الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : D'ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

ETUDE ET REALISATION D UN MICRO-SYSTEME AVEC QUATRE UNITES ARITHMETIQUES RAPIDES EN PARALLELE

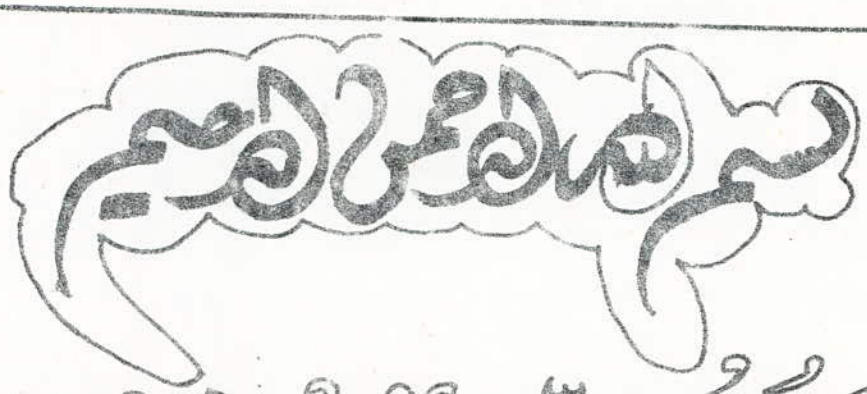
Proposé par : Mr Guettache

Etudié par : Mlle Bordjiba Dzair x
Mlle Menzer Chafia

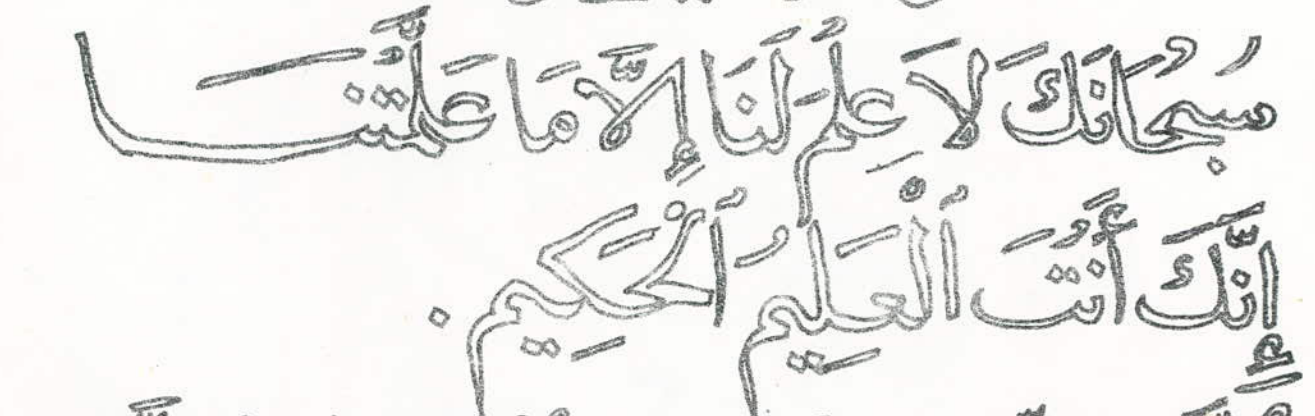
Dirigé par : Mr Guettache

PROMOTION : JUIN 84

E.N.P. 10, Avenue Hacen Badi - EL-HARRACH - ALGER


بسم الله الرحمن الرحيم


وقل رب زدني علما

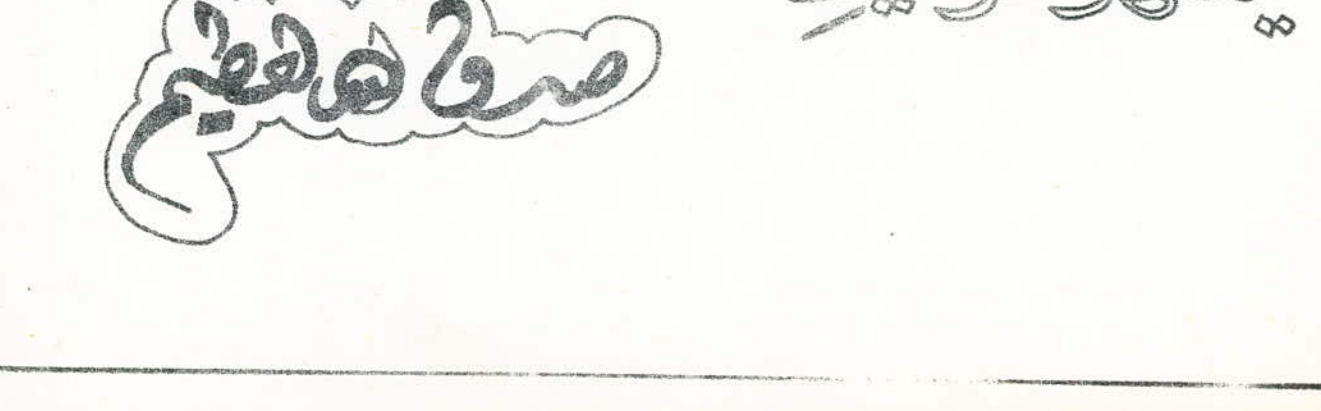

سبحانك لا علم لنا إلا ما علمتنا

إنك أنت العليم الحكيم.


اللهم علمنا ما ينفعنا وانفعنا بما علمتنا


رب اشرح لي صدري ويسر

لي أمري واحلل عقدة من لساني


يفقهوا قولي

صدق الله العظيم

DEDICACES

A mes chers parents qui m'ont montré le chemin de l'ecole

A mes frères, leurs femmes et leurs enfants

A mes sœurs, leurs maris et leurs enfants

A tous mes amis (ies)

A tous ceux que j'aime

Je dedie ce modeste travail

Djahida

A la mémoire de feu ma grand-mère

A mes chers parents pour tout ce qu'ils m'ont donné

A mes frères

A mes sœurs, leurs maris et leurs enfants

A ma sœur Yasmina

A mon amie Yasmina

A tous mes amis (ies)

A tous ceux qui me sont chers

Chafia

REMERCIEMENTS

Au terme de notre travail, nous tenons à exprimer notre profonde reconnaissance à Monsieur GUETTACHE de nous avoir proposé le sujet et suivi durant toute l'élaboration de ce travail.

Nous remercions également tout le personnel du Laboratoire "Architecture des systèmes" pour l'aide et le matériel qu'ils nous ont fournis.

