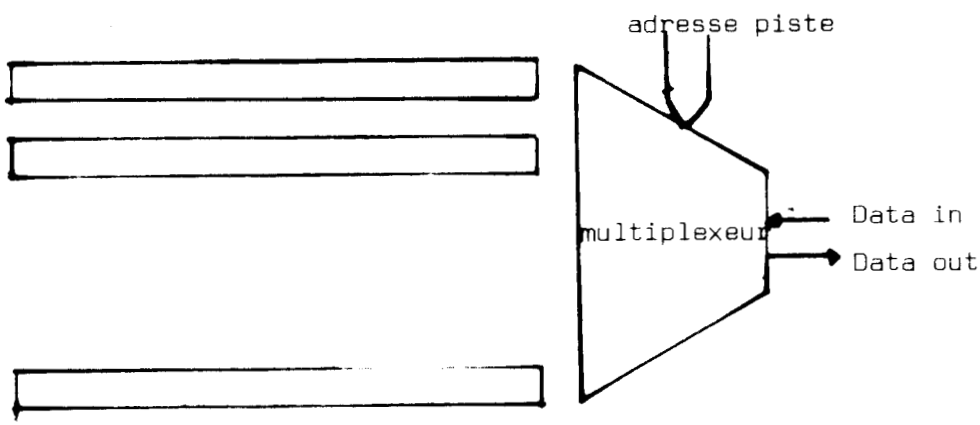


analogie avec les tambours magnétiques

Fig. I.2 Mémoire CCD à pistes courtes



a) Organisation série-parallèle série créant des boucles longues



b) Multiplexage des boucles longues

Fig. I.3 Mémoires CCD à boucles longues

Dans ce type de chips, on conserve cependant un accès aléatoire en groupant 16 circuits (fig. I.3.b)). La vitesse peut alors atteindre 5 MHZ avec un temps d'accès mayen de 400 µsec.

La première solution est plus intéressante des points de vue temps d'attente et consommation (il est possible de faire varier la vitesse de rotation pour explorer les pistes aléatoirement). La seconde est plus intéressante au point de vue intégration (moins de dispositifs de régénération), et donc au point de vue prix. Elle s'intégrerait mieux à vue architecture d'ordinateur qui ne serait pas une copie de celles qui existent, mais prendrait en compte la nature séquentielle de l'information. Elle deviendra d'ailleurs probablement un standard de l'industrie avec l'abandon récent de la fabrication des I 2464 par Intel.

I.1.2 Les mémoires à bulles magnétiques

I.1.2.1 Les mémoires à bulles 'classiques'

Les mémoires à bulles utilisent la propriété de certains matériaux, hétérogène du point de vue magnétique. Ceci permet la création de "bulles magnétiques" microscopiques [MAR 77]. Celles-ci se déplacent sur le substrat, le long de "chemins" constitués par un réseau ferromagnétique (permalloy en général)

Il existe différentes formes de réseaux : TI-bar, Y-bar, disques contigus, chevrons et croissants [TI 2/77], [RAN 9/77]. Les bulles sont créées ou détruites par l'action d'un champs magnétique convenablement appliqué. La lecture se fait en augmentant la taille de la bulle par un dessin particulier, en détectant sa présence par magnétorésistance (variation de l'ordre de 1 %), puis en amplifiant. Le fonctionnement d'une piste est purement statique : on peut ralentir et même arrêter le déplacement des bulles à condition de les laisser dans une position stable (position 4 dans la fig. I.4).

...

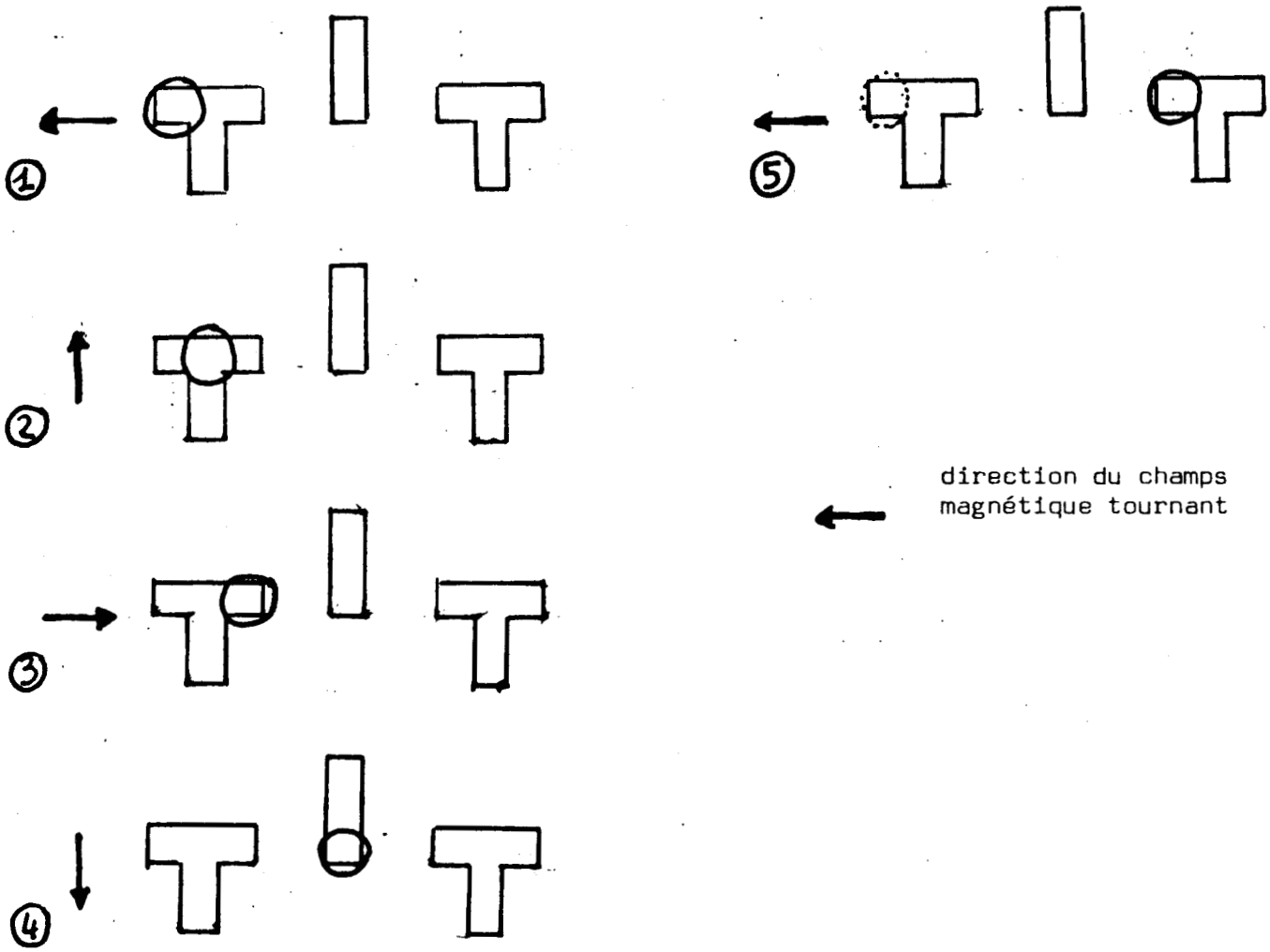


Fig. I.4 Déplacement d'une bulle magnétique dans un réseau TI-bar

La consommation est proportionnelle à la vitesse (nulle à l'arrêt). La longueur des pistes n'est pas limitée : il existe par exemple une réalisation expérimentale de 1 M bits ; mais la difficulté technologique de réaliser un réseau parfait croit comme le carré de la capacité, et le rendement d'une chaine de fabrication limite actuellement les réalisations disponibles à quelques centaines de Kilobits. La vitesse de propagation est faible (5µs par pas).

Pour établir un compromis acceptable entre la capacité et la vitesse, on est conduit à préférer des organisations du type boucle majeure/ boucle mineure (fig. I.5). Parmi les boucles mineures, un certain nombre (10 % environ), testées lors de la production, présentent un défaut et ne doivent donc pas être prises en compte à l'utilisation.

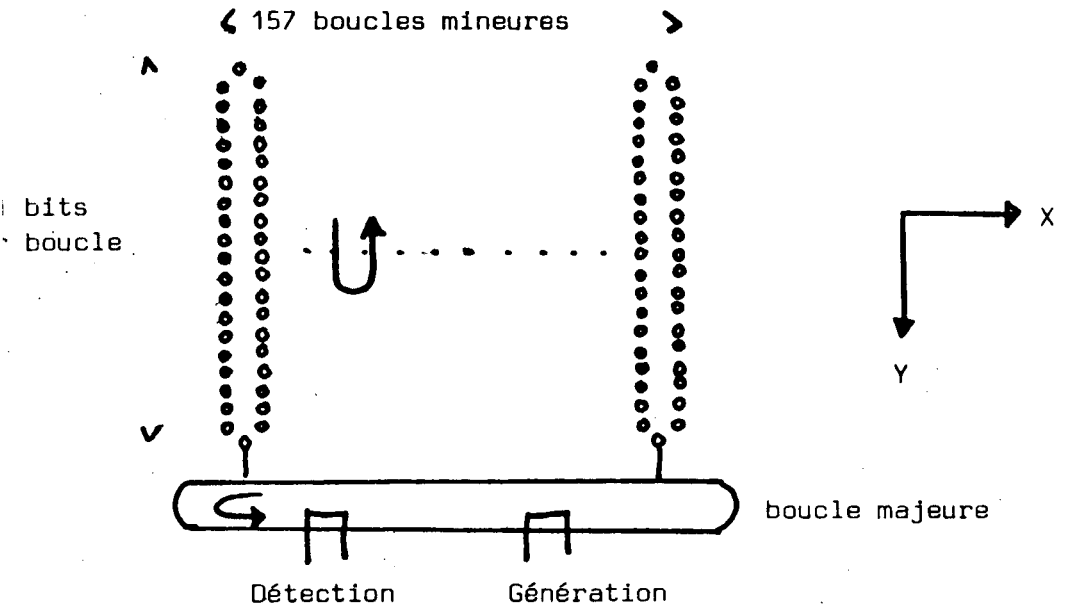


Fig. I.5 Une organisation de mémoire à bulle en boucle majeure/boucle mineure (type TI, T BM 0603)

...

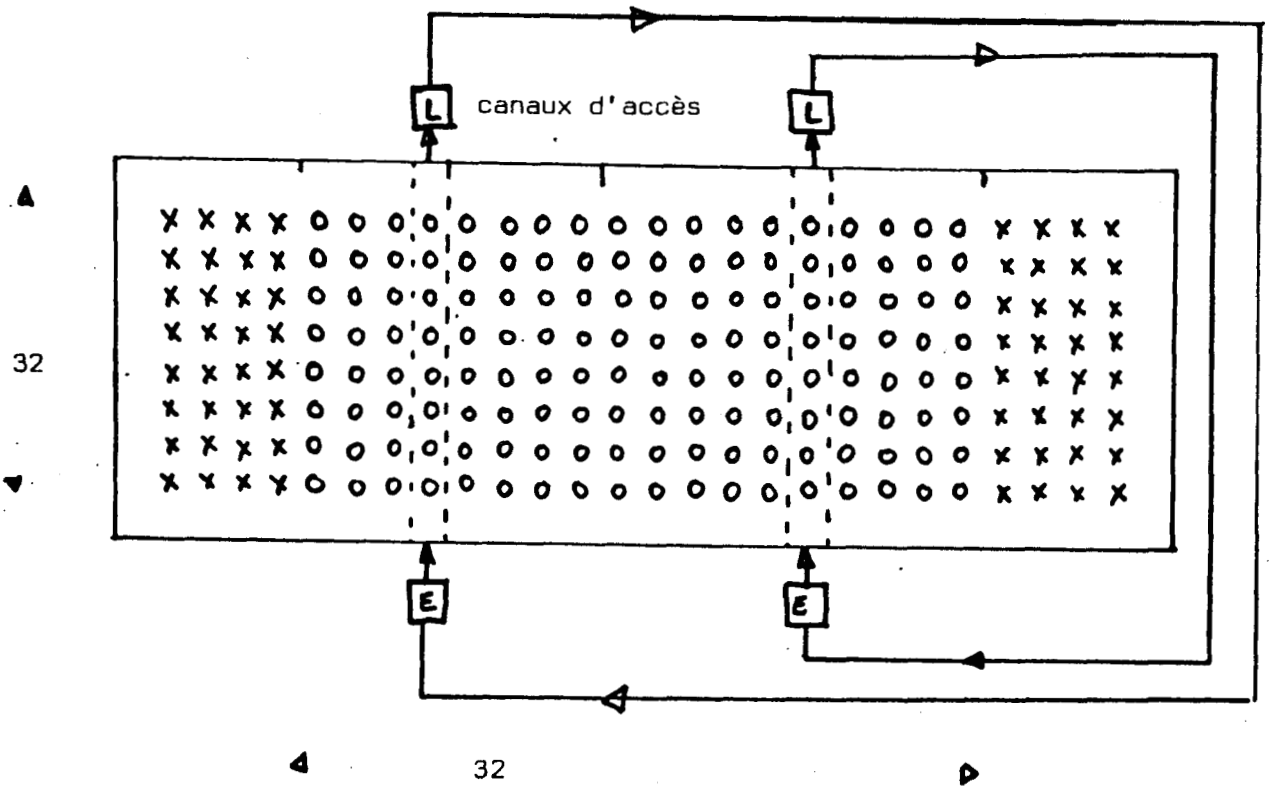
1

I.1.2.2

Les 'Magnetic bubble Lattice Files' (BLF)

Ce type de mémoires à bulles est développé par IBM. Il en est encore au stade de l'expérimentation, puisque seul un chip de 1 K bit a été réalisé jusqu'ici [MAR 4/78].

L'organisation est la suivante (fig. I.6)



X; zone buffer
O; bulle magnétique

Fig. I.6 Organisation d'une mémoire BLF

...

L'information est codée par l'état de surface de la bulle (et non pas par la présence ou l'absence d'une bulle). L'accès à un bit se fait en décalant l'ensemble de la mémoire horizontalement, jusqu'à ce que la colonne désirée se trouve sous une voie d'accès (fixe) ; il est alors possible de lire (et de réécrire) les informations en décalant cette seule colonne jusqu'au bit recherché.

Le premier avantage de ce type de mémoire est la grande densité possible : le fait que toutes les cases contiennent une bulle provoque une grande cohésion de l'ensemble qui permet de réduire la distance entre bulles : 11.2 µ pour des bulles de 5 µ, soit une densité de 1 MB/cm^2. Le second avantage est de ne pas nécessiter de champs magnétique tournant, les deux types de décalages étant réalisés par des conducteurs électriques.

I.1.3 Autres types de mémoires circulantes

I.1.3.1 Les registres à décalage MOS

On distingue les registres à décalages dynamiques, ancêtres des CCD, qui nécessitent plusieurs transistors MOS par bit, ils atteignent donc une densité moindre (2 K bit/chip, 5MHZ), et les registres à décalage statiques, qui sont constitués par une suite de bascules, et peuvent donc être arrêtés sans perte de l'information (1 K bit/chip, 4 MHZ).

I.1.3.2 Les mémoires à propagation de domaines magnétiques

D'un principe voisin de celui des mémoires à bulles, les mémoires à propagation de domaines sont développées par une société française (CROUZET).

Les domaines magnétiques (5 × 30 µ) se déplacent sur une surface de verre, perpendiculairement à des conducteurs qui provoquent les décalages [BAT 77].
Elles sont organisées en 32 blocs de 512 mots de 8 bits par substrat (5 cm × 5 cm) soit 130 K bit.

...

Elles sont comparables aux MBM des points de vue consommation, asynchronisme et non volatilité ; elles ont l'inconvénient d'être beaucoup moins intégrées (15 K bit/cm^2) du fait de la taille des domaines ; mais elles sont plus rapides que les mémoires à bulles, et comparables aux CCD, du point de vue débit (et temps d'accès).

I.2 UTILISATION DES MEMOIRES CIRCULANTES DANS LES ARCHITECTURES D'ORDINATEURS

I.2.1 Utilisation comme niveau intermédiaire dans la hiérarchie des mémoires

Dès l'apparition des mémoires circulantes ("disques électroniques"), il a semblé que celle-ci venaient combler le fossé qui séparait les mémoires électroniques à accès aléatoire des mémoires magnéto-mécaniques (disques et bandes) aux points de vue prix et temps d'accès (cf. fig. I.8). Or, le fonctionnement optimal d'une mémoire hiérarchisée nécessite des rapports de vitesse d'un niveau à l'autre qui ne soient pas trop disparates (chargement des pages avec un débit maximum pour le niveau le plus rapide). De plus, l'aspect séquentiel des mémoires circulantes n'est nullement défavorable, puisque les changements de niveau se font page par page, donc en séquence [**WEN 75**], [**PAN 77**].

Ce type d'application est dès maintenant mis en pratique par Burroughs pour son ordinateur scientifique de haut de gamme (le BSP) de type "array processor", qui contient une mémoire fichier CCD de 4 à 67 M mots à 32 bits [**MON 77**]

I.2.2 Utilisation dans les traitements parallèles

Outre l'utilisation de mémoires circulantes comme niveau de mémoire intermédiaire dans un calculateur parallèle, on peut envisager de les utiliser même par le niveau de mémoire le plus bas (fig. I.10).

...

	Taille du Chip	Coût	Temps d'accès	Débit	Densité	Consommation	Consommation en attente (à l'arrêt)	volatilité	Tensions d'alimentation
	bits	centimes/bit	µs	Mb/sec	Kb/cm^2	µW/bit	µW/bit		V
RAM bipolaires	4 K (1)	10	0,05	20	20	200	20 (1)	OUI	+5
RAM MOS Statiques	4 K	1,5	0,3	3	50	100	10	OUI	+5
RAM MOS Dynamique	16 K	1	0,3	3	90	50	2	OUI	±5,+12
RAM C MOS	2 K	10	0,5	2	20	30	0,05	OUI	+5
ROM	64 K	5	0,5	2	50	30	30 (0)	NON	+5
PROM	16 K	3	0,1	10	20	100	100 (0)	NON	+5
REPROM UV	16 K	7	0,5	2	50	30	10 (0)	NON	+5
EE PROM	4 K	2	2	0,5 (2)	10	100	35 (0)	NON	+5,-12,-30
CCD	64 K	0,1	10^2	5	100	15	1,5	OUI	+12,-5
MBM	92 K	0,1	4.10^3	0,05	300	7 (3)	0 (3)	NON	+5,+12
Domaines	131 K (4)	0,2	450	5	15	50	0,4 (0)	NON	±5,±12

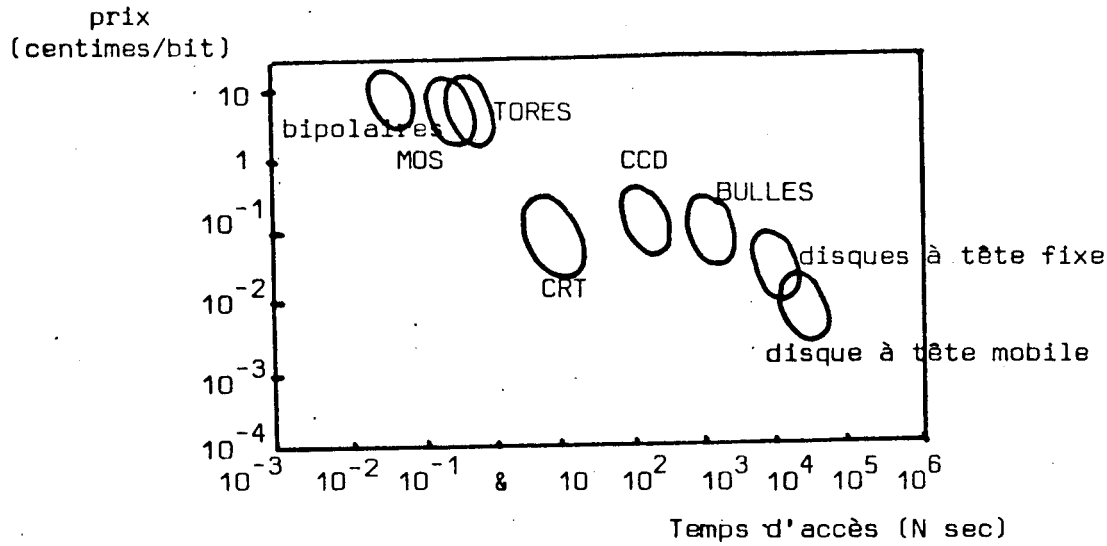
Fig. I.7. Tableau comparatif des mémoires intégrées

(1) Dynamique

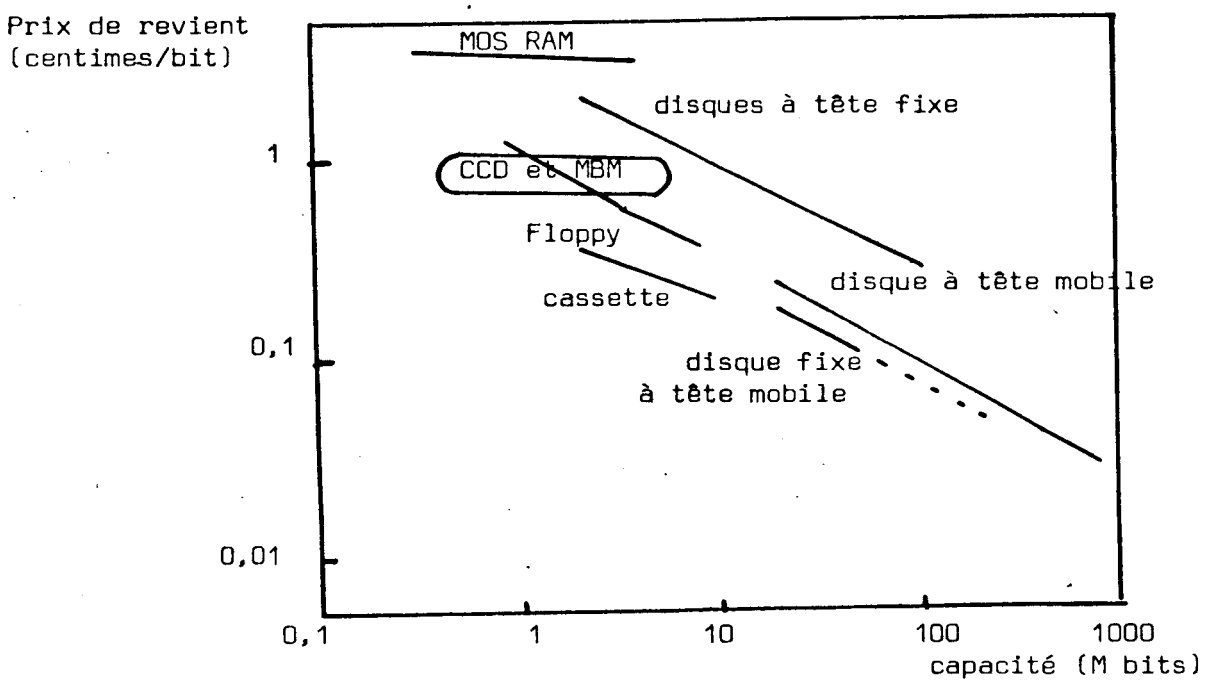
(2) Effacement 10 ms, écriture 1 ms

(3) Interfaces non compris

(4) Pour 1 substrat de 5 cm × 5 cm

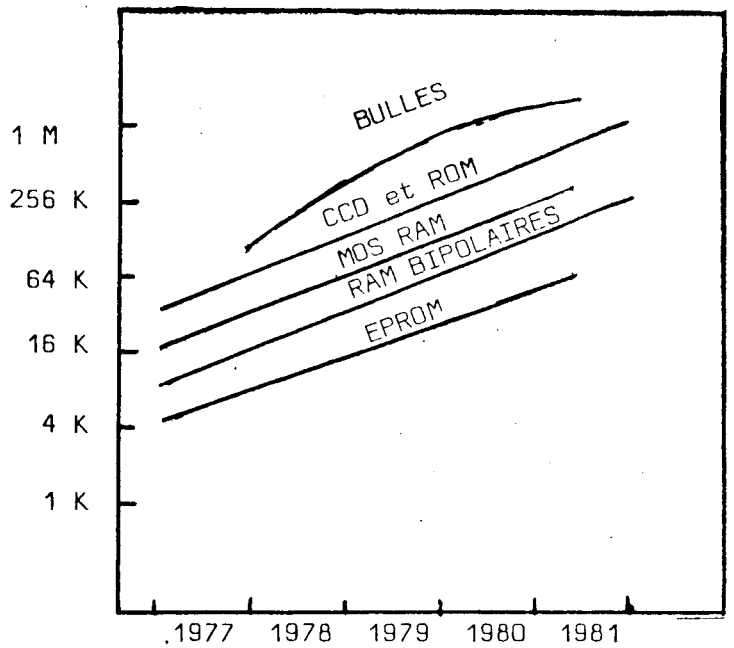


Prix des mémoires en fonction du temps d'accès

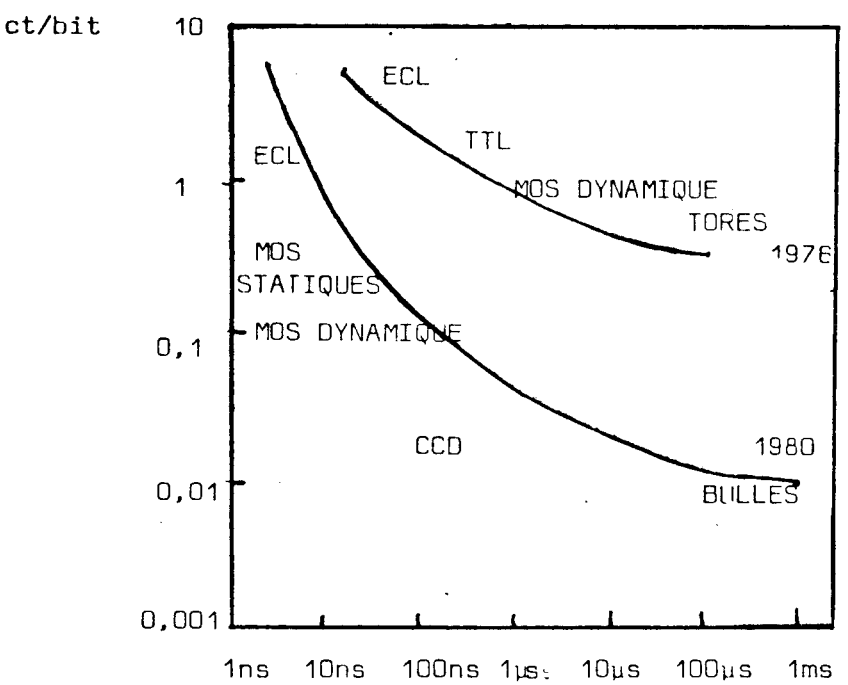


Prix des mémoires (interfaces compris) suivant la capacité

Fig. I.8. Les mémoires circulantes dans la hiérarchie des mémoires.



Densité prévisible des chips pour les années à venir



Coût/bit en fonction du temps d'accès

Fig. I.9 Evolution prévisible des mémoires intégrées dans le temps

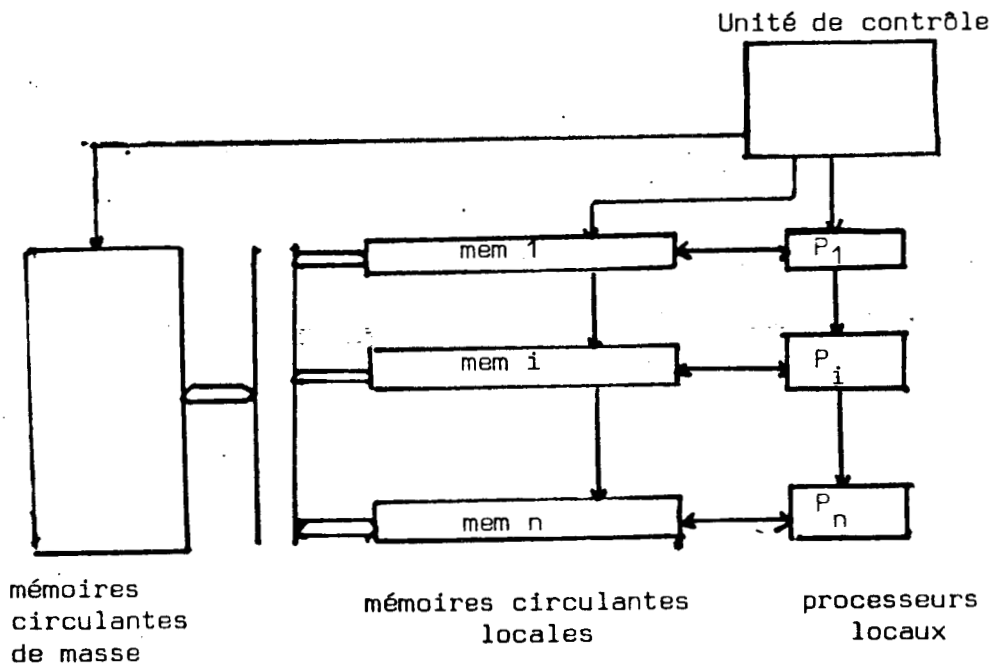


Fig. I.10 Traitement parallèle avec mémoires circulantes

Le chargement des mémoires locales se fait à partir de la mémoire de masse, au débit maximum de chacunes d'elles. Les données à traiter étant du type tableau, une mémoire à accès partiellement aléatoire convient bien (chargement par ligne et/ou par colonnes).

I.2.3 Utilisation d'une mémoire circulante en mémoire associative

Les mémoires associatives parallèles, qui apportent un gain de performance très important dans certains types de traitements, sont peu utilisés du fait de la complexité des circuits ; elles sont donc chères et peu intégrées. L'accès associatif à une mémoire RAM peut donc se faire soit, dans des cas limités, à l'aide d'une circuiterie externe importante (ex : dans STUD), soit par un examen sucessif de chaque mot de la mémoire. Dans le second cas, l'adressage devient inutile, et l'ordre d'examen des mots devient indifférent : il est alors simple et avantageux d'implementer une mémoire à accès associatif, réalisée à l'aide de mémoires circulantes (fig. I.11).

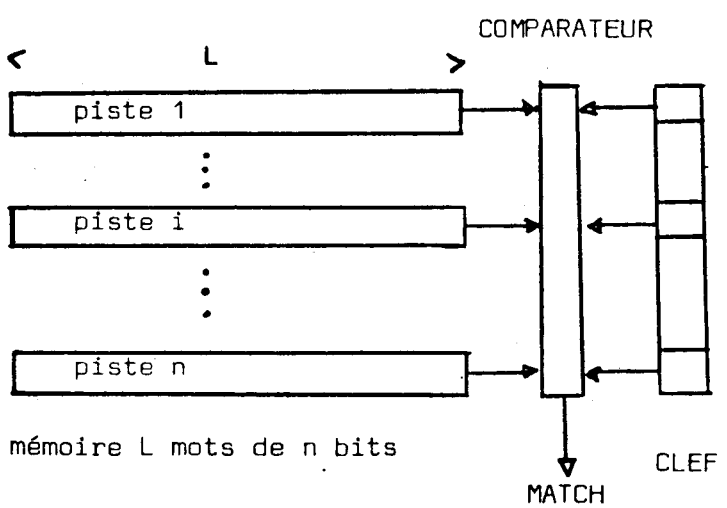


Fig. I.11 Mémoire circulante à accès associatif

...

Ce type d'utilisation, déjà réalisé avec des mémoires magnétiques, a été proposé avec des mémoires intégrées pour des machines traitant des bases de données, avec des mémoires CCD [**UNG 77**], ou des MBM [**JIN 78**]. C'est aussi un accès associatif qui est proposé par Wong et Yue [**WON 76**] pour l'implementation de leur algorithme de transposition avec des BLF (cf. § I.2.4).

I.2.4

Diminution du temps d'accès

L'intégration croissante des mémoires circulantes à un revers : c'est la longueur croissante des pistes, et donc l'augmentation des temps d'accès (à nombre de fenêtres constantes).

Plusieurs méthodes ont été proposées pour y remédier. Celle de Wong et Yue, exposée pour des BLF, consiste à proposer une alternative de l'algorithme LRU : l'application du LRU consiste à placer les données dans l'ordre de dernière utilisation :

	fenêtre ↓	
Ordre initial	C(1), C(2), ... , C(k-1), C(k), ... , C(n)	
LRU	C(k), C(2), ... , C(k-1), C(k+1) , ... , C(n)	après utilisation de C(k)
Transposition	C(1), C(2), ... , C(k), C(k-1), ... , C(n)	

L'implémentation de l'algorithme LRU nécessite le déplacement bidirectionnel des bulles (k décalages à gauches à travers le buffer, k-1 décalages à droite sans buffer).

...

L'algorithme de Stone, par exemple, introduit deux permutations : l'échange et le shuffle (fig. I.10) ; il permet d'accèder à un élément quelconque en Log_2 N permutations élémentaires

L'algorithme de transposition consiste à inverser l'ordre de C(k) et de C(k-1) lorsque C(k) est demandé. Au bout d'un nombre suffisant d'accès, les données le plus récemment et le plus souvent utilisées sont proches de la fenêtre d'accès, les autres étant rejettées peu à peu à l'autre extrémitée ; le résultat est donc une combination des algorithmes LRU et MFU.

Pour appliquer cet algorithme aux BLF, les auteurs proposent de modifier leur fabrication en doublant les ports de lectures et d'écriture, de façon à accéder à deux colonnes contigues, chaque colonne constituant une donnée C(i). Un mode de connexion simple permet alors d'échanger le contenu de ces colonnes lors d'un accès mémoire.

Une autre série de recherches, à laquelle se sont attachés Aho et Ullman [**AHO 74**], Stone [**STO 75**], Wong et Tang [**WON 77**], consiste à déterminer les caractéristiques d'une mémoire circulante qui présenterait, outre l'accès séquentiel, un accès aléatoire beaucoup plus rapide que N/2 décalages en moyenne (N étant la longueur de la mémoire). Pour cela ils proposent de donner aux mémoires des possibilités de réarangement différents de la simple rotation. Le nombre de figures ('fan out') ainsi introduites doit être minimal pour des raisons technologiques, et ils doivent permettre un accès rapide pour toutes les tailles de mémoires.

...

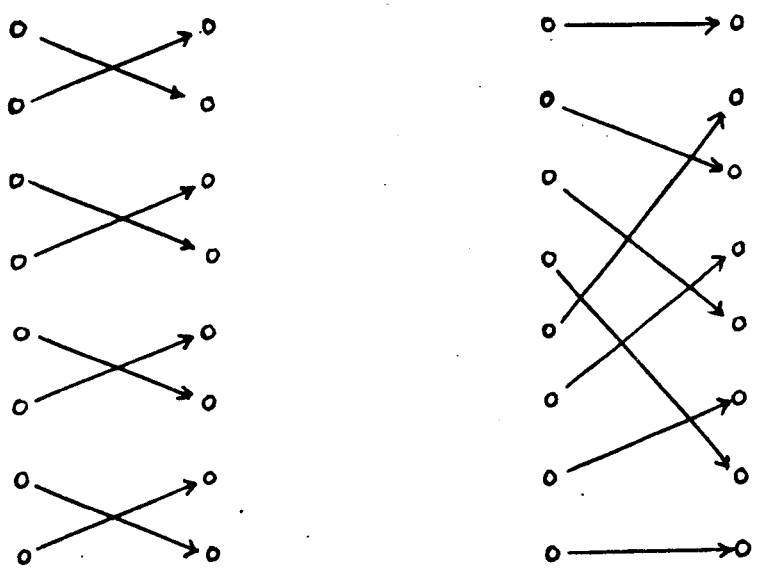


Fig. I.12 Les deux types de transpositions pour l'algorithme de Stone (N=8)

...

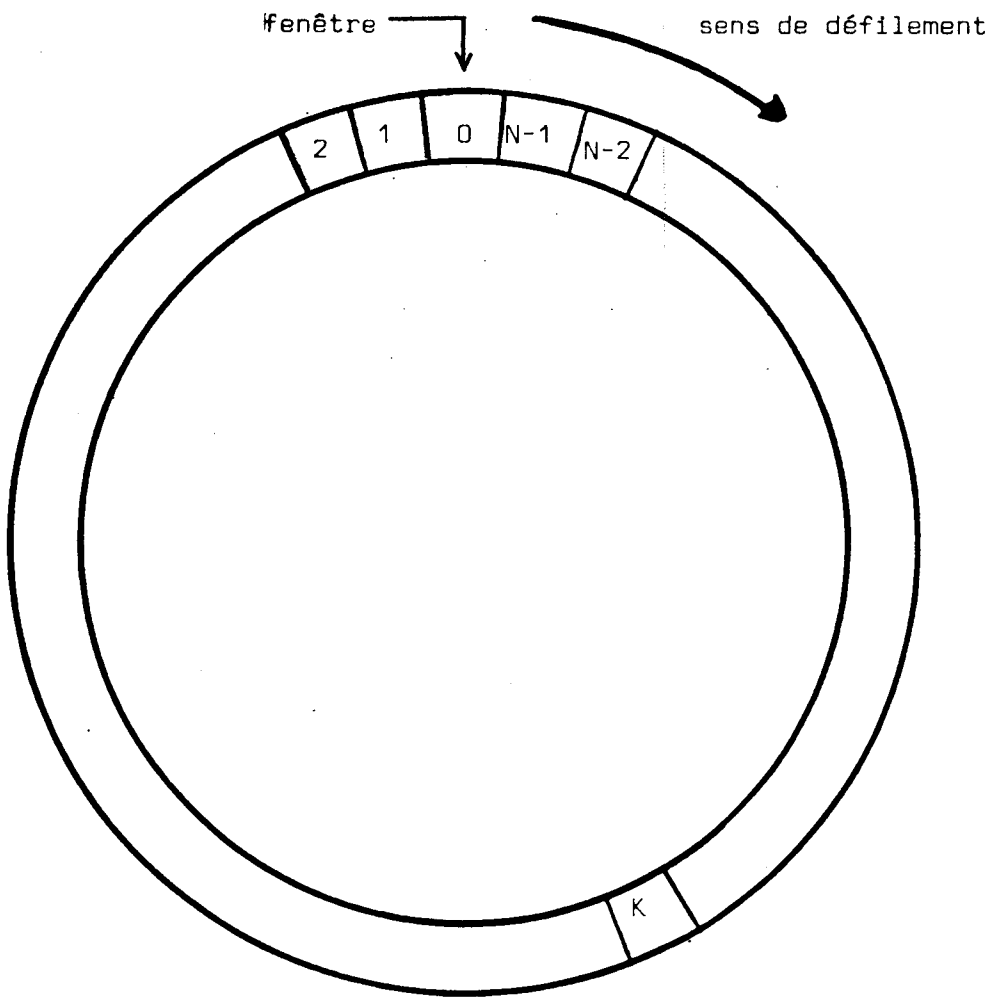


Fig. I.13 Réduction du temps d'accès au zéro d'une mémoire circulante dynamique

I.2.5
Architecture à niveau unique de mémoire

La faible prix et l'intégration poussée de mémoires circulantes amène à envisager des architectures qui soient adaptés à leur usage, et cela jusqu'au niveau de l'exécution. Les contraintes qu'elles imposent peuvent sembler incompatible avec un traitement classique : en fait l'usage séquentiel de l'information est déjà en partie une réalité pour ce qui est des instructions : le compteur ordinal d'une machine classique est 'le plus souvent' simplement incrémenté pour adresser une mémoire à accès aléatoire. Pour des applications particulières qui traiteraient des données de manière strictement séquentielle, une telle machine fonctionnerait aussi vite qu'une machine classique. Dans le cas général les ruptures de séquences existent, la pénalisation en temps d'exécution sera d'autant plus faible que ces ruptures de séquences seront peu fréquentes. Il faudra donc définir des applications-types, utilisant par nature peu de ruptures de séquences, et une méthodologie de programmation et de compilation, voire même un langage, qui linéarisent au maximum le déroulement du programme.

Parallèlement au déroulement des instructions, il faudra étudier la structure des données sur lesquelles elles opérent : pour être intéressante, la machine à définir doit pouvoir également stocker ces données de manière séquentielle, au moins dès que le volume des données dépasse nettement celui des instructions (car si le rapport

$$\frac{\text{capacité (RAM)}}{\text{capacité (mémoire circulante)}}$$

est très supérieur à 1, l'économie réalisée devient négligeable).

CHAPITRE II

MACHINES À FLUX MAITRE

ET

À FLUX SIMULTANÉS

INTRODUCTION

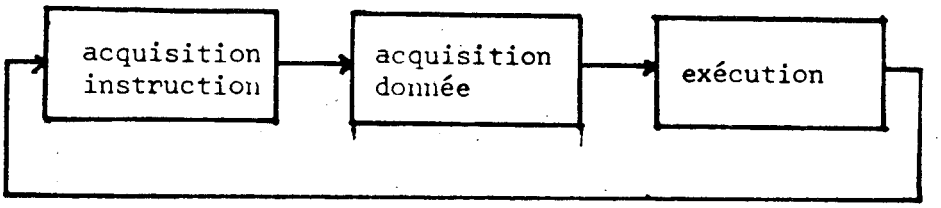
II.1.

Les machines classiques du type Von Neumann fonctionnent selon un cycle immuable qui peut se découper en trois phases :

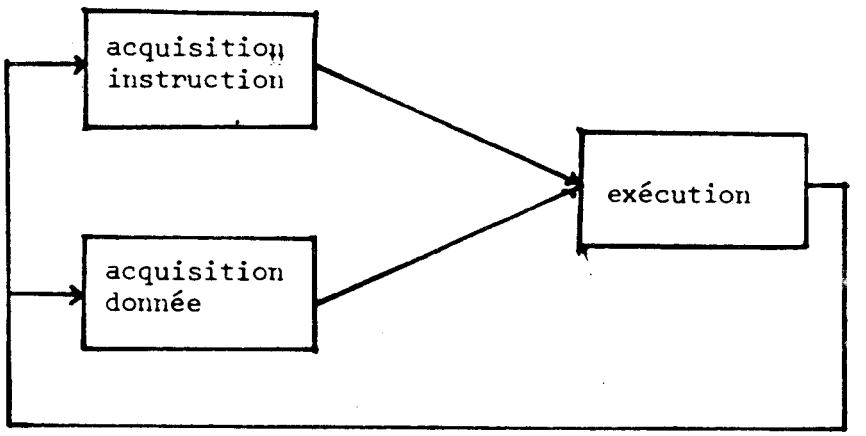
Acquisition de l'instruction
Acquisition de l'opérande
Exécution

Même si, comme cela se fait dans certaines architectures pipe-line, il existe un recouvrement entre ces trois phases, l'enchainement logique de l'une sur l'autre doit être conservé. En outre, comme la même mémoire est concernée par les deux acquisitions à réaliser avant l'exécution, le caractère séquentiel des phases dans le cycle de base est inévitable.