Nous sommes très sensibles à l'aide précieuse que Monsieur YAHIAOUI nous a apporté, ce pourquoi nous le remercions vivement.

Que tout le Personnel de la SNTF de Constantine ainsi que Monsieur HOCINI Responsable de BATIMETAL et ses Secretaires Mesdemoiselles TLEMSANI et KHIMOUN, trouvent ici l'expression de nos sincères remerciements pour nous avoir aidé à la mise au point de ce polycopié.

Nous n'oublions pas de témoigner notre profonde gratitude aux enseignants qui ont contribué de près ou de loin à notre formation.

-O- PLAN -O-

INTRODUCTION

CHAPITRE I : MULTIPROCESSING

A - Structure logique
B - Structure physique
C - Classification des architectures en parallèle.

PRESENTATION DU SYNOPTIQUE

CHAPITRE II :: I - INTRODUCTION AU UP MC 6800

A - Caractéristiques
B - Organisation interne
C - Lignes d'interface
D - Jeu d'instruction
E - Mode d'adressage
F - Structure de la pile
G - Calcul numérique par le MC 6800

II - PRESENTATION DE L'UNITE ARITHMETIQUE RAPIDE AM9512

A - Caractériqtiques
B - Description fonctionnelle
C - Représentation des nombres en virgule flottante
D - Algorithme des opérations en virgule flottante

CHAPITRE III COUPLAGE DE L'UNITE DE TRAITEMENT AU MICROPROCESSEUR MC 6800

A - Décodage d'adresse
B - Activation et désactivation des buffers de données et d'adresse
C - Décodage d'adresse de l'U.A.R.
D - Logique de commande et de contrôle
E - Indicateur d'état de transfert
F - Programme d'introduction des données dans l'unite de traitement.