L'idée de base de la machine à flux simultanés consiste à effectuer en parallèle (simultanément) les deux phases d'acquisition (cf. Fig. II.1)



Cycle de base d'une machine classique



Cycle de base d'une machine à flux simultanés

Fig. II.1

Si l'on considère la machine en termes de flux, comme l'on proposé Flynn [FLY 72] puis Händler [HAN 77], c'est à dire en se plaçant du point de vue de l'organe d'exécution, on constate qu'elle est le produit du (des) flux d'instructions Φ_I et du (des) flux de données Φ_D, l'un des deux 'pilotant' l'autre.

On distingue donc les machines à pilotage par les instructions (machine de Von Neumann) et, plus récemment introduites, les machines à pilotage par les données (data driven).

II.2. LES ARCHITECTURES CLASSIQUES

II.2.1.

Machines pilotées par les instructions

La machine de Von Neumann fait jouer aux deux flux Φ_I et Φ_D des rôles très différent. Le premier est construit, a priori, par le programmeur ou le compilateur, et sa structure est totalement supportée par le mécanisme d'adressage en mémoire, le plus souvent en séquence, et sous forme explicite de ruptures de séquence pour les branchements, sauts ou appels de procédures. Le second, Φ_D, n'a aucune existence réelle dans la machine. Il est construit en pas à pas par le flux d'instructions.

A cette fin, chaque instruction contient des informations volumineuses et variées, en général une adresse, qui lui permet de définir l'élément du flux de données qui la concerne.

Ceci impose, comme on l'a dit plus haut, la mise en séquence des deux opérations d'acquisition dans Φ_I et dans Φ_D. Il en résulte en outre la nécessité de doter la machine d'un mécanisme complexe d'adressage sélectif de l'opérande.
Ce mode de fonctionnement consiste donc à adjoindre aux instructions une information permettant d'accéder aux données (Fig. II.2).

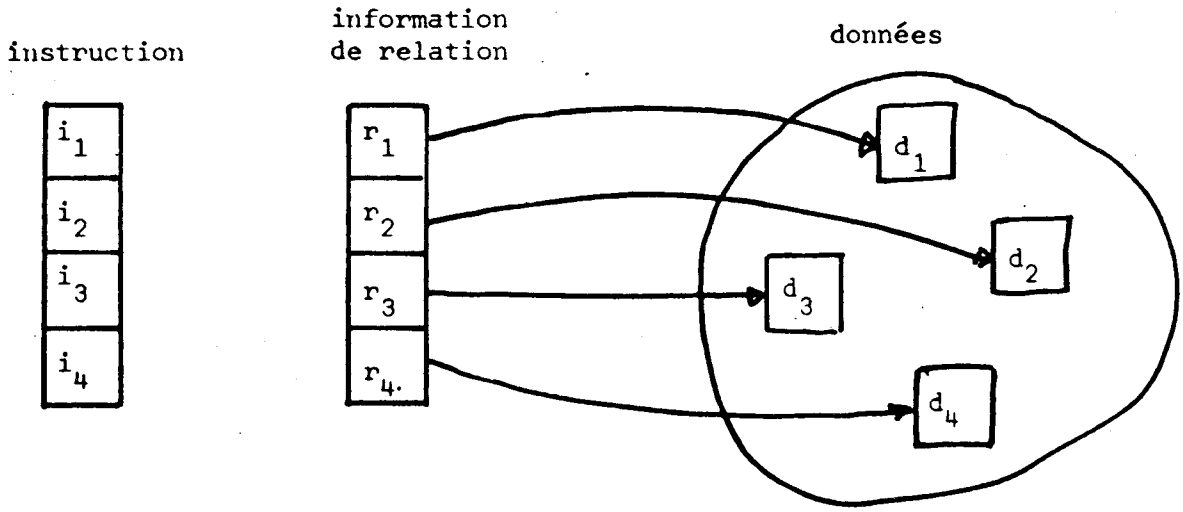
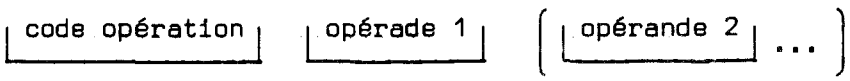


Fig. II.2. Pilotage par les instructions.

Les accès mémoires fournissant donc une 'instruction' à plusieurs champs composée de la façon suivante :



Le code opération indique le traitement à effectuer sur le(s) opérande(s). On accède à ceux-ci par un mécanisme du type

$$\alpha := [r] ; [[r]]$$

ou r peut être une adresse mémoire ou le nom d'un registre.

Le déroulement d'un programme est contrôlé par le compteur ordinal, auto-incrémenté à chaque exécution, et accessible par programme (instructions de ruptures de séquence).

Il faut noter que l'emplacement des données dans la mémoire est a priori indifférent, ce qui oblige à disposer d'un mécanisme d'accès sélectif à l'ensemble de l'espace adressable, ceci pour chaque instruction. Dans l'exemple suivant pour le 8080 :

```
ETIQ : LXI  H, REL1
       ADD  M
       STA  REL2
```

ETIQ, REL1, et REL2 sont des adresses, sur deux octets, fixés arbitrairement entre 0000 et FFFF.

II.2.2

Machines pilotées par les données

Plus récemment, les machines à pilotage par les données [DEN 74] [COM 74] proposent de renverser sous des formes différentes le rôle des deux flux et d'exprimer explicitement dans la machine une structure de données. Cette structure est habituellement un graphe.

L'analyse de ce graphe permet alors de construire en pas à pas le flux d'instruction.

Les instructions ne sont donc plus exécutées dans un ordre prédéfini, mais seulement lorsqu'elles ont été 'prévenues' que toutes les informations (opérandes) dont elles ont besoin ont été évaluées. Le résultat de cette exécution est en général un opérande pour d'autres instructions, ces dernières vont alors être averties de l'évaluation qui en a été faite.

En prenant pour exemple le système 'data driven' à assignation unique décrit dans [SYR 76], les instructions se présentent sous la forme suivante (instructions à deux opérandes) :

code opération	adresse résultat	adresse opérande 1	adresse opérande 2	C_o	C_1	C_2

C_1 et C_2 sont des bits relatifs aux opérandes 1 et 2 qui indiquent si ceux-ci ont été ou non évalués, C_o permet de tenir compte des exécutions conditionnelles : une instruction est donc exécutée dès que $C_o.C_1.C_2=1$

Le format des opérandes est le suivant :

valeur	Lien 1	Lien 2

C_d

Le bit C_d valide le champ 'valeur'. Les liens 1 et 2 renvoient aux instructions qui utilisent la valeur comme opérande.

Le déroulement d'un programme se fait de la manière suivante :

1 Recherche d'une instruction dont les bits C_i sont à '1' (ex : les opérandes sont des données initiales)

2 Exécution de l'instruction (calcul de la valeur du résultat)

3 Rangement de la valeur, positionnement du bit C_d

4 Positionnement des bits C_i dans les instructions (pointées par lien 1 et lien 2) qui utilisent la valeur comme opérande

5 Retour en 1

Ce type de processeur nécessite une recherche associative des instructions, mais il a l'avantage de permettre un parallélisme total au niveau des instructions : si plusieurs instructions sont exécutables simultanément ($C_o.C_1.C_2=1$) elle peuvent être exécutées sur des processeurs banalisés, sans limitation de nombre.

II.3 NOTION DE FLUX SIMULTANES

A la différence des deux prinicpes que nous venons d'étudier, la machine à flux simultanés ne privilégie pas l'un des flux : il sont rangés en mémoire dans un ordre identique, et l'accès se fait simultanément pour l'instruction (code opération) et la donnée (l'opérande) pour être traités par l'organe d'exécution (Fig. II.3).

L'ordre normal du déroulement est séquentiel, c'est donc notemment pour tirer partie des mémoires séries (CCD et MBM) que cette architecture a été introduite [TOU 78 a].

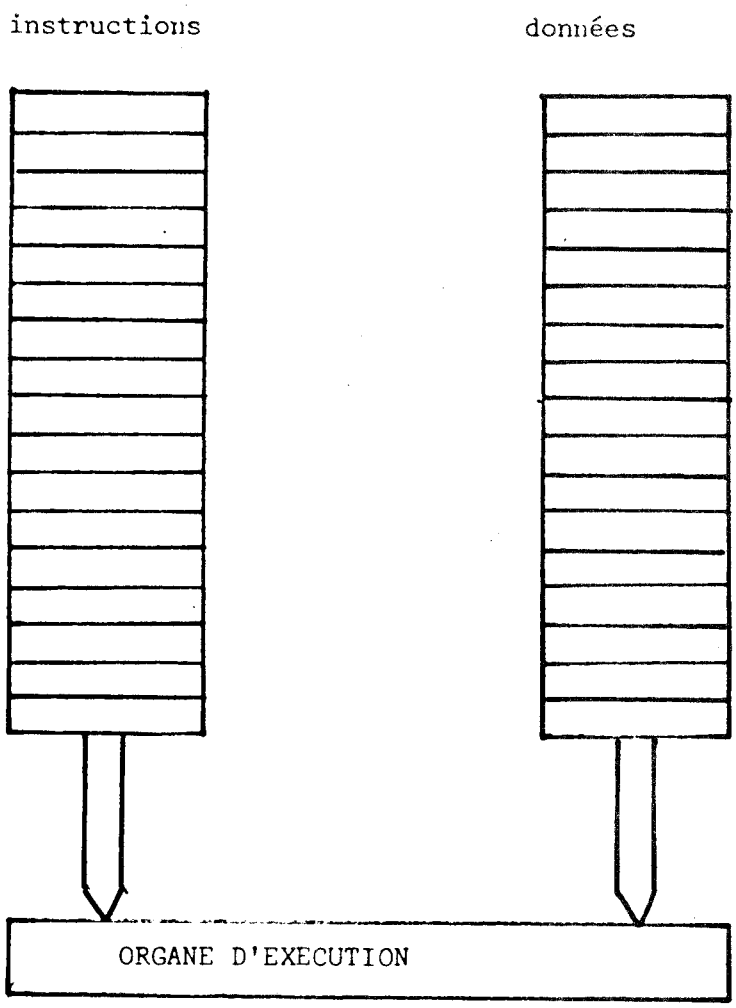


Fig. II.3 Machine à flux simultanés

On peut espérer d'une telle machine un très bon rapport débit/coût ; de plus une modularité au niveau du nombre de flux traités permet de travailler sur plusieurs données (instructions à plusieurs opérandes, ou machines SIMD suivant Flynn) et d'être alors bien adapté au traitement vectoriel. Dans la présentation qui suivra, on se limitera à un seul flux d'instruction et un seul flux de données. L'extension à plusieurs flux pouvant en partie s'obtenir par duplication des mécanismes associés à chacun d'eux.

Il apparaît immédiatement qu'un tel mécanisme est inapte à prendre en charge les ruptures de séquences d'une part, et d'autre part à gérer les données intermédiaires (résultats de traitements réutilisés par la suite).

Dans les cas simples, un seul pointeur (compteur ordinal) suffira à gérer les ruptures de séquences. En général, il sera nécessaire de désynchroniser les deux flux en associant un compteur aux instructions et un autre aux données, ceux-ci étant accessible à l'unité de traitement.

Pour le rangement des résultats intermédiaires, il faut des réinsérer dans le flux de données par un accès a priori aléatoire. Si l'on utilise des mémoires circulantes, il faudra donc créer un mécanisme de réinsertion qui concilie les performances avec une complexité acceptable, proche de la simplicité initiale.

On parlera de machine à flux simultanés symétriques si, de la même manière que des instructions particulières provoquent un branchement dans le flux d'instruction et donc une synchronisation du flux de données, on prévoit dans le flux de données des ordres de branchement pour le flux d'instruction. Les ruptures de séquences de chacun des flux sont traités dans les processeurs de commande de flux (Fig. II.4).

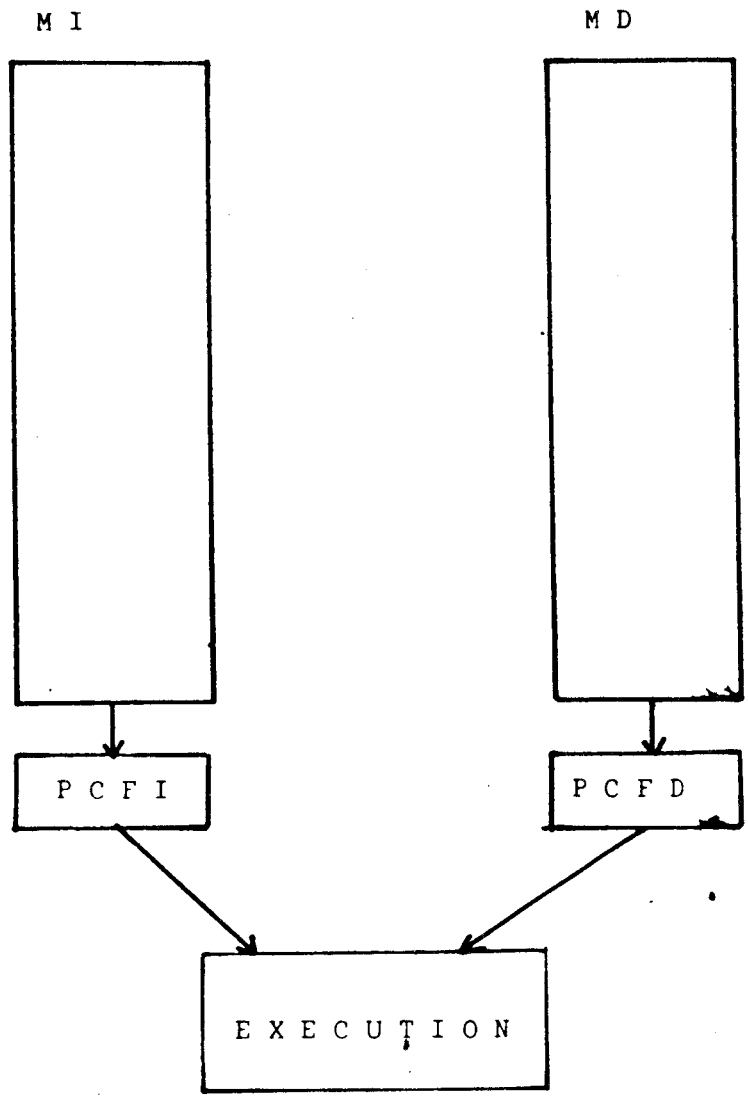


Fig. II.4 Schéma fonctionnel de la MFSS

II.4 LE MECANISME DE REINSERTION

En général, dans le cours d'un programme, les opérande ne sont pas des données initiales, connues avant l'exécution, mais elles sont évaluées par d'autres instructions antérieures du programme et doivent être rangées dans la mémoire de données avant l'exécution.

Voici un exemple qui met en oeuvre le mécanisme de réinsertion, il s'agit de faire le calcul :

$$x = \frac{c+(a+b)}{c-(a+b)}$$

nous utiliserons l'algorithme suivant :

```
1    u = a + b
2    w = c - u
3    v = c + u
4    x = v/w
```

Les écritures de ce programme pour une machine traditionnelle et pour une MFS sont comparées à la Fig. II.5.

On constate qu'il a fallu ajouter une information, l'information d'insertion. Celle-ci permet de ranger la donnée qui vient d'être calculée dans le flux de données, à l'endroit où elle va être utilisée : la valeur de u, calculée au pas 2 va être rangée dans le flux de données au pas 4 (2+2) pour la soustraction à C. Cette information d'insertion est une forme d'adressage relatif, plus simple et plus limitée que dans le cas des machines traditionnelles : elle est parfaitement déterminée à la compilation, et sa valeur n'est nullement arbitraire, contrairement aux adresses u, v, w et x du programme traditionnel ; elle est de plus compatible avec les translations que l'on pourra effectuer sur le programme.

Pas	Instruction	Adresse de donnée	Commentaires
1	LDA	a	Acc ← a
2	ADD	b	Acc ← (Acc) + b
3	STA	u	u ← Acc
4	LDA	c	Acc ← c
5	SUB	u	Acc ← (Acc) - (u)
6	STA	w	w ← (Acc)
7	LDA	c	Acc ← c
8	ADD	u	Acc ← (Acc) + (u)
9	DIV	w	Acc ← (Acc) + (w)
10	STA	x	x ← (Acc)

Ecriture du programme pour une machine traditionnelle.

Pas	Instruction	Donnée	Information d'insertion	Commentaires	
1	LDA	a	ϕ	Acc ← a	
2	ADD	b	+2	Acc ← (Acc) + b	MD(4) ← (Acc)
3	LDA	c	*+3	Acc ← c	MD(6) ← c
4	SUB	Ω	*+3	Acc ← (Acc) - u	MD(7) ← u
5	STA	ϕ	+3		MD(8) ← w
6	LDA	Ω	ϕ	Acc ← c	
7	ADD	ϕ	Ω	Acc ← (Acc) + u	
8	DIV	Ω	η	Acc ← (Acc) : w	MD(η+8) ← x

Ecriture du programme pour une machine MFS

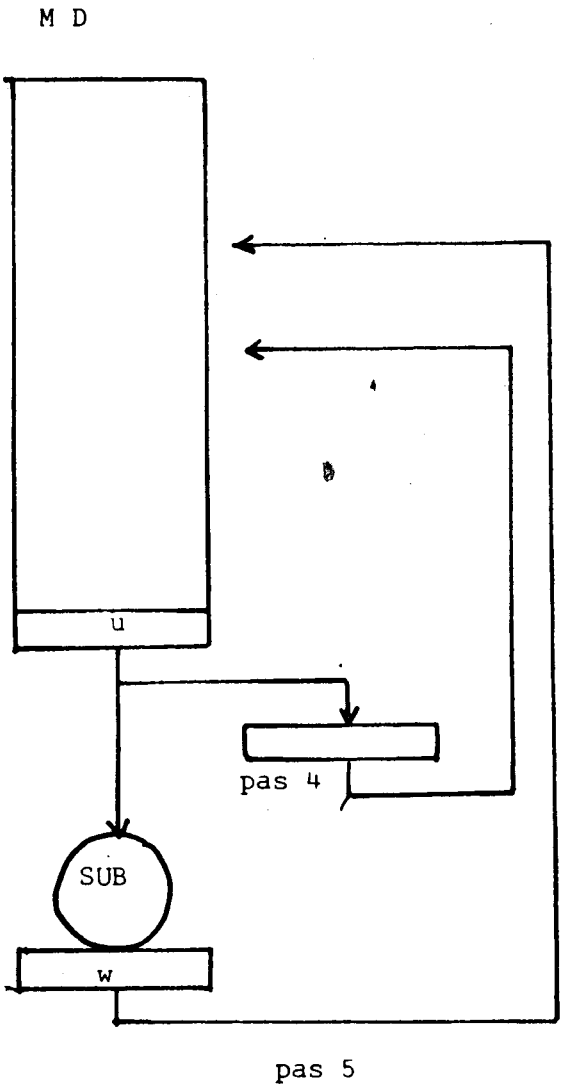
Fig. II.5 Comparaison entre les modes de traitement.

Suivant le format des instructions qui sera retenu, on peut envisager soit de conserver des instructions à trois champs (code opération, donnée, insertion), soit de les compacter ; en effet un certain nombre de champs sont inutilisés (ceux qui sont notés ø). On peut donc envisager de regrouper donnée et insertion, en dédoublant les instructions lorsqu'il faut effectuer des rangements d'opérandes ; il faut alors prévoir un buffer supplémentaire, et le pas 4 de l'exemple précédent devient

```
4 LOAD Buffer          Λ
5 SUB  Buffer & copy +3
```

On peut aussi distinguer dans le cas de mémoires séries la mémoire de données prêtes (les constantes et les valeurs intermédiaires que l'on a pu ranger dans le flux de données) et une mémoire de faible capacité qui contient les valeurs récemment calculées. A chaque pas, un bit associé à la mémoire de donnée nous indique si celle-ci est vide ; si oui, on accède à la mémoire annexe qui a 'préparé' les données récente dans leur ordre d'utilisation.

Les différents choix possibles sont étudiés à la figure II.6



au pas 4, on désire ranger u, et effectuer la soustraction

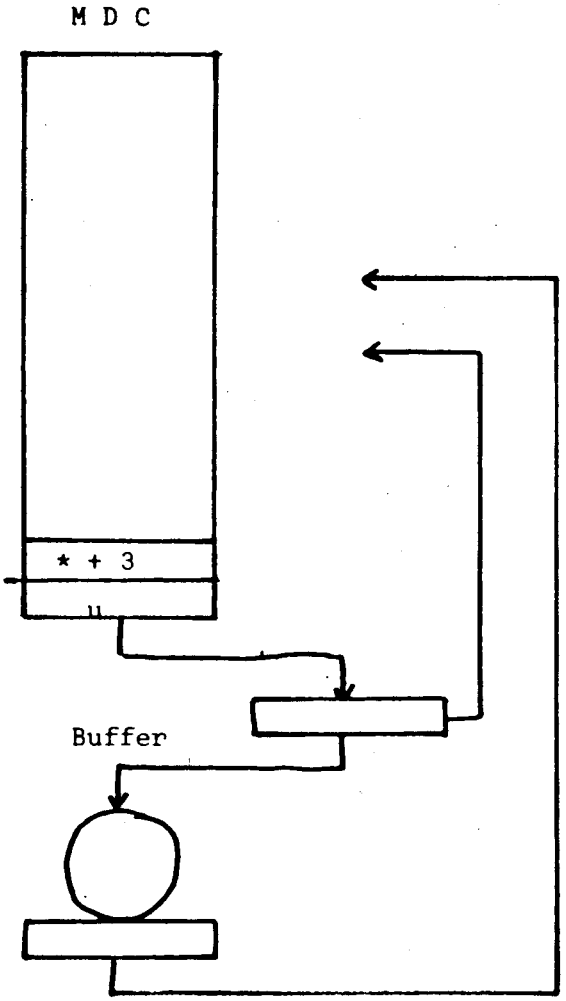
w ← (Acc) - u

4	SUB	^ *	+ 3
5	STA	Ø	+ 3

Insertion de l'opérande et du résultat (Mémoire de contrôle et de données disjointes).

Fig. II.6 Le problème de l'insertion, les choix possibles

Fig. II.6 (Suite)



au pas 4, on charge u

au pas 5, on effectue l'opération

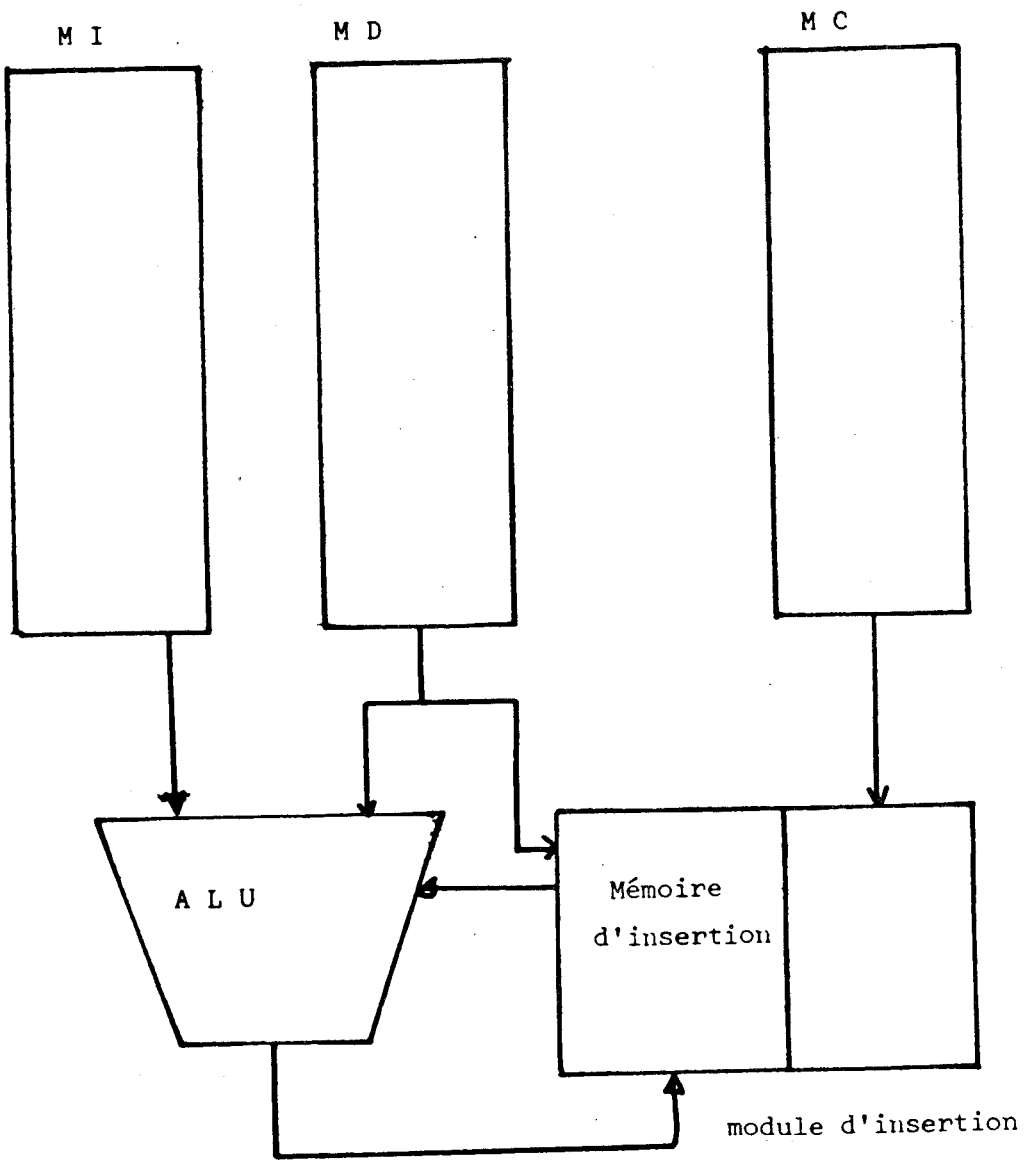
et on range u

au pas 6, on range le résultat

```
4 LD Buffer    ^
5 SUB Buffer * + 3
6 STA          + 3
```

Insertion de l'opérande et du résultat avec une mémoire de contrôle et donnée

Fig. II.6 (Suite et fin)



Existence d'une mémoire dans le module d'insertion - schéma fonctionnel

La traduction en 'langage machine' MFS est la suivante (fig. II.)

Pas	MI	MD	MC	Commentaires	
1	LDBuff	1	+2		MD(3) ← 1
2	LDA	n	+4	A ← M	MD(6) ← n
3	MUL	Λ	0	A ← (A)	MD(3) ← fac
4	STA	ϕ	+3		MD(9) ← fac
5	DCR	Λ	0	A ← i-1	MD(5) ← i
6	JNZ	-3	0	saut en 3 si i≠ 0	
7	STBuff	Λ	x		MD(9+x) ← fac

On peut envisager d'utiliser comme mémoire d'insertion (Fig. II.6) en plus d'une mémoire de type RAM associée à un processeur de Tri, une pile qui stockerait les valeurs, de types paramètres de procédures, ou, dans le cas de la boucle, les paramètres de boucles comme la valeur fac ; cela éviterait de faire à chaque exécution de la boucle une copie de la valeur courante vers l'extérieur (pas 5) qui 'écrase' la valeur précédente.
Le programme devient alors (Fig. II.10).

II.5 LES RUPTURES DE SÈQUENCE

L'exemple précédemment traité était "linéaire" : dans le cas des ruptures de séquence, si 'condition' alors 'f_1' sinon 'f_2' par exemple, une même valeur peut être utilisée dans les deux alternatives, il faut donc prévoir une copie dans chaque branche. De même une valeur peut être calculée d'une manière différente dans chaque branche, et être utilisée par la suite (Fig. II.7).

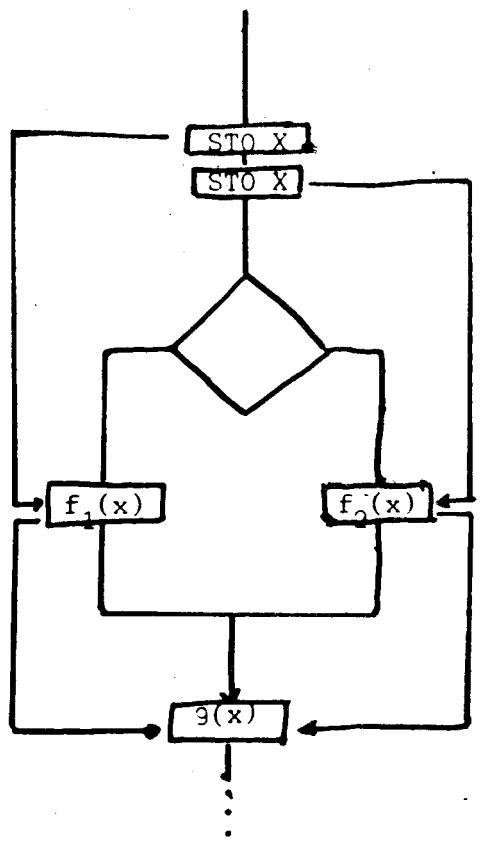


Fig. II.7 Duplication des copies dans le cas d'une alternature

Pour assurer la simplicité du mécanisme, et pour une étude rationnelle des informations d'insertion, on supposera que les programmes à traiter respectent les règles de la programmation structurée (Fig. II.8).

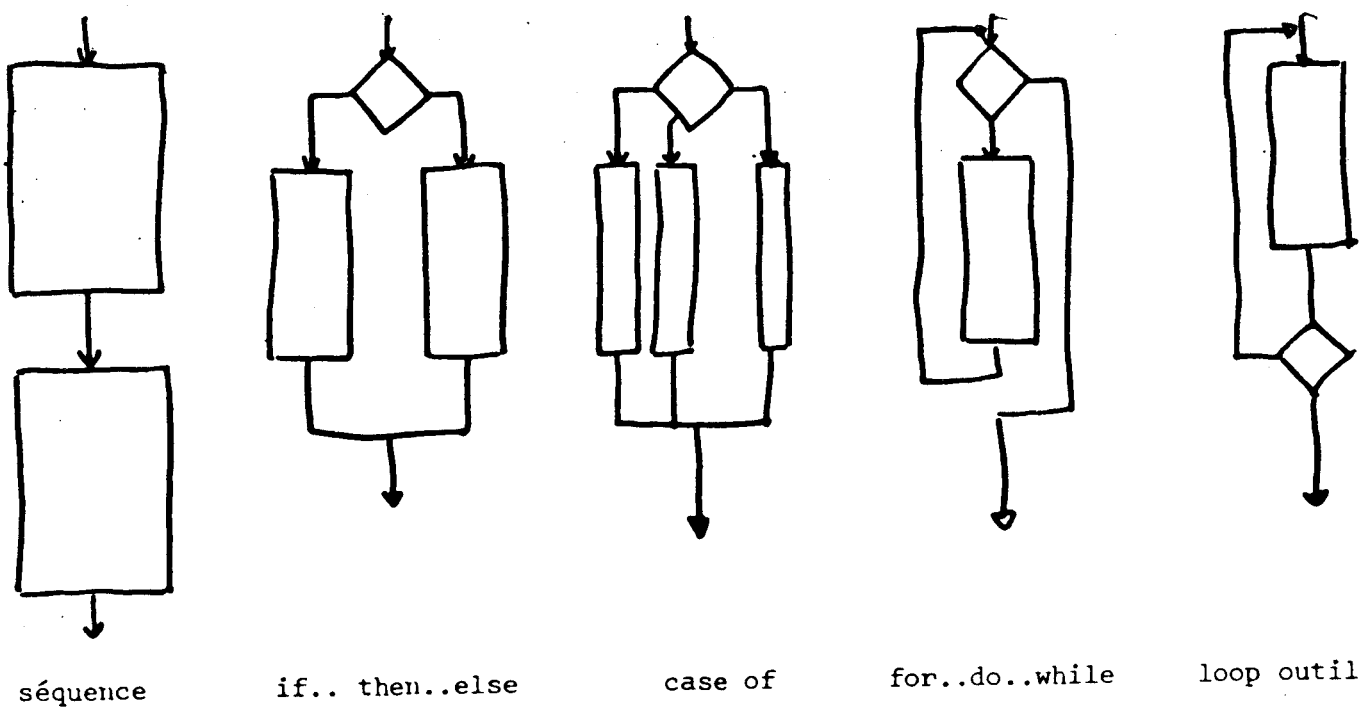


Fig. II.8 Les règles de construction de blocs en programmation structurée

Dans la mesure où le programme doit être compilé, on peut envisager, pour pouvoir créer les informations d'insertion, d'obliger le programmeur à l'écrire de manière structurée ; l'étude de la structure du programme se faisant lors de l'analyse syntaxique.

La manière de réinsérer les données dans le cas des ruptures de séquences est illustrée dans le cas d'une boucle par l'exemple suivant, calcul de factoriel n. Le programme en langage évolué (type pascal) est le suivant :

```
program n factoriel ;
var fac, i, n, integer ;
begin    read (n) ;
         i := n ;
         fac := 1 ;
         reapeat begin
                    fac := fac*i
                    i := i-1
                 end
         until i = o ;
         write (fac) ;
end
```

La traduction en langage MFS est la suivante : (fig. II.9).

On peut envisager d'utiliser comme mémoire d'insertion (fig. II.6) en plus d'une mémoire de type RAM, associée à un processeur de tri, une pile qui stockerait les valeurs, de types paramètres de procédures, ou, dans le cas de la boucle, les paramètres de boucles comme la valeur fac ; cela éviterait de faire, à chaque exécution du corps de la boucle une copie de la valeur courante vers l'extérieur (pas 4 dans la fig. II.9). qui "écrase" la valeur précédente. Le programme devient alors : (fig. II.10).

pas	MI	MD	MC	Commentaires
1	LOAD	1	+2	MD(3) ← 1
2	LDA	n	+3	A ← n MD(5) ← n
3	MUL	Λ	0	A ← (A) * MD : MD(3) ← fac
4	STA	Ø	+3	MD(7) ← fac
5	DCR	Λ	0	A ← i-1 ; MD(5) ← (A)
6	JNZ	-3	0	saut en 3 si i ≠ 0
7	STO	Λ	x	MD(9+x) ← fac

Fig. 2.9. Programme factoriel en langage machine MFS.

pas	MI	MD	MC	Commentaires
1	PUSH	1	Ø	SdP ← 1
2	LDA	n	+4	MD(6) ← n
3	POP	Λ	Ø	Acc ← fac
4	MUL	Λ	Ø	Acc ← (Acc) * fac
5	PUSH	Λ	Ø	SdP ← fac
6	DCR	Λ	0	Acc ← (MD)-1 ; MD(6) ← (Acc)
7	JNZ	-4	0	saut en 3 si (Acc) ≠ 0
8	POP	Λ	x	MD(8+x) ← fac

Fig. 2.10. Programme factoriel en langage machine MFS avec une pile.

CHAPITRE III

ARCHITECTURE ET LANGAGE MACHINE

III.1 COMPILATION ET EXECUTION

La conclusion évidente du chapitre précédent est que le langage de la machine à flux simultanés est compilable, mais difficilement interprétable : la gestion des informations de réinsertion se réalise de manière statique. Le problème n'est guère différent de la gestion de la table des étiquettes dans une compilation traditionnelle, mais l'insertion d'une instruction modifie non seulement le flux de données pour les instructuions de saut, mais également l'ensemble des informations d'insertion. La gestion dynamique d'une telle table serait donc si lourde (on peut considérer que chaque instruction est étiquetée) qu'il semble préférable de se limiter à des applications compilables, le gain apporté par la machine au point de vue performances étant acquis à l'exécution.

Le domaine d'activités privilégié d'une telle architecture se caractérise donc par deux points :

- en premier lieu, les structures de données doivent être relativement simples et bien définies ; ceci permet d'optimiser à la compilation la gestion de la mémoire qui les reçoit

- il faut qu'en outre, la compilation soit justifiée par de nombreuses exécutions ultérieures

Parmi les applications qui répondent à ces deux critères, on peut citer :

- contrôle de processus
- traitement du signal
- contrôle de terminaux et de communications
- machines spécialisées à programmes fixes stockés en mémoire morte
- traitements vectoriels, matriciels, FFT
- traitements de textes.

...

III.2 ARCHITECTURE DE LA MFS

La machine à flux simultanés, telle qu'elle a été présentée au chapitre 2, nécessite des instructions à trois champs (code opération, champ donnée, champ insertion). L'implémentation physique tiendra compte des caractéristiques différentes de ces champs, en particulier du fait que le champ "données" contrairement aux deux autres, n'est pas complétement défini à la compilation.

Il convient de distinguer le type de support utilisé comme mémoire : les options ne seront pas les même suivant que l'on emploie une mémoire à accès aléatoire (RAM ou ROM + RAM) ou une mémoire séquentielle (CCD, bulle ...). Dans le premier cas, c'est la vitesse d'exécution qui sera le critère principal, dans le second c'est le coût minimal de la machine.

III.2.1 Architecture d'une MFS avec mémoire à accès aléatoire

Une architecture simple, mais néanmoins suffisamment puissante pour être comparée à celle des microprocesseurs actuels, est présentée à la fig. 3.1.

Les trois champs sont physiquement répartis dans trois mémoires (Mémoire d'Instruction, Mémoire de Donnée, Mémoire de Contrôle).

La mémoire d'instruction contient les codes opération, on y accède en lecture seule lors de l'exécution.

La mémoire de contrôle contient le déplacement à calculer pour ranger le résultat (ou l'opérande) ; son fonctionnement est identique à celui de la mémoire d'instruction.

La mémoire de donnée est plus particulière : elle joue en effet le rôle de mémoire de données initiales, comme défini précédemment, et celui de mémoire de réinsertion des données intermédiaires. A cet effet elle dispose d'un double accès : une voie de lecture et une voie d'écriture. On y accède en lecture pour acquérir les données nécessaires à l'exécution et en écriture pour y ranger les résultats de traitements.

...

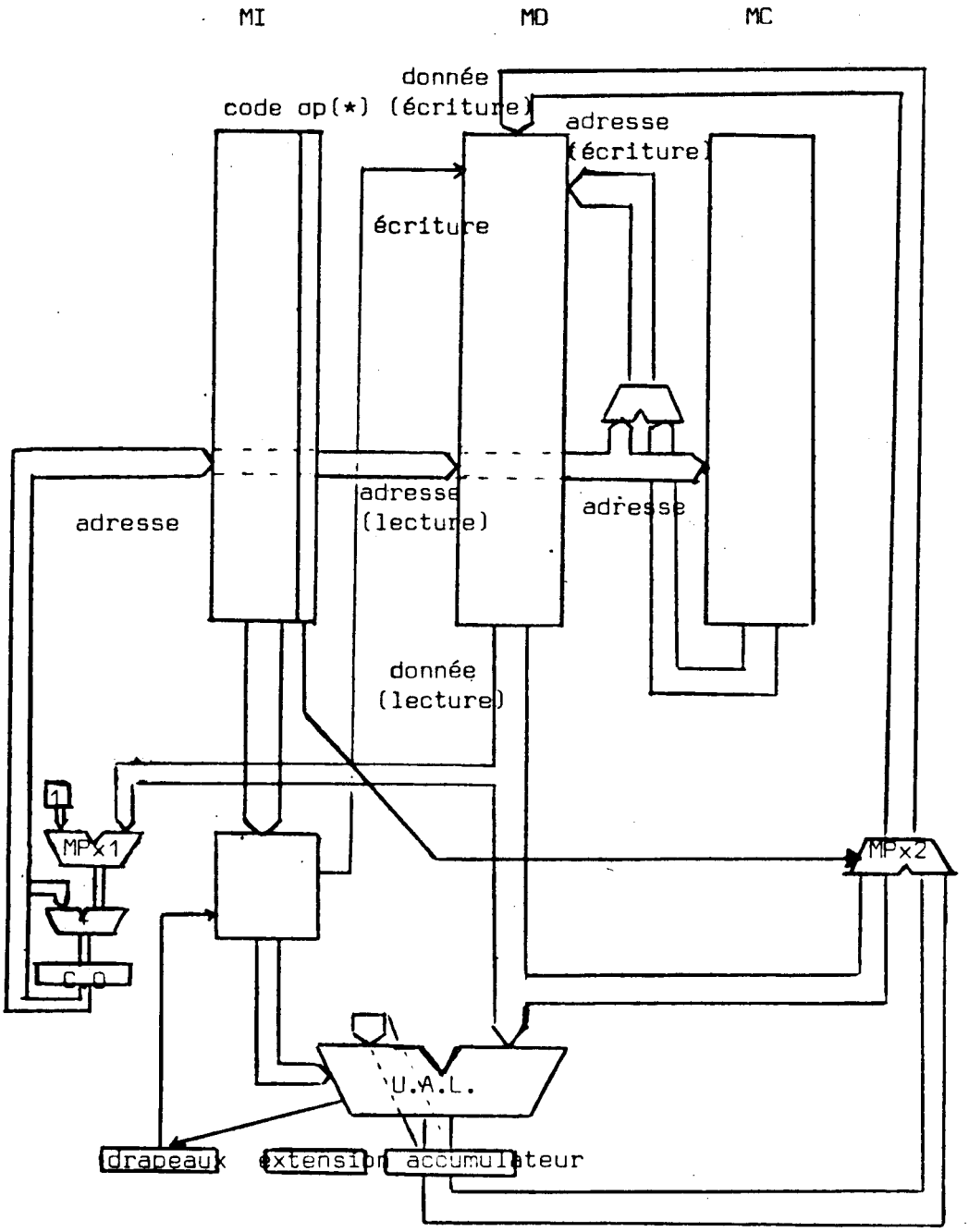


Fig. 3.1. Architecture d'une MFS avec mémoire à accès aléatoire.

Ce type de mémoire à double accès permettant une lecture et une écriture simultanée n'est pas disponible avec un haut niveau d'intégration. Cependans leur réalisation ne pose pas de problème technologique particulier. La seule limitation étant la taille des cellules qui est plus importante que celle des mémoires RAM statiques. La solution couremment employée est plutôt de faire des accès séquentiellement, sur des RAM standard, ce qui double le temps d'accès.

Le multiplexeur 1 permet soit d'incrémenter le compteur ordinal, soit d'effectuer un saut :

$$co \leftarrow (co) + (MD)$$

Le multiplexeur 2, commandé par l'indicateur * (figuré ici dans le champ instruction), permet de faire

Adresse écriture = (accumulateur) : rangement du résultat
ou Adresse écriture = (MD) : rangement de l'opérande

Plusieurs options peuvent être prises pour la réalisation de cette architecture. La nécessité de limiter la taille de MD à une valeur raisonnable peut amener à concevoir un fonctionnement par pages, chaque page contenant alors une tâche relativement autonome. MI, MD et MC sont des mémoires RAM. Les pages qui ne sont pas en cours d'exécution sont rangées sur un support externe (CCD par exemple) ; on retrouve alors une situation classique de gestion d'une hiérarchie de mémoire. La nécessité de limiter les changements de pages doit amener à regrouper dans une même page les parties de programmes qui sont le plus fortement couplées, et même à dupliquer les routines partagées par plusieurs tâches.

Une architecture réalisée dans cet esprit permetrait un espace d'adressage très vaste : la nécessité de conserver un débit important imposant seulement que chaque tâche ne dépasse pas la taille de la mémoire centrale (MI + MD + MC). Leur ordre de grandeur, en l'état actuel de la technologie, est estimable à quelques K-octets, et les performances comparables au

...

aux micro-ordinateurs universels actuels (type I 8080) - cf. Ch. 4.