.../...

-O- INTRODUCTION -O-

INTRODUCTION

Le niveau de performances des machines telles que les micro-ordinateurs domestiques; les superordinateurs doit beaucoup aux progrés rapides de la micro-electronique.

L'apparition de nouveaux concepts dans l'architecture de ces ordinateurs a joué un rôle tout aussi important; le mot architecture désigne ici l'organisation logique de la machine.

Les plus importantes de ces innovations sont celles qui permettent à la machine de réaliser en // un grand nombre d'opérations similaires alors que les 1ers ordinateurs obligeaient le programmeur à décomposer son problème en une série de pas élémentaires destinés à être éxécutés un par un. Les derniers nés des superordinateurs permettent à l'utilisateur de spécifier que de nombreux pas élémentaires doivent être éxécutés simultanément.

L'ensemble des dispositions mises en oeuvre pour assurer ce traitement en // peut se répartir en 2 catégories : le traitement segmenté (pipelining) et le traitement en // (multiprocessing) ce dernier point fera l'objet de notre présente étude qui sera suivie de la réalisation d'un micro-système basé sur le UP MC 6800 où plusieurs unités arithmétiques rapides travaillant en parallèle.

De même que nous essaierons de déterminer l'effet de saturation d'un tel micro-système sur un exemple précis qui est le problème d'inversion de matrice.

CHAPITRE I

-O- LE MULTIPROCESSING -O-

LE MULTIPROCESSING

C'est une technique de multitraitement où dans une même machine plusieurs unités centrales (dites processeurs) travaillent simultanément sur des programmes ou des morceaux de programmes différents.
Ces processeurs se partagent des ressources communes : mémoires, unités périphériques et chemins de communications.

A) - STRUCTURE LOGIQUE

Ici la structure logique se rapporte à la façon dont la responsabilité du contrôle est distribuée parmi les éléments du système à processeurs multiples qui doivent être bien définis.
Il existe deux organisations qui sont :

a) - Organisation verticale

Les éléments sont structurés par hiérarchie impliquant une relation maître-esclave et à tout moment un seul élément peut se comporter comme un maitre. Les inter-communications se faisant par l'intermédiaire du maitre et sont initiées par lui.

b) - Organisation horizontale

Les éléments sont égaux logiquement, impliquant une relation maitre-maitre. Tout élément peut communiquer avec un autre élément du système et la coordination entre les éléments peut-être faite avec un ou plusieurs contrôleurs.

B) - STRUCTURE PHYSIQUE

Cette structure fait appel à la façon dont l'information est échangée. Elle est en fonction de l'arrangement de la communication entre les processeurs et la topologie de l'interconnexion.

a) - Arrangement de la communication entre processeurs.

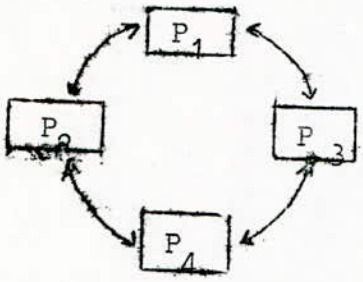
Les données sont transférées entre processeurs et sont transmises à l'extérieur à travers une structure de mémoire commune ou une structure de bus.
Dans la structure mémoire commune, les éléments n'accèdent pas les uns aux autres.
Dans la structure de bus, une chaine logique établit un moyen de communication entre les éléments.

b) - Topologie d'interconnexion
Le mode de transfert des données et la topologie d'interconnexion sont les critères de base de l'inter communication entre les éléments du système.

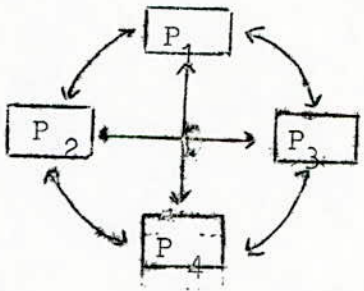
.../...

La notion de topologie introduit forcément la géométrie d'interconnexion, nous présentons quelques unes des géométries éxistantes.

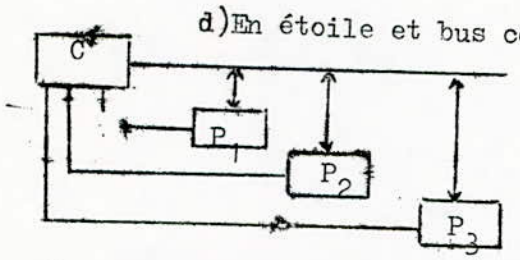
a)En anneau



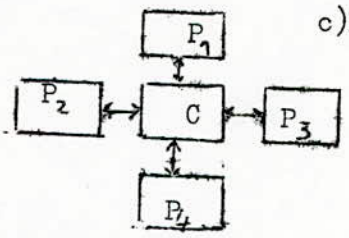
b) 2 à 2



d)En étoile et bus commun



c)En etoile



C) - CLASSIFICATION DES ARCHITECTURES EN PARALLELE

La classification la plus répandue est celle de FLYNN qui définit l'activité d'un ordinateur comme l'intersection de deux flots : les instructions et les données, chacun de ces flots peut-être simple ou multiple; ce qui définit quatre types de génériques des systèmes informatiques.

- Les systèmes SISD : simple flot d'instruction; simple flot de données. Il ya éxécution séquentielle des instructions.
- Les systèmes SIMD : simple flot d'instructions, multiple flot de données. Le même flot d'instruction est éxecuté sur plusieurs ensembles de données. En général ces systèmes ont une seule unité centrale de contrôle qui cherche et décode les instructions, ensuite elle les distribue parmi les éléments.
- Les système MISD : multiple flot d'instructions, simple flot de données. Le même ensemble de données sert d'opérandes à différents flots d'instructions s'éxécutant simultanément. Dans ces machines, il ya segmentation des calculs en des étapes consécutives.
- Les systèmes MIMD : multiple flot d'instruction, multiple flot de données. Ils sont caractérisés par le fait que différents flots d'instructions s'éxécutent simultanément, chacun prenant ses opérandes dans un ensemble de données préalablement choisi.

Dans ces machines, chaque processeur a sa propre unité de contrôle et sa mémoire locale et le programme informatique est découpé en tâches différentes et chacune de celles-ci est affectée à un seul processeur. Comme l'étude du microsystème de notre projet est basée sur la structure SIMD, nous développerons cette sturcture dans ce qui suit.

.../...

STRUCTURE SIMD

Dans les simulations de champ continu et beaucoup d'autres gros problèmes scientifiques, il est possible d'effectuer une séquence particulière d'opérations qui décrit les changements de variables du champ en un point donné entre les instants t et t + 1 et cette même séquence d'opérations peut-être appliquée simultanément à tous les points du réseau puisque les opérations correspondants à ces ≠ points sont indépendantes, elles peuvent s'effectuer en parallèle.
On étudie alors avec un soin tout particulier les algorithmes de résolutions demandant de nombreux calculs afin de maximaliser le nombres d'opérations qui peuvent être conduites en // sur cette nouvelle architecture.
Les considérations sur la vitesse des superordinateurs sont souvent fondées sur la notion de " Mégaflop ".
Un mégaflopreprésente un million d'opérations éxecutées en virgule flottante par seconde et il faut inclure dans la définition de la vitesse en mégaflops; le temps nécessaire pour aller chercher les informations en mémoire et y renvoyer les résultats; le taux de parallélisme propre au problème de la possibilité d'adaptation de l'ordinateur au calcul en // ainsi que l'habilité du programmeur.