Le second champ d'applications peut-être défini comme celui des micro-ordinateurs monochip (type I 8048) qui sont dédiés à une activité relativement simple, et qui contiennent toute la mémoire nécessaire à leur mise en oeuvre.

Le 8048 contient 1K octets de mémoire ROM et 64 octets de RAM (dont les registres et la pile). Pour obtenir une puissance de calcul comparable, l'ensemble des mémoires devrait être de l'ordre du kilo-octet (voir justification au Ch. 4). Si on limite chaque champ à 8 bits, on peut accéder directement à 256 instructions "machine à double flux", et la taille totale est alors de 3/4 K-octet. Le codage des instructions nécessite un champ moins large à puissance égale qu'un microprocesseur classique (absence de registres généraux) ce qui laisse de la place pour l'indicateur * (opérande/résultat) et pour d'autres mécanismes. La configuration s'apparente alors plus à un automate qu'à un véritable ordinateur. Dans ce cas, tout particulièrement, l'architecture proposée prend un avantage très net sur l'organisation classique car elle est à la fois plus rapide est plus simple.

Pour obtenir des versions plus puissantes, il suffit d'augmenter la taille de MC en largeur, et l'ensemble en longueur dans le rapport correspondant.
Par exemple MI et MD : 1K × 8 bit
MC : 1K × 10 bit
Ce qui fait une puissance correspondant à 3K-octets pour un micro-ordinateur classique.

MI et MC peuvent être du type ROM, alors que MD, à lecture/écriture, doit être chargé à la mise sous tension par des valeurs stockées dans une petite ROM annexe.

Le passage d'une version basse à une version haute (256 à 1K instructions par exemple) s'obtient sans modification du logiciel, et avec la seule modification du matériel ci-dessus. Le langage machine étant translatable, l'assemblage de programmes partiels est très simple.

...

III.2.2

Architecture d'une MFS avec mémoire séquentielle

Le principe original de l'architecture à flux simultanés vise plus à valider l'emploi de mémoires séquentielles que de mémoires à accès direct.
dans l'état actuel de la technologie et pour encore quelques années, les mémoires séquentielles présentent plusieurs avantages.

a) A capacité comparable, le coût du stockage est deux fois moindre. L'arrivée sur le marché des mémoires à bulles de grande capacité va confirmer et même renforcer cet avantage.

b) Le nombre de broches est indépendant de la capacité, alors qu'il croit comme le logarithme de la capacité dans une RAM. Pour des capacités élevées, ceci devient déterminant.

c) L'intégration d'une cellule de stockage n'exige pas que celle-ci communique avec l'extérieur, mais simplement avec ses deux voisins. Le gain qui en résulte concerne la simplicité de conception des masques et la densité des cellules. A surface de silicium équivalente, la capacité du stockage peut être plus que doublée.

L'emploi d'une mémoire série à nombre de fenêtres d'accès (cf. Ch. I) réduit impose des contraintes quand au temps d'exécution. Si on peut estimer que celles-ci sont acceptables pour le flux d'instruction (les ruptures de séquence imposant des délais pour accéder à l'instruction à exécuter), on ne peut envisager de mettre à jour systématiquement la mémoire de données. Il est donc nécessaire de prévoir une mémoire auxiliaire pour stocker les résultats intermédiaires (mémoire d'insertion).

La mémoire d'insertion contiendra des informations sous la forme d'un couple (adresse, donnée), l'adresse étant délivrée par la mémoire de contrôle, la donnée étant le résultat de l'instruction en cours (ou son opérande). L'accès en lecture à cette mémoire devra être du type associatif, et se fera dans deux cas.

...

1 L'adresse est égale à la valeur du compteur ordinal.
La donnée est alors l'opérande de l'instruction à exécuter.

2 L'adresse est égale à celle d'une autre fenêtre d'accès à la mémoire de donnée.
On range alors la donnée en libérant un emplacement de la mémoire de réinsertion.

L'architecture est décrite à la figure 3.2.

III.2.3

Réduction des temps morts dans une architecture à mémoire série

L'utilisation de mémoires séries pour le stockage des instructions et des données pose deux problèmes :

1 l'attente des instructions dans le cas des sauts (surtout pour les sauts arrières, du type boucles)

2 la taille de la mémoire de réinsertion qui augmente lorsque le nombre de fenêtres d'accès diminue.

La multiplication des fenêtres d'accès n'est pas une solution dans la mesure où elle s'oppose au principe d'intégration croissante des mémoires. Le choix d'une disposition judicieuse de ces fenêtres et du mode de circulation des informations influence considérablement les deux paramètres cités.
Plusieurs organisations peuvent être proposées.

III.2.3.1

Configuration logarithmique (Fig. 3.3.).

Cette configuration est relativement complexe à réaliser, mais présente les avantages suivants :

...

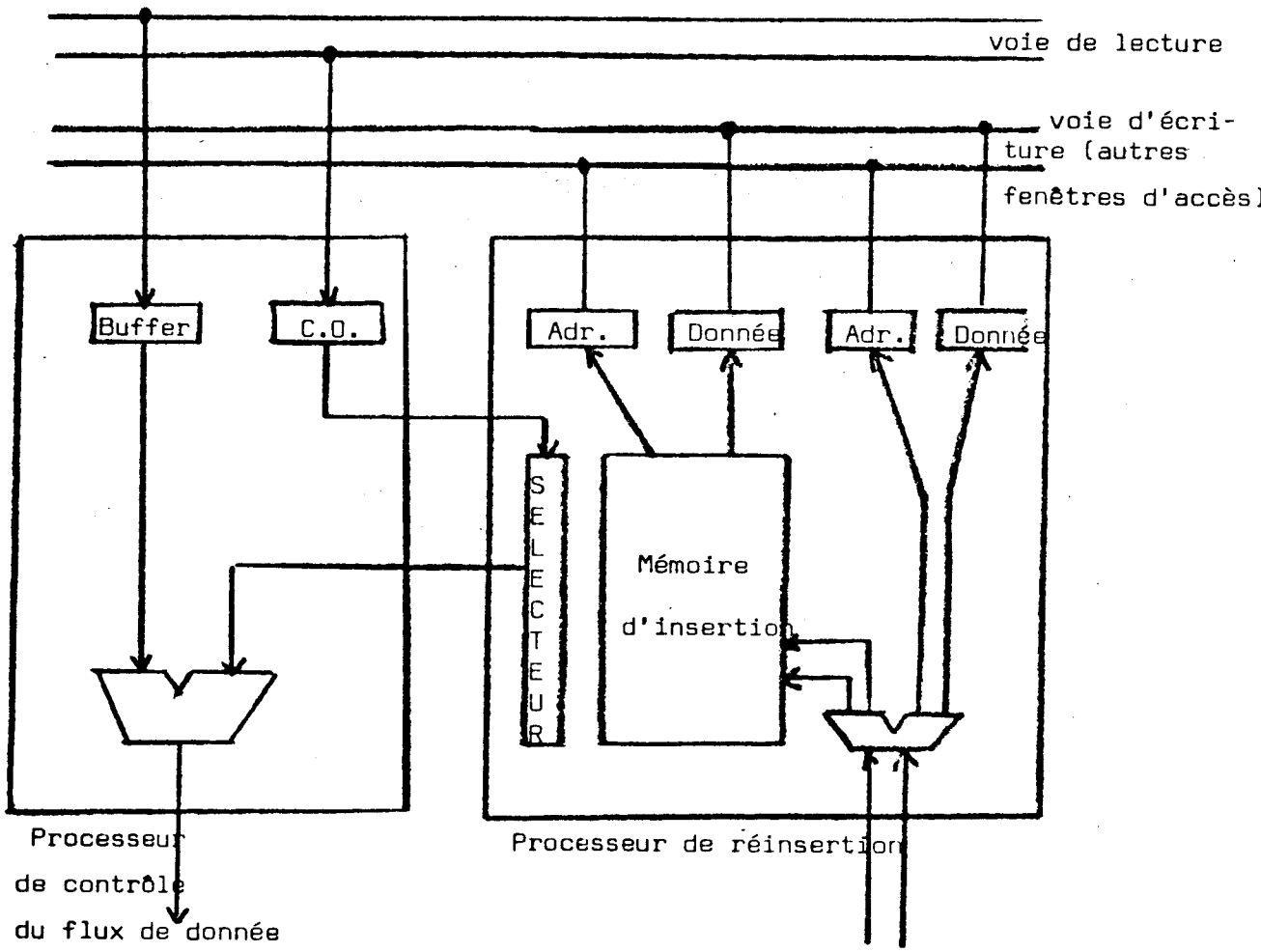


Fig. 3.2 Réalisation du mécanisme de réinsertion.

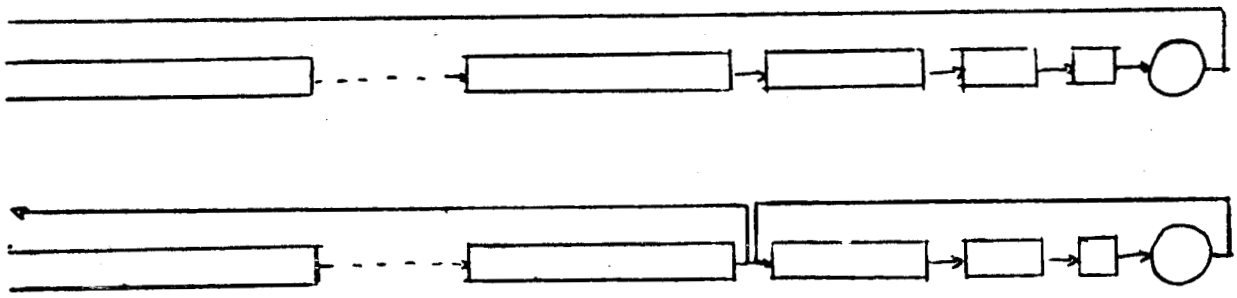


Fig. 3.3. Mémoire à accès logarithmique

Fig. 3.4. Mémoire à configuration circulaire

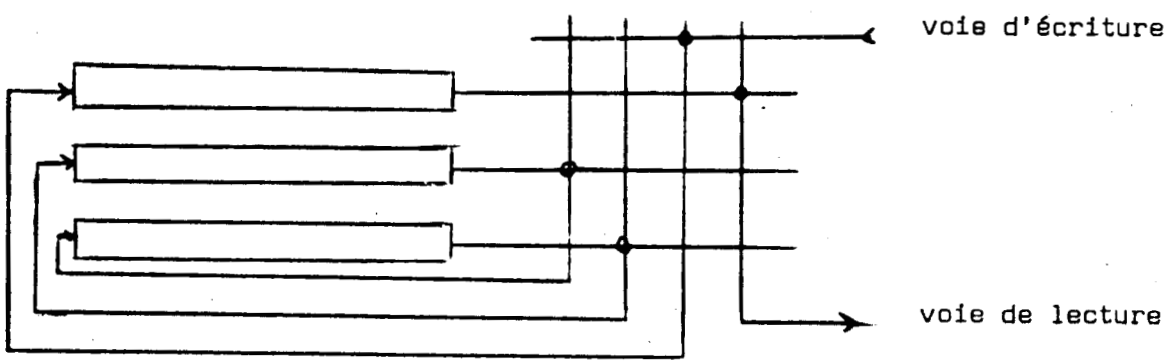


Fig. 3.5. Mémoire à configuration matricielle.

i pour la mémoire d'instructions, elle permet de limiter l'attente après un saut : une boucle de longueur l, avec $2^n \leq \ell < 2^{n-1}$ s'exécutera sur les n pages de plus bas niveau, le nombre d'instructions non utiles à faire circuler sera inférieur à 2^{n-1}.

ii pour la mémoire de données, le fait de disposer d'un nombre de fenêtres croissant lorsqu'on se rapproche du point d'exécution est très favorable ; le principe de localité nous dit en effet que la probabilité de devoir accèder à une page (donc d'avoir une donnée à réinsérer) décroît lorsque la distance au point d'exécution augmente.

La mémoire de réinsertion a alors une charge proche de l'optimale : la probabilité d'avoir à ranger une donnée dans la page n varie comme l'inverse de la longueur de la page, donc avec la probabilité d'apparition de son emplacement à une fenêtre d'accès.

III.2.3.2 Configuration circulaire.

Cette configuration, dite aussi "saute-mouton" a été adoptée dans MAUD [LEC 78]. Elle consiste à faire circuler l'ensemble de la mémoire dans une boucle en autorisant une (ou plusieurs) page(s) à s'isoler : elle(s) boucle(nt) alors sur elle(s)-même. Les pages sont de taille égales.

III.2.3.3 Configuration matricielle

Cette configuration permet de faire succéder à chacune des n pages (plus une ou plusieurs lignes d'écriture) l'une quelconque d'entre elles (plus une ou plusieurs lignes de lecture).

Cette flexibilité est acquise au prix d'une difficulté technologique de réalisation lorsque le nombre de pages devient important. Il faut pour n pages prévoir pour tous les cas (n-1)! matrices de permutations distinctes.

...

III.3 DISTRIBUTION DU CONTROLE ENTRE LES FLUX DE DONNEES ET D 'INSTRUCTIONS

III.3.1

Mode de fonctionnement Le fait d'imposer une programmation structurée amène à implémenter, au niveau de l'architecture, un système de traitement des boucles qui rende leur exécution automatique et qui permette aux flux de se synchroniser mutuellement.

Le processeur de commande de flux dispose :

- du compteur ordinal C.O
- de registres généraux, affectés par les ordres de commande du flux
- d'un registre de condition T, qui permet de réaliser le "case" et le "goto calculé" par

$$CO \leftarrow (CO) + (T)$$

- d'un registre inhibiteur de lecture (RIL) qui permet de présenter N fois (avec N = (RIL)) l'information contenue dans le buffer au(x) circuit(s) d'exécution. le contenu de RIL étant décrémenté à chaque cycle machine jusqu'à son annulation ; une fois nul, il devient inopérant

- d'une triple Pile (Pile (A, L, C)) qui permet l'emboitement des boucles, et qui contient

  i l'adresse de la première information de la boucle
  ii la longueur maximum de la boucle
  iii la condition de bouclage

  (cf. fig.3.6)

Le programme en langage machine est structuré, on autorise exclusivement les types de boucles suivant :

...

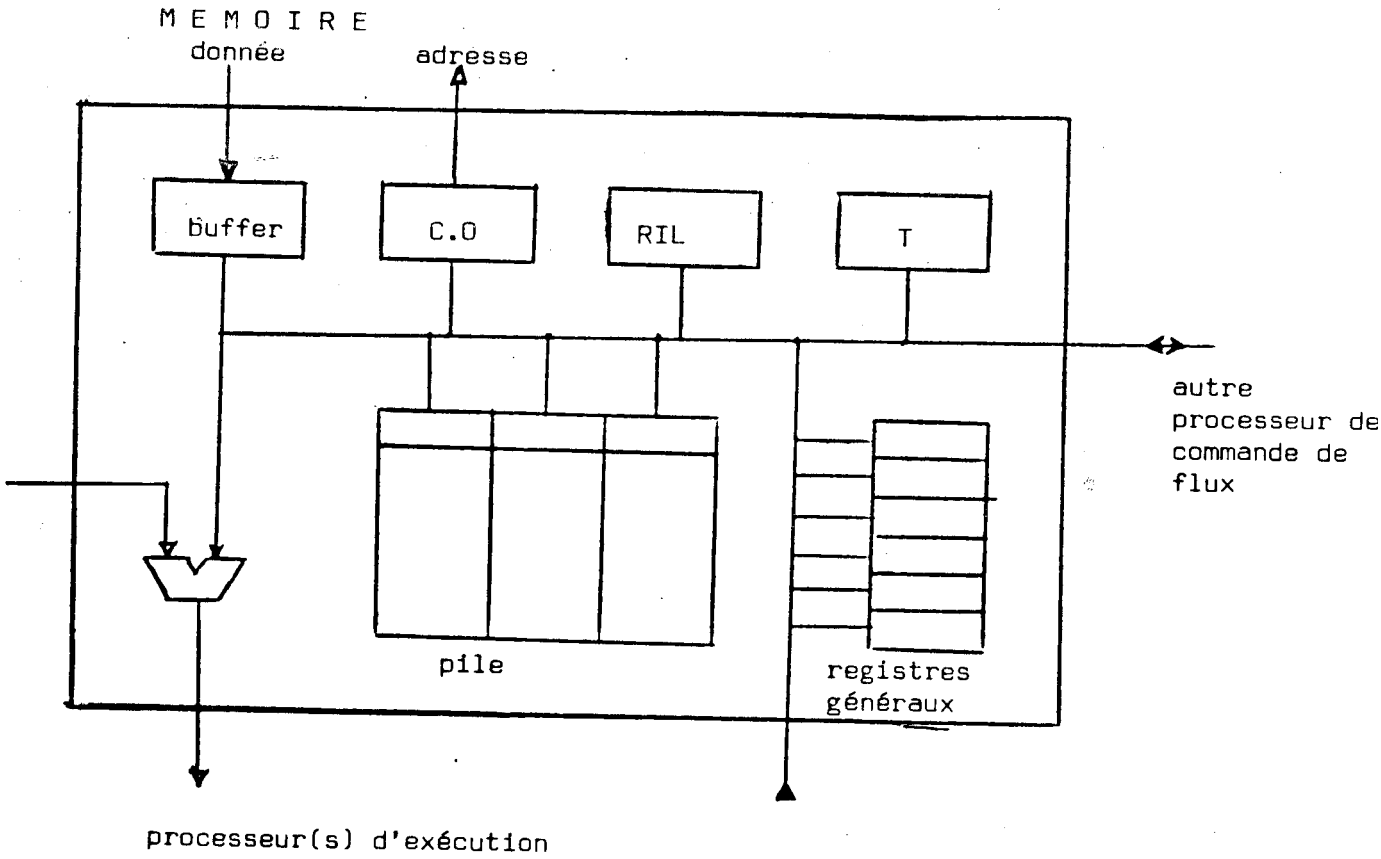


Fig. 3.6. Processeur de commande de flux

- répéter n fois
- répéter tant que
- répéter jusqu'à ce que

On ne sort d'une boucle que par une fin normale ou par un ordre "exit n", ou n précise le nombre de boucles dont on sort simultanément (boucles emboitées) [NOL 73].

Dans le langage de contrôle du flux, une boucle apparait comme un ordre :

répéter, type, longueur

Dans la cas d'une boucle "répéter n fois", la valeur de n est chargée dans le troisième mot de la pile, la longueur dans le second, et la valeur du C.O. dans le premier. A chaque fin de boucle, on effectue :

pile (C) ← [Pile (C)] - 1

si [Pile (C)] ≠ 0 alors CO ← [Pile (A)]
sinon CO ← [Pile (A)] + [Pile (B)] + 1
dépiler d'un cran

Les boucles "tant que" et "jusqu'à ce que" fonction de la même manière, en chargeant la condition explicite de la boucle dans Pile (C) a chaque exécution de la boucle.

L'exécution de EXIT n consiste à dépiler n fois, et à faire :

CO ← [Pile (A)] + [Pile (L)] + 1

III.3.2
Application aux traitement de nature vectorielle

La possibilité pour un processeur de commande de flux d'accèder aux registres d'un autre permet de réaliser avec une grande souplesse des traitements de nature vectorielle.

...

Le produit vectoriel de deux vecteurs de p composantes s'exécute de la façon suivante :

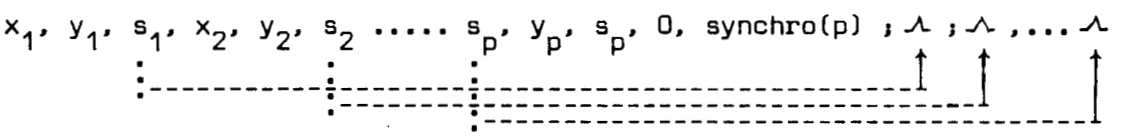
> on répéte p fois la boucle qui calcule les produits, puis la somme des produits est effectuée grâce à un ordre de synchronisation dans le flux de données, qui, à l'aide du registre RIL du P.C.F.I., fait répéter p fois l'instruction d'addition (cf. fig. 3.7)

La méthode de Gauss de résolution d'un système linéaire de dimension n comporte n-1 étapes ; à l'étape i, les n-i lignes suivantes sont modifiées de façon à trianguläriser la matrice. Les flux correspondants sont décrits à la figure 3.8.