ARCHITECTURE D'UNE TELLE MACHINE (SIMD)

Afin d'introduire des considérations sur l'architecture d'une telle machine, il est nécesseire de rappeler les constituants d'un ordinateur.
Il comporte 3 parties essentielles : Une mémoire, un processeur d'instructions et un processeur de données (il y a aussi le système d'entrées/sorties pour la communication avec le monde éxterieurs). (voir fig 1-a).
La mémoire contient à la fois les données et les instructions; les emplacements mémoires sont repérées par une adresse.
Le processeur d'instruction des 1ers ordinateurs était inactif durant la phase déxécution de l'instruction et on remarqua qu'on pouvait économiser plusieurs cycles en faisant rechercher l'instruction suivante au processeur d'instruction pendant que le processeur de donnée était entrain de manipuler l'instruction en cours; ceci pourvu que les 2 processeurs n'entrent pas en conflit pour accéder à la mémoire.
Un tel recouvrement de la recherche et de l'éxecution a été le concept précurseur de ce que l'on appelle aujourd'hui l'organisation en pipline ou segmentée et qui s'est imposée notament dans ce que l'on appelle les calculateurs vectoriels.

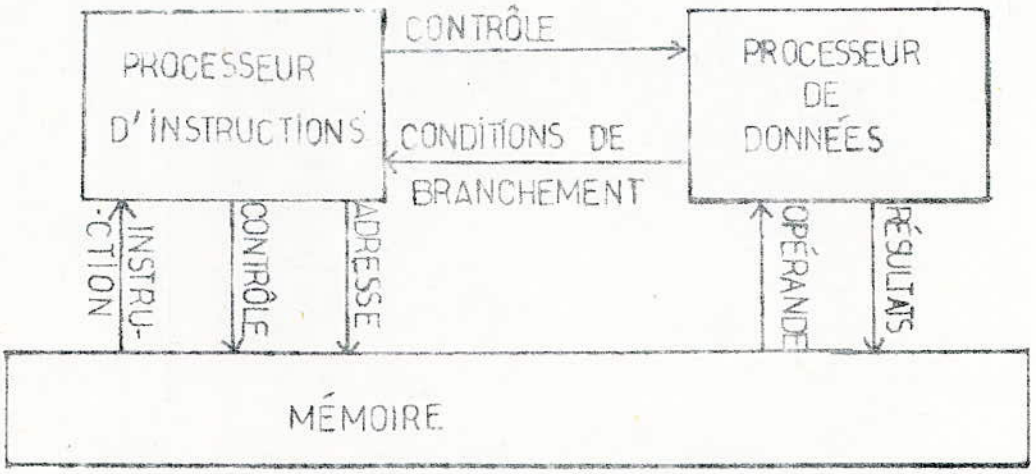
.../...

Le traitement vectoriel est une forme simple de calcul en // dont on dispose dans un superordinateur moderne.
Un vecteur est simplement une liste ordonée de données dont les éléments sont emmagasinés en mémoire de façon séquentielle.
Le nombre d'éléments de la liste est appelé longueur du vecteur.
A toute opération s'appliquant à un seul élément numérique, il existe une opération vectorielle correspondante qui consiste à appliquer la même opération à chaque élément du vecteur, la même chose pour une opération s'appliquant à une paire d'éléments de 2 vecteurs de longueur égale, pris dans un certain ordre
Les ordinateurs vectoriels dont les superordinateurs font partie, appliquent différentes stratégies pour accelerer l'execution des opérations vectorielles d'une consiste à introduire des instruction vectorielles dans le jeu d'instructions.
Une seule instruction vectorielle a pour effet de faire éxecuter la même opération sur toutes les paires d'éléments de 2 vecteurs, ce qui constitue l'opération vectorielle totale.
Aussi une instruction vectorielle précise t-elle en plus de l'opération à éxécuter, les 1eres adresses des 2 vecteurs opérandes, celle du vecteur résultant et leurs longueurs communes.

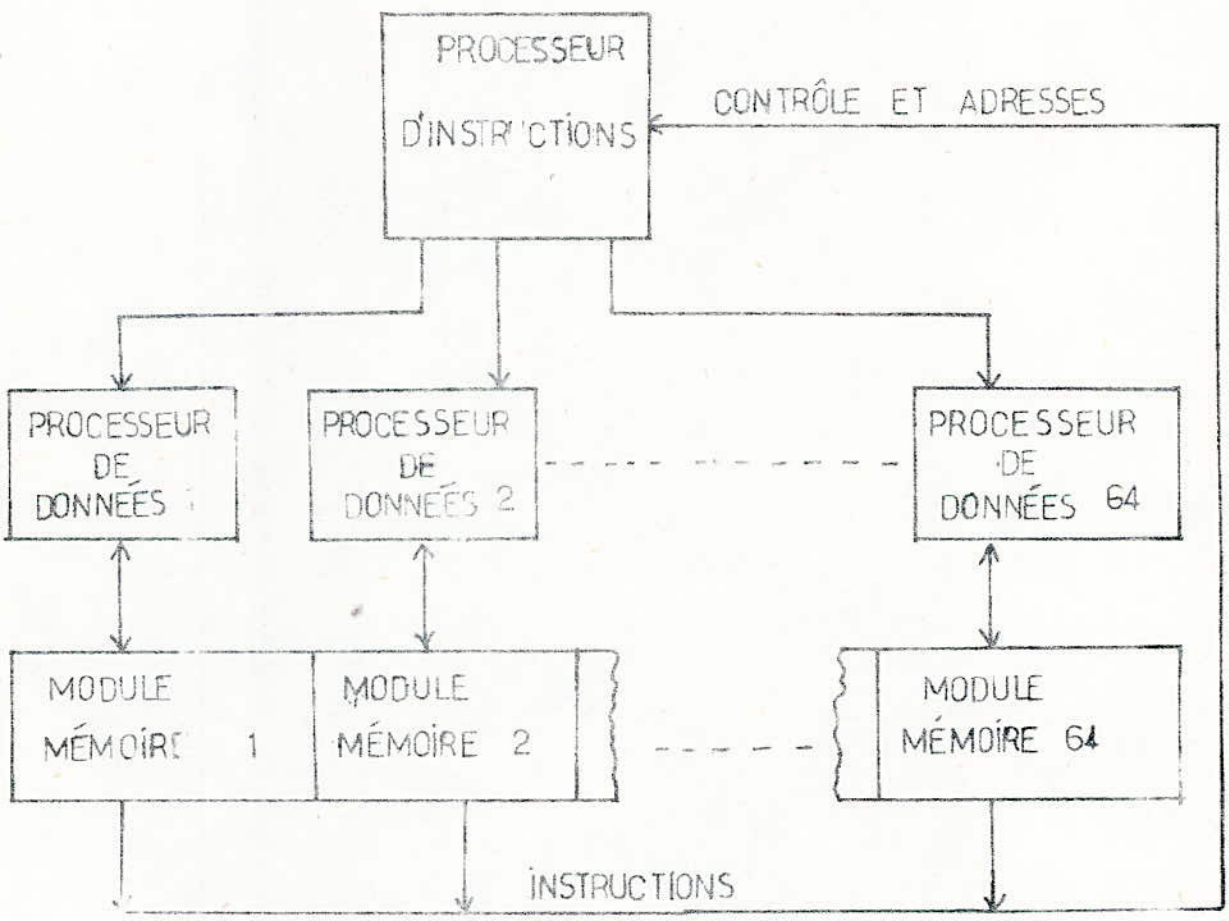
EXEMPLE DE MACHINE SIMD

* Le superordinateur ILLIAC IV.
Pour qu'une machine mérite (en 1982) le titre de superordinateur, elle doit être capable de maintenir une vitesse typique de 20 mégaflops pour une variété de problème ittératifs courants dont l'ensemble de données de base comporte un million de mots au moins.
Le 1er ordinateur capable d'atteindre 20 mégaflops ou mieux fut ILLIAC IV, unique en son genre et qui fut construit à la fin des années 1960 début des années 1970 par la compagnie Burroughs.
Il traita certains des plus vastes problèmes d'aérodynamique jamais programmés
Après sa mise hors service en septembre 1981, il ne reste au USA que 2 familles de superordinateurs opérationnels. L'un Cray 1 construit par Cray Research et l'autre Cyber 205 construit par Contrôl-data.
Ni Cray 1, ni Cyber 205 ne sont indiscutablement plus rapides qu'ILLIAC.
Ce dernier comprend dans sa structure une forme d'organisation permettant le traitement en parallèle.
La machine comportait 64 processeurs de données identiques travaillant en synchronisme sous la conduite d'un processeur d'instruction unique .
La mémoire d'ILLIAC IV était partagée en 64 modules, chaque module correspondant à un processeur de données (voir fig. 1-b).