Flux d'instruction

répéter (p fois, longueur 3) ; LDA ; MULT ; STA ; ⌐ ; ADD ; STA

Flux de données



avec synchro (p) : RIL (PCFI) ← P

Fig. 3.7. Multiplication de deux vecteurs

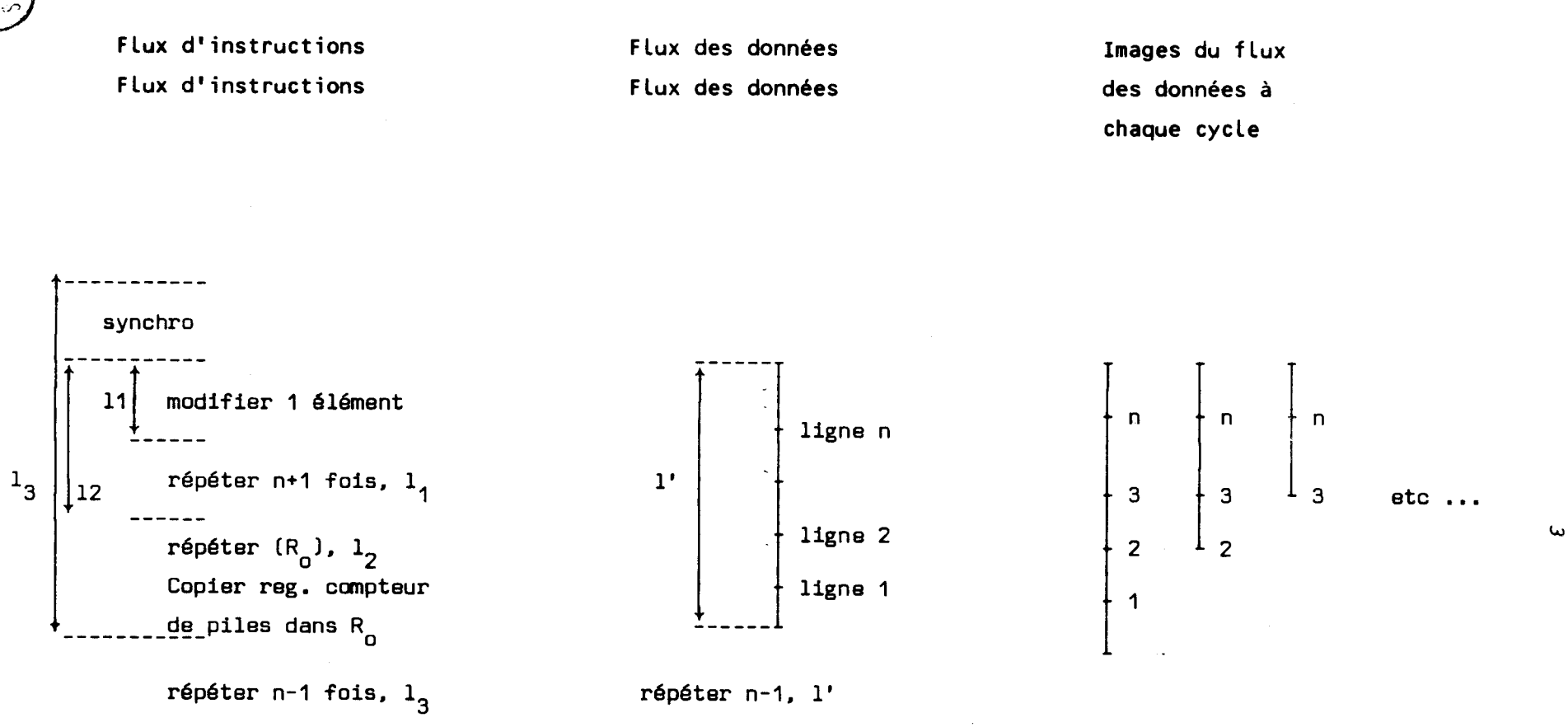


Fig. 3.8. Méthode de Gauss

CHAPITRE IV

EVALUATION ET SIMULATION

Pour définir les caractéristiques et les performances que l'on peut attendre de la machine à double flux, il faut étudier des problèmes réels qui donneront deux types de renseignements. La partie évaluation consiste à déterminer les caractéristiques des ruptures de séquences pour pouvoir fixer certains paramètres de l'architecture proposée. La partie simulation consiste à comparer les temps d'exécution de programmes types pour une machine classique et pour une machine à double flux avec mémoire à accès aléatoire. Pour situer la comparaison à un niveau qui le justifier, il faut que la nature, la complexité et le coût estimé des deux hypothèses soient du même ordre de grandeur. De ce qui a été dit plus haut, il ressort que l'architecture proposée trouve sa justification en bas de gamme, on conviendra donc de prendre comme référence un microprocesseur classique très répandu : le 8080 d'Intel.

IV.1 EVALUATION

Dans le cas d'une machine à flux simultanés utilisant des mémoires circulantes, on a vu que les ruptures de séquences pouvaient augmenter de façon critique les temps d'exécution des programmes. Il faut adapter la taille des pistes de manière à minimiser les temps d'attente lors des sauts (sauts en arrière principalement). Pour cela, un programme a été mis au point, il exécute les programmes observés en relevant les sauts de tous types (voir listing en annexe).

Pour chaque saut, on incrémente le compteur de saut (N) et on adjoint dans un fichier la "distance" (d_i) du saut : différence entre les valeurs du compteur ordinal après et avant l'exécution du saut.
Un compteur de pas de programme est également incrémenté à chaque exécution d'une instruction (NEX).

Une mesure du nombre d'instructions (NI) du programme est également faite pour étudier une éventuelle correlation entre la taille d'un programme, le nombre de sauts, et la distance qu'ils parcourent dans le programme.

Les entrées/sorties ne sont pas prises en compte, car jugées non significatives.
Une fois les données recueillies, on calcule sur

$$D = \{d_i\} \qquad (N = \mathrm{Card}(D))$$

les valeurs suivantes :

...

- nombre moyen d'instructions en séquence : $R = \frac{NEX}{N}$

- nombre de sauts en avant : $N+ = Card(D+)$

avec $D+ = \{d_i \mid\mid d_i > 0\}$

- nombre de sauts en arrière : $N- = Card(D-)$

avec $D- = \{d_i \mid\mid d_i < 0\}$

- moyenne géométrique des sauts : $GM = \sqrt{\frac{\sum_{D} a_i^2}{N}}$

- moyenne arithmétique des sauts : $D = \frac{\sum_{D} d_i}{N}$

- écart-type $\sigma = \sqrt{\frac{\sum_{D} (d_i - D)^2}{N}}$

- moyenne des sauts positifs : $D+ = \frac{\sum_{D+} d_i}{N+}$

- écart-type sur D_+ : $\sigma+ = \sqrt{\frac{\sum_{D+} (d_i - D+)^2}{N+}}$

...

- moyenne des sauts négatifs : $D^- = \frac{\sum D^- d_i}{N^-}$

- écart-type sur D- : $\sigma^- = \sqrt{\frac{D^- \cdot \cdot_i}{N^-}}$

Les programmes étudiés sont d'une part un programme mathématique comprenant une multiplication de matrices 5 × 5 d'autre part un programme de M_i tri (voir listings en annexe).

Pour obtenir des résultats exploitables, il a fallu faire des hypothèses sur l'ensemble D : un grand nombre de sauts correspondent à des distances très supérieures à la longueur du programme NI (nombre de mots machine). Ceci s'explique par l'implantation (aléatoire et non contigüe) du programme objet, et des routines qu'il utilise dans l'espace mémoire. Pour ne pas faire intervenir ces valeurs non significatives dans les calculs, on a choisi deux hypothèses de travail.

1 L'hypothèse optimiste (dite d_{min}) consiste à considérer que les procédures utilisées par le programme sont implantées à l'endroit où elles sont utilisées.

exemple

programme source

```
début
  .
  .
  .
A = sin (B)
  .
  .
  .
fin
```

programme objet

```
début
  .
  .
  .
procédure sinus
  début
  .
  fin
  .
fin
```

...

On travaillera alors sur l'ensemble D' suivant :

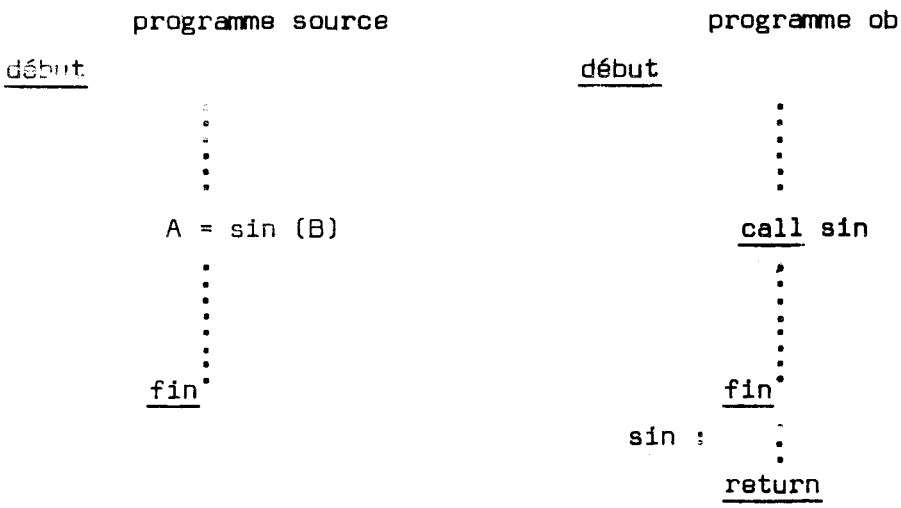
$$D' = \{d_i \;||\; \forall d_j \in D,\ |d_j| > NI \Longrightarrow d_i = 0$$

$$|d_j| \leq NI \Longrightarrow d_i = d_j\}$$

Les distances relevée qui sont supérieures à la longueur du programme sont affectées d'une valeur nulle.

2 L'hypothèse pessimiste (dite dmax) consiste à dire que les procédures appelées par le programme sont implantées immédiatement à la suite de celui-ci. L'ordre de grandeur du saut à effectuer dans le cas d'un appel est alors NI

exemple



L'ensemble D' sur lequel sont effectués les calculs est alors :

$$D' = \{d_i \;||\; \forall d_j \in D,\ |d_j| > NI \Longrightarrow d_i = NI$$

$$|d_j| \leq NI \Longrightarrow d_i = d_j\}$$

...

Cette seconde hypothèse donne donc une limite supérieure aux valeurs calculées. Le cas d'un système monoprogrammé pour lequel l'implémentation en mémoire est judicieusement faite devra être un intermédiaire entre ces deux extrêmes.

Les résultats correspondant aux deux programmes sont présentés aux figures 4.1 et 4.2.

Pour le second programme, deux versions sont présentées, ainsi que plusieurs passages pour chacun d'eux : le programme comportait un sous-programme (CHG, qui intervertit l'ordre de deux éléments). Pour mettre en évidence l'influence de l'utilisation des sous-programmes, on a fait des passages "avec call" puis des passages "sans call" pour lequels le corps du sous-programme a été inséré dans le programme principal.

L'analyse des résultats met en évidence :

1 la fréquence relative des ruptures de séquence, assez stable (une pour 4,4 à 6,2 instructions) ce qui recoupe d'autres analyses. Ce fait est défavorable pour l'utilisation des mémoires circulantes comme mémoire programme au niveau de l'exécution

2 les sauts avant et arrière ne présentent pas les même caractéristiques. dans le programme mathématique, les sauts arrière sont nettement plus nombreux (près de 3 fois) ; la structure de boucle est en partie conservée au niveau du langage machine, ce qui donne une valeur de D négative. Pour le programme de tri, ce sont les sauts avant qui sont les plus nombreux (le programme comportait une instruction IF) mais la proportion décroit avec le nombre de données traitées (la valeur de D, positive pour un petit nombre de données, devient négative à partir de 50 données)

3 le paramètre important dans le cas de l'utilisation de mémoire série est celui qui précise les caractéristiques des sauts arrière : on peut en effet prévoir, autour du point d'exécution, un accès aléatoire aux instructions, si le saut arrière dépasse la longueur (l) de ce tampon, il faut revenir à la fenêtre précédente (L décalages) (fig. 4.3)

...

NI = 46

	d min	d max
N EX	3992	
N	723	
R	5.5	
N+	130	
N-	366	
GM	9.5	24.4
D, σ	- 6.2, 8.8	- 4.2, 24.0
D+, σ +	5.0, 3.8	21.4, 17.7
D-, σ -	- 10.2, 7.9	- 17.2, 14.4

Fig. 4.1. Résultats statistiques pour le programme mathématique

Tri (NI = 40)

Nombre d'éléments		d min sans call	d min avec call	d max sans call	d max avec call
10	NEX	3086	3868		
	[illegible]	586	839		
	R	5.3	4.6		
	N+	284	376	409	547
	N-	82	151	177	292
	GM	12.0	11.0	28.3 ; 15.0	26.7
	D,σ	4.55 ; 11.8	4.22 ; 10.8	6.10 ; 27.6	3.96 ; 26.4
	D+,σ+	11.7 ; 7.9	10.0 ; 9.1	21.7 ; 15.0	19.6 ; 15.5
	D-,σ-	- 18.5 ; 6.2	- 11.7 ; 7.2	- 30.0 ; 11.5	- 25.4 ; 15.1
20	NEX	7076	10612		
	N	1247	2391		
	R	5.7	4.4		
	N+	540	1060	816	1440
	N-	257	569	31	951
	GM	12.2	10.7	26.9	25.0
	D,σ	2.10 ; 12.1	2.86 ; 10.5	4.61 ; 26.5	1.93 ; 24.9
	D+,σ+	11.4 ; 7.5	9.8 ; 9.5	21.1 ; 14.8	17.8 ; 15.9
	D-,σ-	- 17.4 ; 4.9	- 10.1 ; 5.8	- 26.5 ; 11.7	22.1 ; 15.3
30	NEX	12300	19202		
	N	2128	4361		
	R	5.8	4.8		
	N+	921	1936	1334	2552
	N-	532	1141	794	1809
	GM	12.5	10.7	25.8	24.2
	D,σ	.79 ; 12.5	2.28 ; 10.6	3.38 ; 25.5	1.13 ; 24.2
	D+,σ+	11.3 ; 7.1	9.6 ; 9.4	20.2 ; 14.5	16.9 ; 15.4
	D-,σ-	- 17.3 ; 4.0	- 10.1 ; 5.4	- 24.8 ; 11.1	- 21.1 ; 15.1
50	NEX	26290	44310		
	N	4323	10153		
	R	6.1	4.4		
	N+	1831	4481	2512	5692
	N-	12.7	2981	1811	4461
	GM	- 1.06 ; 12.7	10.6	23.9	23.2
	D,σ	10.9 ; 6.5	1.60 ; 10.5	1.54 ; 23.9	.08 ; 23.2
	D+,σ+	- 16.9 ; 3.2	9.2 ; 9.5	18.8 ; 14.1	15.7 ; 15.2
	D-,σ-		- 9.8 ; 4.8	- 22.4 ; 10.2	- 20.0 ; 14.9
100	NEX	80784			
	N	12802			
	R	6.2			
	N+	5352		6072	
	N-	5256		6099	
	GM	13.0		21.1	
	D,σ	- 2.98 ; 12.8		0.88 ; 21.0	
	D+,σ+	10.2 ; 5.4		12.2 ; 12.9	
	D-,σ-	- 16.4 ; 2.3		19.7 ; 8.4	

Fig. 4.2. Résultats statistiques pour le programme de tri

La valeur qui donne 1 pour minimiser cas cas de défaut de pages est $|D-| + \sigma-$ (si la loi est normale, la distance d_i d'un saut arrière est inférieure à cette valeur dans 68 % des cas).
Dans les exemples traités, cette valeur est de l'ordre de NI dans l'hypothèse d_{max}, et de NI/2 dans l'hypothèse d_{min}. Ces résultats peu favorables sont dus à la faible valeur de NI. Dans des programmes plus importants, le principe de localité doit amener des valeurs proportionnellement plus faible. (Les programmes étudiés doivent être considérés comme des routines).

4 La comparaison entre les résultats "sans call" et "avec call" met en évidence la pénalisation du point de vue des ruptures de séquences qu'apporte l'utilisation des sous-programmes : le nombre de sauts est doublé (tant les sauts arrière que les que les sauts avant). La pénalisation est moins forte pour le nombre de pas exécutés, mais reste importante).

Pour une machine à mémoire série, les sauts sont très pénalisants, il convient donc de réduire le nombre de sous-programmes, et de leur préférer, au niveau du langage d'assemblage, les macro-instructions, quitte à augmenter l'encombrement mémoire d'un programme.

IV.2 SIMULATION

La réalisation d'une machine à double flux doit permettre une amélioration des caractéristiques globales par rapport à une machine classique. Cette amélioration peut se situer sur deux plans :

- soit sur le coût de la machine qui peut être nettement inférieur à capacité égale, grâce à l'utilisation de mémoires circulantes,
- soit, à coût comparable, par l'augmentation du débit (throughput).

...

Cette seconde version nécessite l'usage de mémoires à accès aléatoire, du moins à un niveau proche de l'exécution. On prendra comme architecture celle qui a été définie au §.3.2.1. On étudiera d'abord sur deux exemples les temps d'exécution comparés d'un microprocesseur existant (I 8080) avec ceux du processeur à double-flux. Cela permettra de mettre en évidence le gain en temps machine et les types d'instructions sur lesquelles portent ce gain.

Le premier exemple traité est un programme de génération d'un nombre pseudo-aléatoire (fig. 4.4). On trouve en regard, d'une part le programme en assembleur 8080, d'autre part sa traduction en assemble "double flux". Les mnémoniques utilisés sont inspirés de l'article de Nicoud [**NIC 76**].

On obtient un gain en temps d'exécution de 50 % avec un encombrement mémoire à peine supérieur.

Le second exemple est celui de la multiplication de deux octets (fig. 4.5).

Le gain en débit est là encore significatif : près de 40 %. On serait tenté de le mettre au compte de l'augmentation de la taille du champ instruction : en fait le compte du nombre d'octets montre que l'encombrement est comparable, dans les deux cas, c'est donc bien le débit global qui est augmenté.

Pour vérifier la valeur de ce gain. On établit une table de correspondance (fig. 4.6) entre le nombre de cycles des instructions 8080 et celui des instructions "double flux". Les hypothèses qui ont permis d'élaborer cette table sont les suivantes :

- les instructions de type MOV reg, Mem et MOV reg, reg disparaissent (le traitement est assuré par le champ d'insertion),

- les instructions à opérandes immédiat ou direct sont aussi rapides que les instructions sur registre (un seul accès mémoire), les instructions de saut sont plus rapides pour la même raison, la plupart des autres instructions sont affectées de la même durée,

...

ligne	adresse				nombre de cycles
1	1		MVI	H,ØØ H	7
2	3		MVI	L, ØFF H	7
3	5		MOV	C, H	5
4	6		MOV	A, M	7
5	7		CMP	H	4
6	8		JNZ	SKIP	10
7	11		MOV	A, L	(5)
8	12	SKIP :	MOV	B, A	5
9	13		ANI	1DH	7
10	15		JPE	PAR	10
11	18		MVI	C, 80 H	(7)
12	10	PAR :	MOV	A, B	5
13	21		RRC		4
14	22		ANI	7 FH	7
15	24		ADD	C	4
16	25		MOV	M, A	7
			RET		
	25 octets				101 cycles

8080

ligne	MI	MD	MC	nombre de cycles
1	LOAD	0	+9	5
2	LDA	RAN	+6	5
3	JNZ	+2	Ø	5
4	LOAD	FFH	+4	(5)
5	ANI	1DH	Ø	4
6	JPE	+2	Ø	5
7	LOAD	80H	+3	(4)
8	RR	A	Ø	4
9	ANI	7FH	Ø	4
10	ADD	A	-8	4
11	STA	Ø	+x	5
	27 octets (Ø non compris)			50 cycles

Machine à flux simultanés

Fig. 4.4. Programme de génération de nombre pseudo-aléatoire

ligne	adresse			nombre de cycles	ligne	MI	MD	MC	nombre de cycles	nombre d'exécutions
1	1		MVI D, MUR	7	1	LOAD	MUR	+8	5	1
2	3		MVI C, MUD	7	2	LOAD	MUD	+3	5	
3	5	MULT :	MVI B, 0	7	3	LOAD	0	+5	5	1
4	7		MVI E, 9D	7	4	LOAD	9	+2	5	
5	9	MULTO:	MOV A, C	5	5	RR	A	0	4	9
6	10		RAR	4						
7	11		MOV C, A	5						
8	12		DCR E	5	6	DCR	A	0	5	
9	13		JZ DONE	10	7	JZ	+5	Ø	5	
10	16		MOV A, B	4						
11	17		JNC MULT1	10	8	JNC	+2	Ø	5	8
12	20		ADD D	4	9	ADD	B	*D	4	4 (1)
13	21	MULT1:	RAR	4	10	RR	A	*-3	4	
14	22		MOV B, A	5	11(2)	STORE	Ø	+X	5	8
15	23		JMP MULTO	10	12	JMP	-7	Ø	5	
		DONE:								
		577		temps d'exécution (cycles)					354	
		25		nombre d'octets "programme"					28	
		139		nombre d'octets "exécutés"					132	

(1) cette ligne est exectué de 0 à 8 fois
(2) cette instruction peut être redondante : on peut envisager d'écrire
11 JMP -6 +X
car l'instruction de saut n'affecte pas le contenu de l'accumulateur

Fig. 4.5. Execution de la multiplication de deux octets par le 8080 et par la machine à flux simultanés

	nombre de cycles pour le 8080	pour le M 5
MOV r_1, r_2	5	0
MOV M, r r, M	7	0
HLT	7	7
MVI r	7	0
MVI M	10	0
INR DCR M	10	5
instructions arithmétiques et logiques		
- sur registres	4	4
- sur mémoire	7	4
- immédiates	7	4
sauts	10	5
CALL	17	12
call conditionnels	11/17	6/12
RET	10	10
return conditonels	5/11	5/11
RST	11	11
IN/OUT	10	7
LXI, paire de registres	10	10
PUSH, paire de registres	11	11
POP, paire de registres	10	10
STA/LDA	13	0/5
XCHG	4	4
XTHL	18	18
SPHL/PCHL	5	5
DAD, paire de registres	10	10
STAX/LDAX, paire de registres	7	7
INX/DCS, paire de registrus	5	5
SHLD/LHLD	16	0/10
[illegible]	4	4

Fig. 4.6. Correspondance entre les temps machines (exprimés en nombre de cycles) pour le 8080 et la machine à flux simultanés avec mémoire à accès.

de manière à fournir une limite inférieure du gain (en particulier pour les instructions sur paires de registres qu'il n'est pas possible de "traduire" directement).

Cette table a permis d'évaluer les temps d'exécutions de differents programmes tirés de la bibliothèque du 8080 ; tous les résultats concordent et donne un gain de l'ordre de 50 %.