.../...


PROCESSEUR
D'INSTRUCTIONS
CONTRÔLE
CONDITIONS DE
BRANCHEMENT
PROCESSEUR
DE
DONNÉES
INSTRU-
CTION
CONTRÔLE
ADRESSE
OPÉRANDE
RÉSULTATS
MÉMOIRE

(fig. 1-a)


PROCESSEUR
D'INSTRUCTIONS
CONTRÔLE ET ADRESSES
PROCESSEUR
DE
DONNÉES
PROCESSEUR
DE
DONNÉES 2
PROCESSEUR
DE
DONNÉES 64
MODULE
MÉMOIRE 1
MODULE
MÉMOIRE 2
MODULE
MÉMOIRE 64
INSTRUCTIONS

(fig. 1-b)

Les éléments d'un vecteur occupaient des emplacements correspondants dans de différents modules mémoires.

En conséquence la longueur optinale des vecteurs pour ILLIAC IV était de 64 bits, car alors les opérations vectorielles étaient effectuées environ 64 fois plus vite que la séquence des opérations scalaires ordinaires (par opposition aux opérations vectorielles, les opérations scalaires opèrent sur une paire de nombres à la fois ou sur des nombres uniques).

La machine pouvait donc réaliser simultanément jusqu'à 64 calculs correlés. Les opérations portant sur des vecteurs plus longs étaient divisées en opérations portant sur des segments de 64 termes et étaient éxécutées par une boucle de programme comme elles le sont dans une machine séquentielle.

Pour les vecteurs plus courts (ou pour le reste d'un vecteur long dont la longueur n'était pas divisible par 64) certains des processeurs de données pouvaient être mis hors circuit.

ILLIAC IV pouvait fonctionner comme s'il possédait 128 processeurs quand la longueur des mots pouvait être ramenée à 32 bits sans perte de la précision nécessaire.

CHAPITRE II

-O- PRESENTATION DU SYNOPTIQUE -O-

-O- INTRODUCTION AU MICROPROCESSEUR -O-

M C 6800

PRESENTATION DU SYNOPTIQUE

Afin de réaliser un simulateur numérique de réacteur nucléaire dont le modèle mathématique est régit par des équations differentielles, le laboratoire " Architecture des systèmes" a adopté la technique du multiprocessing . Ainsi le système conçu pour la simulation est réalisé à base de plusieurs UP "MC 6800 de MOTOROLA " qui travaillent en //, chacun d'eux résolvant une équation différentielle.

Un UP dit " maitre " assurera le dialogue entre les autres UP dits " Esclave ". La structure du système global correspond aux machines MIMD; suivant l'organisation "maitre Esclave".