CONCLUSION

La machine à flux simultanés, dans les différentes options qui sont proposées peut couvrir un grand nombre d'applications.

Rappelons d'abord ses limites : ce n'est pas une machine universelle, et elle ne se prête pas à la multiprogrammation mais, avec la baisse des coûts du matériel, cela n'est pas génant pour bien des applications, en particulier à l'échelle des µP.

Elle n'est pas destinée à exécuter de nombreux programmes, leur mise au point étant moins aisée que sur une machine classique : son langage est difficilement interprétable.

...

Par contre sa justification apparait parfaitement dans les cas suivants :

- en bas de gamme, elle peut être une machine spécialisée dans le traitement de données de divers type (processus, texte, signal, ...)

- en haut de gamme, la machine à flux simultanés symétriques (avec contrôle indépendant de chacun des flux) peut être appliquée à des calculs vectoriels complexes.

L'approfondissement de cette étude peut se faire dans les directions suivantes :

i l'analyse des structures de contrôle et leur répartition dans les différents flux, la définition d'un langage machine structuré et à utilisation unique des données, ainsi que les problèmes de compilation associés à ces notions

ii une analyse statistique plus approfondie des déroulements de programmes des points de vue du point des données (réinsertions) et de celui des instructions (ruptures de séquence)

iii la simulation sur des problèmes de natures diverses, qui passe par l'écriture d'un compilateur

iv la réalisation d'un prototype, utilisant des microprocesseurs bipolaires, qui pourrait être de l'architecture proposée au §.3.2.1

v la définition d'une architecture multiprocesseurs mettant en oeuvre ces principes avec de nombreux flux en parallèle.

*

* *

BIBLIOGRAPHIE

[AHO 74] **A.V. AHO, S.D. ULLMANN**
Dynamic memories with fast random and sequential acces
IEE T.O.E.C, C.23, 1974

[ALT 2/78] **L. ALTMAN**
Digital desing : scaling new heights
Electronics, 16 feb. 1978

[ALT 4/78] **L. ALTMAN**
New arrivals in the bulk storage inventory
Electronics, 13 apr. 1978

[ARM 77] **L. ARMSTRONG**
The CCD's future takes on a bright hue
Electronics, 10 nov. 1978

...

[ARN 78] **W.F. ARNOLD**

Memory Makers brace for bubble battle

Electronics, 16 feb. 1978

[ARV 77] **ARVIND, K.P. GOSTELOW**

A computer capable of exchanging processors for time

IFIP 1977, pp. 849-853

[BAT 77] **C. BATTAREL**

Mémoires à propagation de domaines plans

Technique de l'Ingénieur, Dec. 1977

[BEL 78] **G. BELL, S.H. FULLER & D. SEWIAREK**

Optimal shift strategy for a block-Transfert CCD Memory

CACM, V.21, n° 3, May 1978

[BHA 75] **D.P. BHANDARKAR, J.E. JULIUSSEN**

Tutorial : computer system advantages of Magnetic bubble Memories

Computer, nov. 75

[BOR 77] **J. BOREL**

Mémoires à semi-conducteur

Technique de l'Ingénieur, Dec 1977

[COR 78] **V. CORDONNIER**

L'emploi des mémoires circulantes dans l'architecture des ordinateurs

Rapport technique USTL, Dépt. Informatique, janv. 1978

[COM 74] **D. COMTE, G. DURRIEU, O. GELLY, A. PLAS & J.C. SYRE**

Etude et spécification d'une architecture de calculateur exploitant le concept d'assignation unique

Rapport contrat SESORI 74167, Vol.7, LAU system

[DAU 77] **M. DAUCHET**

Flux d'informations dans les ordinateurs

7ème Thèse, Université de Lille 1, 1977.

[DEN 74] **J.B. DENNIS, D.P. MISUNAS**

A preliminary architecture for a basic data-flow processor

Proc. 3 annual ACM-SIGARCH-IEEE symposium on Computer architecture 3.4 dec. 1974

[FLY 72] **M.J. FLYNN**

Some computer organization and their effectiveness

IEE T.O.E.C, Vol C.21, n° 9, sept. 1972

[GAL 77] **M. GALINIER**

La programmation structurée

Technique de l'Ingénieur, Dec. 1977

[GLI 77] **P. GLIZE**

Etude et mesure de performances de mémoires centrales séquentielles

Thèse de Docteur de spécialité, Université de Toulouse, mars 1977

[GUE76] **J.P. GUENIN**

Gestion d'une mémoire à couplage de charge

Minis et micros, sept. 1976

[HAN 77] **W. HANDLER**

On classification schemes for computer systems in the pos-Von-Neumann Era

Support cours IRIA, Traitement Parallèle, 31 jan. - 3 fév. 1977

[HOE 76] **S. HOENER, W. ROEHDER**

Modular multi-microprocessor architecture with virtual memory

Euromicro, Symposium on micro-architecture

...

[JIN 78] **M. JINO, J.W.S. LIN**
Intelligent magnetic bubble memories
Sigarch newsletter, Vol. 6, n° 7, apr. 1978

[INT 77] **INTEL Corp.**
16 K CCD serial memory
Data catalog, 1977

[JUL 76] **J.E. JULIUSSEN**
Magnetic Bubble systems approach practical use
Computer design, oct. 1976

[JUL 77] **J.E. JULIUSSEN, D.M. LEE & G.M. COX**
Bubbles appearing first as micorprocessor mass storage
Electronics, aug. 1977

[KAR 76] **S.I. KARTASHEV**
A microcomputer with a shift register memory
IEEE T.O.E.C, Vol. C.25, n° 5, may 1976

[KEL 75] **J. KELLY**
Development of an experimental Electron-beem-adressed memory module computer
Computer, Feb. 1975

[LEC 78] **M.P. LECOUFFE**
Machine d'assignation unique dynamique : MAUD
Publication n° 116, sept. 1978

[LEN 77] **J. LENFANT**
Fast random and sequential acces to dynamic memories of any size
IEEE T.O.E.C, Vol. C.26, N° 9, sept. 1977

...

[LIL 78] **H. LILEN**
Les mémoires intégrées
Editions Radio, 1978

[MAR 77] **E. MARSHALL**
Les mémoires à bulles magnétiques
0.1 Informatique hebdo., 21 nov. 1977

[MAR 2/78] **E. MARSHALL**
Les mémoires à faisceaux d'élections
Minis et micros, 3 fev. 1978

[MAR 4/78] **E. MARSHALL**
Mémoires à bulles : IBM.. et les autres
0.1 Informatique hebdo, 10 avr. 1978

[MON 77] **Ph. MONIN**
Burroughs devoile son géant : le BSP, 1er ordinateur doté d'une mémoire CCD
0.1 Informatique hebdo., 16 mai 1977

[MOT 77] **MOTOROLA**
MC 14500 Industrial Control Unit
Handbook motorola, 1977

[MYE 77] **W. MYERS**
Current development in magnetic bubble technology
Computer, aug. 1977

[NIC 76] **J.D. NICOUD**
A common microprocessor assembly language
2nd symposium on micro-architecture, Euromicro 1976

...

[NOL 73] **L. NOLIN, G. RUGGIU**
A formalization of EXEL
Sigact-Siplan Symposium on the principles of programming languages, Boston 1973

[NUT 77] **G.J. NUTT**
Memory and bus conflict in an array processor
IEEE T.O.E.C, Vol. C.26, n° 6, jun 1977

[PAN 77] **G. PANIGRAHI**
The implications of electronic serial memories
Computer, jul. 1977

[PEY 77] **J.F. PEYRUCAT**
Les mémoires à semi-conducteurs
Toute l'Electronique, oct. 1977

[RAM 77] **I. RAMPIL**
A Floppy disk tutorial
Byte, Dec. 1977

[RAN 77] **D. RANDET**
Technologie des mémoires
Techniques de l'Ingénieur, 1977

[RAN 9/77] **D. RANDET**
Mémoires à enregistrement magnétique. Nouveaux types de mémoires : études et recherches
Technique de l'Ingénieur, sept. 1977

[SCR 78] **S.E. SCRUPSKI**
NCC : memories are growing fast, and so is japanese semi industry
Electronic Design, n° 11, 14 may 1978

...

[STO 78] H.S. STONE

Dynamic memories with fast random and sequential access

IEEE T.O.E.C, C.24, n°12, Dec. 1975

[SYR 76] J.C. SYRE, D. COMTE, G. DURRIEU, O. GELLY & A. PLAS

The LAU parallel system : software definition and implementation through a multi-processor architecture

Euromicro 1976

[THR 1/78] B. THREEWITT

64 K CCD Memory

Progress (Fairchild), jan/fev. 1978

[THR 6/78] B. THREEWITT

CCDs bring solid-state benefits to bulk storage for computers

Electronics, june 22, 1978

[TI 2/77] TEXAS INSTRUMENT

Magnetic Bubble Memories and system interface circuits

T.I. Tentative Data, feb. 1977

[TOU 78a] B. TOURSEL, P. VANLAER

A new organization of information streams for processing with shifting memories

Euromicro, 1978

[TOU 78b] B. TOURSEL, P. VANLAER

Architecture à flux simultanés symétriques

AFCET, 1978

[TOU 78c] B. TOURSEL

Régulation des flux d'informations dans un ordinateur et machines à flux simultanés symétriques

USTL, pub. interne, jan. 1978

...

[UNG 77] UNGARO

Processeur associatif pour base de données relationnelles

Journées du groupe AFCET, Conception des systèmes Informatiques et Automatiques, mars 1977

[WEN 75] J.H. WENSLEY

The impact of electronic disks on system architecture

Computer, feb. 1975

[WID 77] A. WIDORY, G. ROUCAIROL

Deux algorithmes d'aide à la segmentation de données d'un programme

RAIRO informatique, Vol. 11, n° 3, mars 1977

[M.WIL 77] M.H. WILLIAMS

Generating structured flow diagrams : the nature of unstructuredness

The Computer Journal, Vol. 20, n° 1, jan. 1977

[M.WIL 78] M.H. WILLIAMS, H.L. OSSHER

Conversion of instructured flow diagrams to structuredness

The Computer Journal, Vol. 21, n° 2, feb. 1978

[R.WIL 77] R.P. WILLIAMS

Serial integer Arithmetic with Magnetic bubbles

IEEE T.O.E.C, mars 1977

[WON 76] C.K. WONG, P.C. YUE

Data organization in Magnetic Bubble Lattice Files

IBM Journal of Research & Development, nov. 1976

[WON 77] C.K. WONG, D.T. TANG

Dynamic memories with faster random & sequential access

IBM Journal of Research & Development, may 1977

ANNEXES

programme espion

```
                                CSECT   1
                                SYSTEM  SIGSF
                                SYSTEM  FORTLIB
                    1,          SUBROUTINE DISTANCE(T,N,NEX)
00000 P  22 5 00003 A  DISTANCE LI,5     3
00001 P  6A 6 00000 X           BAL,6    9SETUPN
00002 P  01 0 00016 V           INTG     T
00003 P  01 0 00017 V           INTG     N
00004 P  01 0 00018 V           INTG     NEX
00005 P  35 F 00018 V           STW,15   1PTMP
00006 P  33 F 00016 V           MTW,-1   T              OFFSET DUMMY ARRAY
                    2,          INTEGER T(N)
                    3,          INTEGER PASS
                    4,          OUTPUT NEX
00007 P  22 7 0000B P           LI,7     1G
00008 P  6A 6 00000 X           BAL,6    9PRINT
00009 P  6A 6 00000 X           BAL,6    9IODATA
0000A P  81 2 00018 V           INTG,E   *NEX
0000B P  68 0 0000F P     1G    B        2G
0000C P    407D4005 A           TEXT     '('' N'
0000D P    5E7407F A            TEXT     'EX ='
0000E P    407DC75D A           TEXT     ' ''G)'
                    5,          NMAX=40
0000F P  22 9 00028 A     2G    LI,9     40
00010 P  35 9 00002 V           STW,9    NMAX
                    6,          PASS=1
00011 P  22 9 00001 A           LI,9     1
00012 P  35 9 00001 V           STW,9    PASS
                    7,    1000  CONTINUE
                    8,          IF(PASS.EQ.1) IVAL=0
00013 P  22 9 00001 A    1000$  LI,9     1
00014 P  31 9 00001 V           CW,9     PASS
00015 P  69 3 00018 P           BNE      3G
00016 P  22 9 00000 A           LI,9     0
00017 P  35 9 00003 V           STW,9    IVAL
                    9,          IF(PASS.EQ.2) IVAL=NMAX
00018 P  22 9 00002 A     3G    LI,9     2
00019 P  31 9 00001 V           CW,9     PASS
0001A P  69 3 0001D P           BNE      4G
```

```
0001B P   32 9 00002 V            LW,9     NMAX
0001C P   35 9 00003 V            STW,9    IVAL

                        10,       SM=SP=SC=0

0001D P   22 8 00000 A     4G     LI,8     0,0
0001E P   35 8 00006 V            STW,8    SC
0001F P   35 8 00005 V            STW,8    SP
00020 P   35 8 00004 V            STW,8    SM

                        11,       S1=S2=S3=0

00021 P   22 8 00000 A            LI,8     0,0
00022 P   35 8 00009 V            STW,8    S3
00023 P   35 8 00008 V            STW,8    S2
00024 P   35 8 00007 V            STW,8    S1

                        12,       NM=NP=0

00025 P   22 9 00000 A            LI,9     0
00026 P   35 9 00008 V            STW,9    NP
00027 P   35 9 0000A V            STW,9    NM

                        13,       DO 1 I=1,N

00028 P.  22 4 00001 A            LI,4     1
00029 P   35 4 0000C V            STW,4    I

                        14,       K=T(I)

0002A P   B2 9 80016 V     5G     LW,9     *T,4
0002B P   35 9 0000D V            STW,9    K

                        15,       IF(ABS(K),GT,NMAX) K=SIGN(IVAL,K)

0002C P   32 9 0000D V            LW,9     K
0002D P   6A 6 00000 X            BAL,6    9ITOR
0002E P   3B 8 00008 A            LAW,8    8
0002F P   35 8 00019 V            STW,8    1TEMP
00030 P   32 9 00002 V            LW,9     NMAX
00031 P   6A 6 00000 X            BAL,6    9ITOR
00032 P   31 8 00019 V            CW,8     1TEMP
00033 P   68 1 0003C P            BGE      7G
00034 P   32 9 00003 V            LW,9     IVAL
00035 P   6A 6 00000 X            BAL,6    9ITOR
00036 P   3B 8 00008 A            LAW,8    8
00037 P   33 0 0000D V            MTW,0    K
00038 P   68 4 0003A P            BGEZ     6G
00039 P   3A 8 00008 A            LCW,8    8
0003A P   6A 6 00000 X     6G     BAL,6    9RTOI
0003B P   35 9 0000D V            STW,9    K
```

```
                        16,             IF (K) 101,1,100

003C P   32 9 0000D V     7G            LW,9      K
0030 P   69 2 00045 P                   BGZ       100S
003E P   68 3 00052 P                   BEZ       1S

                        17,     101     CONTINUE
                        18,             SM=SM+K

003F P   32 9 0000D V     1013          LW,9      K
0040 P   6A 6 00000 X                   BAL,6     9ITOR
0041 P   3D 8 00004 V                   FAS,8     SM
0042 P   35 8 00004 V                   STW,8     SM

                        19,             NM=NM+1

0043 P   33 1 0000A V                   MTW,1     NM

                        20,             GOTO  200

0044 P   68 0 0004A P                   B         200S

                        21,     100     SP=SP+K

0045 P   32 9 0000D V     100S          LW,9      K
0046 P   6A 6 00000 X                   BAL,6     9ITOR
0047 P   3D 8 00005 V                   FAS,8     SP
0048 P   35 8 00005 V                   STW,8     SP

                        22,             NP=NP+1

0049 P   33 1 00008 V                   MTW,1     NP

                        23,     200     SC=SC+K*K

004A P   32 9 0000D V     200S          LW,9      K
0048 P   6A 6 00000 X                   BAL,6     9ITOR
004C P   35 8 00019 V                   STW,8     1TEMP
004D P   32 9 0000D V                   LW,9      K
004E P   6A 6 00000 X                   BAL,6     9ITOR
004F P   3F 8 00019 V                   FMS,8     1TEMP
0050 P   3D 8 00006 V                   FAS,8     SC
0051 P   35 8 00006 V                   STW,8     SC

                        24,     1       CONTINUE

0052 P   32 4 0000C V     1S            LW,4      I
0053 P   20 4 00001 A                   AI,4      1
0054 P   35 4 0000C V                   STW,4     I
0055 P   81 4 00017 V                   CW,4      *N
```

```
0056 P  68 2 0002A P            BLE     5G

                        25,     DP=SP/NP

0057 P  32 9 0000B V            LW,9    NP
0058 P  6A 6 00000 X            BAL,6   9ITOR
0059 P  35 8 00019 V            STW,8   1TEMP
005A P  32 8 00005 V            LW,8    SP
005B P  3E 8 00019 V            FDS,8   1TEMP
005C P  35 8 0000E V            STW,8   DP

                        26,     DM=SM/NM

005D P  32 9 0000A V            LW,9    NM
005E P  6A 6 00000 X            BAL,6   9ITOR
005F P  35 8 00019 V            STW,8   1TEMP
00060 P  32 8 00004 V           LW,8    SM
00061 P  3E 8 00019 V           FDS,8   1TEMP
00062 P  35 8 0000F V           STW,8   DM

                        27,     D =(SP+SM)/(NP+NM)

00063 P  32 9 0000A V           LW,9    NM
00064 P  6A 6 00000 X           BAL,6   9ITOR
00065 P  35 8 00019 V           STW,8   1TEMP
00066 P  32 9 0000B V           LW,9    NP
00067 P  6A 6 00000 X           BAL,6   9ITOR
00068 P  3D 8 00019 V           FAS,8   1TEMP
00069 P  35 8 0001A V           STW,8   2TEMP
0006A P  32 8 00005 V           LW,8    SP
0006B P  3D 8 00004 V           FAS,8   SM
0006C P  3E 8 0001A V           FDS,8   2TEMP
0006D P  35 8 00010 V           STW,8   D

                        28,     GM=SQRT((SC)/N     )

0006E P  B2 9 00017 V           LW,9    *N
0006F P  6A 6 00000 X           BAL,6   9ITOR
00070 P  35 8 00019 V           STW,8   1TEMP
00071 P  32 8 00006 V           LW,8    SC
00072 P  3E 8 00019 V           FDS,8   1TEMP
00073 P  6A 6 00000 X           BAL,6   9SQRT
00074 P  35 8 00011 V           STW,8   GM

                        29,     DO 2 I=1,N

00075 P  22 4 00001 A           LI,4    1
00076 P  35 4 0000C V           STW,4   I

                        30,     K=T(I)
```

```
00077 P   B2 9 B0016 V    BG        LW,9     *T,4
00078 P   35 9 0000D V              STW,9    K

                          31,       IF(ABS(K),GT,NMAX) K=SIGN(IVAL,K)

00079 P   32 9 0000D V              LW,9     K
0007A P   6A 6 00000 X              BAL,6    9ITOR
0007B P   3B 8 00008 A              LAW,8    8
0007C P   35 8 00019 V              STW,8    1TEMP
0007D P   32 9 00002 V              LW,9     NMAX
0007E P   6A 6 00000 X              BAL,6    9ITOR
0007F P   31 8 00019 V              CW,8     1TEMP
00080 P   68 1 00089 P              BGE      10G
00081 P   32 9 00003 V              LW,9     IVAL
00082 P   6A 6 00000 X              BAL,6    9ITOR
00083 P   3B 8 00008 A              LAW,8    8
00084 P   33 0 0000D V              MTW,0    K
00085 P   6B 1 00087 P              BGEZ     9G
00086 P   3A 8 0000B A              LCW,8    8
00087 P   6A 6 00000 X    9G        BAL,6    9RTOI
00088 P   35 9 0000D V              STW,9    K

                          32,       S1=S1+(K-D)*(K-D)

00089 P   32 9 0000D V    10G       LW,9     K
0008A P   6A 6 00000 X              BAL,6    9ITOR
0008B P   3C 8 00010 V              FSS,8    D
0008C P   35 8 00019 V              STW,8    1TEMP
0008D P   32 9 0000D V              LW,9     K
0008E P   6A 6 00000 X              BAL,6    9ITOR
0008F P   3C 8 00010 V              FSS,8    D
00090 P   3F 8 00019 V              FMS,8    1TEMP
00091 P   3D 8 00007 V              FAS,8    S1
00092 P   35 B 00007 V              STW,8    S1

                          33,       IF (K) 250,2,300

00093 P   32 9 0000D V              LW,9     K
00094 P   69 2 000A1 P              BGZ      300S
00095 P   68 3 000AB P              BEZ      2S

                          34,  250  CONTINUE
                          35,       S2=S2+(K-DM)*(K-DM)

00096 P   32 9 0000D V    250S      LW,9     K
00097 P   6A 6 00000 X              BAL,6    9ITOR
00098 P   3C B 0000F V              FSS,8    DM
00099 P   35 8 00019 V              STW,8    1TEMP
0009A P   32 9 0000D V              LW,9     K
0009B P   6A 6 00000 X              BAL,6    9ITOR
0009C P   3C B 0000F V              FSS,8    DM
```

```
00009D P   3E 8 00019 V                FMS,8    1TEMP
00009E P   3D 8 00008 V                FAS,8    S2
00009F P   35 8 00008 V                STW,8    S2

                        36,            GOTO 2

0000A0 P   68 0 000AB P                B        2S

                        37,     300    S3=S3+(K-DP)*(K-DP)

0000A1 P   32 9 0000D V     3008       LW,9     K
0000A2 P   6A A 00000 X                BAL,6    9ITOR
0000A3 P   3C 8 0000E V                FSS,8    DP
0000A4 P   35 8 00019 V                STW,8    1TEMP
0000A5 P   32 9 0000D V                LW,9     K
0000A6 P   6A 6 00000 X                BAL,6    9ITOR
0000A7 P   3C 8 0000E V                FSS,8    DP
0000A8 P   3E 8 00019 V                FMS,8    1TEMP
0000A9 P   3D 8 00009 V                FAS,8    S3
0000AA P   35 8 00009 V                STW,8    S3

                        38,     2      CONTINUE

0000AB P   32 4 0000C V     23         LW,4     I
0000AC P   20 4 00001 A                AI,4     1
0000AD P   35 4 0000C V                STW,4    I
0000AE P   B1 4 00017 V                CW,4     *N
0000AF P   68 2 00077 P                BLE      5G

                        39,            SIG=SQRT((S1)/N      )

0000B0 P   B2 9 00017 V                LW,9     *N
0000B1 P   6A 6 00000 X                BAL,6    9ITOR
0000B2 P   35 8 00019 V                STW,8    1TEMP
0000B3 P   32 8 00007 V                LW,8     S1
0000B4 P   3E 8 00019 V                FDS,8    1TEMP
0000B5 P   6A 6 00000 X                BAL,6    9SQRT
0000B6 P   35 8 00012 V                STW,8    SIG

                        40,            SIGP=SQRT((S3)/NP)

0000B7 P   32 9 0000B V                LW,9     NP
0000B8 P   6A 6 00000 X                BAL,6    9ITOR
0000B9 P   35 8 00019 V                STW,8    1TEMP
0000BA P   32 8 00009 V                LW,8     S3
0000BB P   3E 8 00019 V                FDS,8    1TEMP
0000BC P   6A 6 00000 X                BAL,6    9SQRT
0000BD P   35 8 00013 V                STW,8    SIGP

                        41,            SIGM=SQRT((S2)/NM)
```

```
000BE P   32 9 0000A V                LW,9      NM
000BF P   6A 6 00000 X                BAL,0     9ITOR
000C0 P   35 8 00019 V                STW,8     1TEMP
000C1 P   32 8 00008 V                LW,8      S2
000C2 P   3E 8 00019 V                FDS,8     1TEMP
000C3 P   6A 6 00000 X                BAL,6     9SQRT
000C4 P   35 8 00014 V                STW,8     SIGM

                          42,         R=NEX/N

000C5 P   82 9 00018 V                LW,9      *NEX
000C6 P   86 9 00017 V                DW,9      *N
000C7 P   6A 6 00000 X                BAL,0     9ITOR
000C8 P   35 8 00015 V                STW,8     R

                          43,         OUTPUT 'NEX/N'.R

000C9 P   22 7 000CD P                LI,7      11G
000CA P   6A 6 00000 X                BAL,0     9PRINT
000CB P   6A 6 00000 X                BAL,6     9IODATA
000CC P   02 2 00015 V                SNGL,E    R
000CD P   68 0 000D4 P     11G        B         12G
000CE P     407D40D5 A                TEXT      '(' N'
000CF P     C5E761D5 A                TEXT      'EX/N'
000D0 P     7D617D40 A                TEXT      ''/'' '
000D1 P     D9407E40 A                TEXT      'R = '
000D2 P     7DF17D7C A                TEXT      ''',1PG'
000D3 P     6BF65D40 A                TEXT      ',6) '

                          44,         OUTPUT NP,NM

000D4 P   22 7 000D9 P     12G        LI,7      13G
000D5 P   6A 6 00000 X                BAL,0     9PRINT
000D6 P   6A 6 00000 X                BAL,6     9IODATA
000D7 P   01 4 0000B V                INTG,C    NP
000D8 P   01 2 0000A V                INTG,E    NM
000D9 P   68 0 000E0 P     13G        B         14G
000DA P     407D40D5 A                TEXT      '(' N'
000DB P     D7407E40 A                TEXT      'P = '
000DC P     7DC7617D A                TEXT      ''',G/'''
000DD P     40D5D440 A                TEXT      ' NM '
000DE P     7E407DC7 A                TEXT      '= ''G'
000DF P     5D404040 A                TEXT      ')   '

                          45,         OUTPUT D,GM,DP,DM,SIG,SIGM,SIGP

000E0 P   22 7 000EA P     14G        LI,7      15G
000E1 P   6A 6 00000 X                BAL,0     9PRINT
000E2 P   6A 6 00000 X                BAL,6     9IODATA
000E3 P   02 4 00010 V                SNGL,C    D
000E4 P   02 4 00011 V                SNGL,C    GM
```

```
000E5 P   02 4 0000E V          SNGL,C   DP
000E6 P   02 4 0000F V          SNGL,C   DM
000E7 P   02 4 00012 V          SNGL,C   SIG
000E8 P   02 4 00014 V          SNGL,C   SIGM
000E9 P   02 2 00013 V          SNGL,E   SIGP
000EA P   68 0 00102 P    156   B        166
000EB P     4D7D40C4 A          TEXT     '(''  D'
000EC P     407E407D A          TEXT     '  =  '''
000ED P     F1D7C74B A          TEXT     '1PG,'
000EE P     F6617D40 A          TEXT     '6/''  '
000EF P     C7D4407E A          TEXT     'GM ='
000F0 P     407DC74B A          TEXT     '  ''G,'
000F1 P     F6617D40 A          TEXT     '6/''  '
000F2 P     C4D7407E A          TEXT     'DP ='
000F3 P     407DC74B A          TEXT     '  ''G,'
000F4 P     F6617D40 A          TEXT     '6/''  '
000F5 P     C4D4407E A          TEXT     'DM ='
000F6 P     407DC74B A          TEXT     '  ''G,'
000F7 P     F6617D40 A          TEXT     '6/''  '
000F8 P     E2C9C740 A          TEXT     'SIG  '
000F9 P     7E407DC7 A          TEXT     '=  ''G'
000FA P     4BF6617D A          TEXT     ',6/'''
000FB P     40E2C9C7 A          TEXT     '  SIG'
000FC P     D4407E40 A          TEXT     'M =  '
000FD P     7DC74BF6 A          TEXT     ''''G,6'
000FE P     617D40E2 A          TEXT     '/''  S'
000FF P     C9C7D740 A          TEXT     'IGP  '
00100 P     7E407DC7 A          TEXT     '=  ''G'
00101 P     4BF65D40 A          TEXT     ',6)  '

                      46,       WRITE(108,5)D,GM,DP,DM,SIG,SIGP,SIGM,(T(I),I=1,N)

00102 P   22 7 0001C V    166   LI,7     5S
00103 P   6A 6 00000 X          BAL,0    #PRINT
00104 P   6A 6 00000 X          BAL,6    #IODATA
00105 P   42 4 00010 V          #SNG,C   D
00106 P   42 4 00011 V          #SNG,C   GM
00107 P   42 4 0000F V          #SNG,C   DP
00108 P   42 4 0000F V          #SNG,C   DM
00109 P   42 4 00012 V          #SNG,C   SIG
0010A P   42 4 00013 V          #SNG,C   SIGP
0010B P   42 0 00014 V          #SNG     SIGM
0010C P   22 9 00001 A          LI,9     1
0010D P   35 9 0000C V          STW,9    I
0010E P   68 0 00113 P          B        186
0010F P   32 1 0000C V    176   LW,1     I
00110 P   6A 6 00000 X          BAL,0    #IODATA
00111 P   C1 0 20016 V          #INT     *T,1
00112 P   33 1 0000C V          MTW,1    I
00113 P   B2 9 00017 V    186   LW,9     *N
00114 P   31 9 0000C V          CW,9     I
```

```
00115 P   68 1 0010F P                BGE       17G
00116 P   6A 6 00000 X                BAL,6     #ENDIOL

                          47,     5   FORMAT(2(F8,3,10X,F8,3/),3(F8,3,10X)/,300(20I6/))
                          48,         IF(PASS.EQ.2) GOTO 2000

00117 P   22 9 00002 A                LI,9      2
00118 P   31 9 00001 V                CW,9      PASS
00119 P   69 3 00116 P                BNE       19G
0011A P   68 0 0011E P                B         2000S

                          49,         PASS=2

0011B P   22 9 00002 A    19G         LI,9      2
0011C P   35 9 00001 V                STW,9     PASS

                          50,         GOTO 1000

0011D P   68 0 00013 P                B         1000S

                          51,  2000 CONTINUE
                          52,         RETURN

0011E P   E8 0 00018 V    2000S       B         *1PTMP

                          53,         END

      *** L O C A L   S T O R A G E ***

                                      CSECT     0
00000 V                               RES       22          LOCAL VARIABLES
00016 V                   T           RES       1           DUMMY
00017 V                   N           RES       1           DUMMY
00018 V                   NEX         RES       1           DUMMY
00019 V                   1TEMP       RES       1           TEMP
0001A V                   2TEMP       RES       1           TEMP
0001B V                   1PTMP       RES       1           PRIVATE TEMP

      *** E X T E R N A L   R E F E R E N C E S ***

                                      REF       9SETUPN,9PRINT,9IODATA,9ITOR
                                      REF       9RTOI,9SQRT,#PRINT,#IODATA
                                      REF       #ENDIOL

      *** E X T E R N A L   D E F I N I T I O N S ***

00000 V                               DEF       V#DISTANCE
00000 P                               DEF       DISTANCE

                                      END
```

```
                                        CSECY    1                                                    2
                                        SYSTEM   SIGSF
                                        SYSTEM   FORTLIB
00000 P  6A 6 00000 X    4MAIN         BAL,6    9INITIAL

                          1.            DIMENSION A(5,5),B(5,5), C(5,5)
                          2.            CALL EXU

00001 P  22 E 00000 A                   LI,14    0
00002 P  6A F 00000 X                   BAL,15   EXU

                          3.            DO 50 I=1,5

00003 P  22 4 00001 A                   LI,4     1
00004 P  35 4 0004B V                   STW,4    I
00005 P  20 4 FFFFA A                   AI,4     -6

                          4.            Y=SIN(I)

00006 P  32 9 0004B V    16             LW,9     I
00007 P  6A 6 00000 X                   BAL,6    9ITOR
00008 P  6A 6 00000 X                   BAL,6    9SIN
00009 P  35 8 0004C V                   STW,8    Y

                          5.            DO 50 J=1,5

0000A P  22 2 00001 A                   LI,2     1
0000B P  35 2 0004D V                   STW,2    J
0000C P  20 2 FFFFA A                   AI,2     -6

                          6.            C(I,J)=0.

0000D P  22 8 00000 A    26             LI,8     0,0
0000E P  22 1 00005 A                   LI,1     5
0000F P  37 1 0004D V                   MW,1     J
00010 P  30 1 0004B V                   AW,1     I
00011 P  35 8 2002C V                   STW,8    C-6,1

                          7.            A(I,J)= I+2.*J

00012 P  32 9 0004D V                   LW,9     J
00013 P  6A 6 00000 X                   BAL,6    9ITOR
00014 P  3F 8 00031 P                   FMS,8    =2.0
00015 P  35 8 0004E V                   STW,8    1TEMP
00016 P  32 9 0004B V                   LW,9     I
00017 P  6A 6 00000 X                   BAL,6    9ITOR
00018 P  3D 8 0004E V                   FAS,8    1TEMP
00019 P  35 8 3FFFA V                   STW,8    A-6,1

                          8.    50      B(I,J)= 2.*I+3.*J
```

```
00019 P   32 9 00040 V    503       LW,9     J
00018 P   6A 6 00000 X              BAL,6    9ITOR
0001C P   3F 8 00032 P              FMS,8    =3.0
0001D P   35 8 0004E V              STW,8    1TEMP
0001E P   32 9 00048 V              LW,9     I
0001F P   6A 6 00000 X              BAL,6    9ITOR
00020 P   3F 8 00031 P              FMS,8    =2.0
00021 P   3D 8 0004E V              FAS,8    1TEMP
00022 P   35 8 20013 V              STW,8    B-6,1
00023 P   33 1 00040 V              MTW,1    J
00024 P   65 2 00000 P              BIR,2    2G
00025 P   33 1 00048 V              MTW,1    I
00026 P   65 4 00004 P              BIR,4    1G

                      9,            CALL     MRMUL(A,B,C,5,5,5)

00027 P   22 E 00006 A              LI,14    6
00028 P   6A F 00000 X              BAL,15   MRMUL
00029 P   02 0 00000 V              SNGL     A
0002A P   02 0 00019 V              SNGL     B
0002B P   02 0 00032 V              SNGL     C
0002C P   01 0 00033 P              INTG     =5
0002D P   01 0 00033 P              INTG     =5
0002E P   01 0 00033 P              INTG     =5

                      10,           STOP

0002F P   22 9 00000 A              LI,9     0
00030 P   6A 6 00000 X              BAL,6    9STOP

                      11,           END
```

*** C O N S T A N T S ***

```
00031 P     41200000 A
00032 P     41300000 A
00033 P     00000005 A
```

*** L O C A L S T O R A G E ***

```
                                    CSECT    0
00000 V                             RES      78                  LOCAL VARIABLES
0004E V                   1TEMP     RES      1                   TEMP
```

*** E X T E R N A L R E F E R E N C E S ***

```
                                    REF      EXU,MRMUL
                                    REF      9INITIAL,9ITOR,9SIN,9STOP
```

*** E X T E R N A L D E F I N I T I O N S ***

programma de tri sans call


```
                                        CSECT   1
                                        SYSTEM  S1650
                                        SYSTEM  FORTLIB
00000 P   6A 6 00000 A      &MAIN       BAL,6   9INITIAL

                          1,            INTEGER A(100)
                          2,            READ (105,1) N

00001 P   22 7 00068 V                  LI,7    1S
00002 P   6A 6 00000 X                  BAL,6   #READ
00003 P   6A 6 00000 X                  BAL,6   #IODATA
00004 P   41 2 00064 V                  #INT,E  N

                          3,            READ (105,2)(A(I),I=1,N)

00005 P   22 7 0000A V                  LI,7    2S
00006 P   6A 6 00000 X                  BAL,6   #READ
00007 P   22 9 00001 A                  LI,9    1
00008 P   35 9 00065 V                  STW,9   I
00009 P   68 0 0000E P                  B       2G
0000A P   32 1 00065 V      1G          LW,1    I
0000B P   6A 6 00000 X                  BAL,6   #IODATA
0000C P   41 0 3FFFF V                  #INT    A-1,1
0000D P   33 1 00065 V                  MTW,1   I
0000E P   32 9 00064 V      2G          LW,9    N
0000F P   31 9 00065 V                  CW,9    I
00010 P   68 1 0000A P                  BGE     1G
00011 P   6A 6 00000 X                  BAL,6   #ENDIOL

                          4,      1     FORMAT (I3)
                          5,      2     FORMAT (20(I3,X))
                          6,         22 CALL EXU

00012 P   22 E 00000 A      22S         LI,14   0
00013 P   6A F 00000 X                  BAL,15  EXU

                          7,            DO 100 I=1,N

00014 P   22 4 00001 A                  LI,4    1
00015 P   35 4 00065 V                  STW,4   I

                          8,            DO 100 J=I,N

00016 P   32 2 00065 V      3G          LW,2    I
00017 P   35 2 00066 V                  STW,2   J

                          9,            IF (A(I)-A(J)) 31,100,100

00018 P   32 9 9FFFF V      4G          LW,9    A-1,4
00019 P   38 9 5FFFF V                  SW,9    A-1,2
0001A P   68 1 00023 P                  BGEZ    100S
```

```
                        10,       31  IX=A(I)

00018 P   32 4 00065 V     315        LW,4      I
00019 P   32 9 9FFFF V                LW,9      A-1,4
0001A P   35 9 00067 V                STW,9     IX

                        11,           A(I)=A(J)

0001B P   32 2 00066 V                LW,2      J
0001C P   32 9 5FFFF V                LW,9      A-1,2
00020 P   35 9 9FFFF V                STW,9     A-1,4

                        12,           A(J)=IX

00021 P   32 9 00067 V                LW,9      IX
00022 P   35 9 5FFFF V                STW,9     A-1,2

                        13,      100  CONTINUE

00023 P   32 4 00065 V     1008       LW,4      I
00024 P   32 2 00066 V                LW,2      J
00025 P   20 2 00001 A                AI,2      1
00026 P   35 2 00066 V                STW,2     J
00027 P   31 2 00064 V                CW,2      N
00028 P   68 2 00018 P                BLE       4G
00029 P   20 4 00001 A                AI,4      1
0002A P   35 4 00065 V                STW,4     I
0002B P   31 4 00064 V                CW,4      N
0002C P   68 2 00016 P                BLE       3G

                        14,           WRITE (108,4) (A(I),I=1,N)

0002D P   22 7 00070 V                LI,7      4S
0002E P   6A 6 00000 X                BAL,6     #PRINT
0002F P   22 9 00001 A                LI,9      1
00030 P   35 9 00065 V                STW,9     I
00031 P   68 0 00036 P                B         6G
00032 P   32 1 00065 V     5B         LW,1      I
00033 P   6A 6 00000 X                BAL,6     #IODATA
00034 P   41 0 3FFFF V                #INT      A-1,1
00035 P   33 1 00065 V                MTW,1     I
00036 P   32 9 00064 V     6B         LW,9      N
00037 P   31 9 00065 V                CW,9      I
00038 P   68 1 00032 P                BGE       5G
00039 P   6A 6 00000 X                BAL,6     #ENDIOL

                        15,       4   FORMAT(20(I3,X))
                        16,           STOP

0003A P   22 9 00000 A                LI,9      0
```

programme de tri avec call


```
                                        CSECT   1
                                        SYSTEM  SIGSF
                                        SYSTEM  FORTLIB
00000 P   6A 6 00000 X      4MAIN       BAL,6   9INITIAL

                            1.          INTEGER A(100)
                            2.          READ (105,1) N

00001 P   22 7 00067 V                  LI,7    1S
00002 P   6A 6 00000 X                  BAL,6   #READ
00003 P   6A 6 00000 X                  BAL,6   #IODATA
00004 P   41 2 00064 V                  #INT,E  N

                            3.          READ (105,2)(A(I),I=1,N)

00005 P   22 7 00069 V                  LI,7    2S
00006 P   6A 6 00000 X                  BAL,6   #READ
00007 P   22 9 00001 A                  LI,9    1
00008 P   35 9 00065 V                  STW,9   I
00009 P   68 0 0000E P                  B       2G
0000A P   32 1 00065 V      1G          LW,1    I
0000B P   6A 6 00000 X                  BAL,6   #IODATA
0000C P   41 0 3FFFF V                  #INT    A-1,1
0000D P   33 1 00065 V                  MTW,1   I
0000E P   32 9 00064 V      2G          LW,9    N
0000F P   31 9 00065 V                  CW,9    I
00010 P   68 1 0000A P                  BGE     1G
00011 P   6A 6 00000 X                  BAL,6   #ENDIOL

                            4.      1   FORMAT (I3)
                            5.      2   FORMAT (20(I3,X))
                            6.          22 CALL EXU

00012 P   22 E 00000 A      22S         LI,14   0
00013 P   6A F 00000 X                  BAL,15  EXU

                            7.          DO 100 I=1,N

00014 P   22 4 00001 A                  LI,4    1
00015 P   35 4 00065 V                  STW,4   I

                            8.          DO 100 J=I,N

00016 P   32 2 00065 V      3G          LW,2    I
00017 P   35 2 00066 V                  STW,2   J

                            9.          IF (A(J),GT,A(I))       CALL CHG(A,I,J)

00018 P   32 9 FFFFF V      4G          LW,9    A-1,4
00019 P   31 9 5FFFF V                  CW,9    A-1,2
0001A P   68 1 00020 P                  BGE     5G
```

```
00018 P   22 E 00003 A                LI,14    3
0001C P   6A F 00000 X                BAL,15   CMG
0001D P   01 0 00000 V                INTG     A
0001E P   01 0 00065 V                INTG     I
0001F P   01 0 00066 V                INTG     J

                      10.     100     CONTINUE

00020 P                       5G      EQU      $
00020 P   32 4 00065 V        100$    LW,4     I
00021 P   32 2 00066 V                LW,2     J
00022 P   20 2 00001 A                AI,2     1
00023 P   35 2 00066 V                STW,2    J
00024 P   31 2 00064 V                CW,2     N
00025 P   68 2 00018 P                BLE      4G
00026 P   20 4 00001 A                AI,4     1
00027 P   35 4 00065 V                STW,4    I
00028 P   31 4 00064 V                CW,4     N
00029 P   68 2 00016 P                BLE      3G

                      11.             WRITE(108,4) (A(I),I=1,N)

0002A P   22 7 0006F V                LI,7     4S
00028 P   6A 6 00000 X                BAL,6    #PRINT
0002C P   22 9 00001 A                LI,9     1
0002D P   35 9 00065 V                STW,9    I
0002E P   68 0 00033 P                B        7G
0002F P   32 1 00065 V        6G      LW,1     I
00030 P   6A 6 00000 X                BAL,6    #IODATA
00031 P   41 C 3FFFF V                #INT     A-1,1
00032 P   33 1 00065 V                MTW,1    I
00033 P   32 9 00064 V        7G      LW,9     N
00034 P   31 9 00065 V                CW,9     I
00035 P   68 1 0002F P                BGE      6G
00036 P   6A 6 00000 X                BAL,6    #ENDIOL

                      12.     4       FORMAT(20(I3,X))
                      13.             STOP

00037 P   22 9 00000 A                LI,9     0
00038 P   6A 6 00000 X                BAL,6    9STOP

                      14.             END

        *** L O C A L   S T O R A G E ***

                                      CSECT    0
00000 V                               RES      105              LOCAL VARIABLES

        *** E X T E R N A L   R E F E R E N C E S ***
```