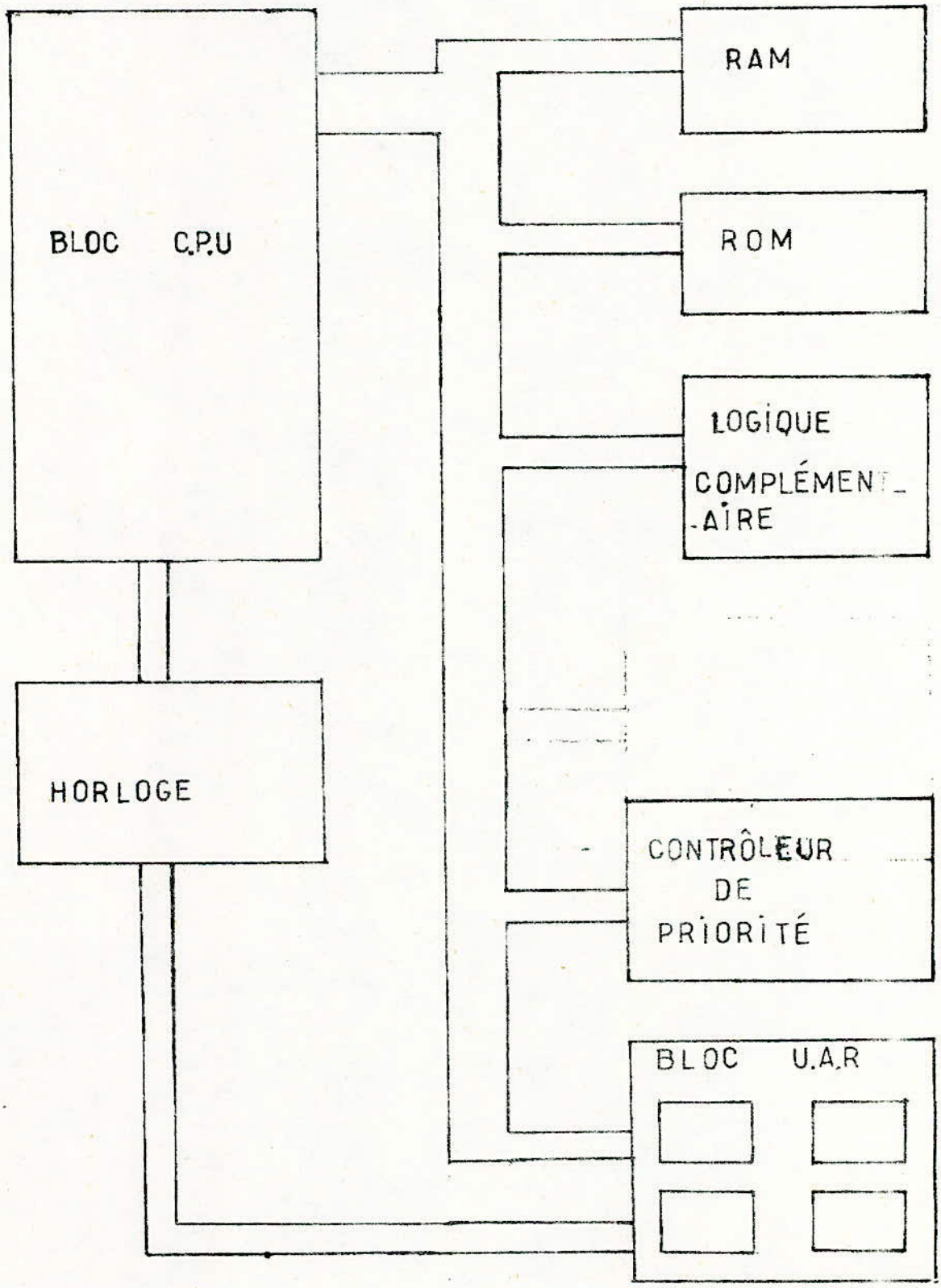
Le micro-système que nous nous sommes proposés d'étudier, représente un des " Esclaves" du système global de simulation.

La carte a été réalisée autour du UP MC 6800 qui constitue l'unité centrale du micro-système qui comprend :

- 4 U.A.R travaillant en parallèle et par interruption
- Un contrôleur de priorité le circuit MC 6828 qui gère les interruptions.
- Des circuits mémoires ROM et RAM
- Une logique complémentaire servant au couplage des ≠ blocs

Le UP MC 6800 communique avec ces différents blocs par l'intermédiaire de 3 bus d'adresse, de données et de contrôle de même il éxecute les opérations de traitement et gère l'utilisation des 4 U.A.R se comportant ainsi en "micro-maitre" à son niveau d'où la structure adoptée peut être assimilée à une structure SIMD.

SYNOPTIQUE DU MICRO _ SYSTÉME



A - CARACTERISTIQUES

Le MC 6800 est un microprocesseur monolithique 8 bits réalisant la fonction d'unité centrale pour la famille MC 6800.
Ce up est réalisé en technologie MOS à canal N (N M O S) est possède un espace d'adressage de 64 K octets de mémoire .
Il nécéssite une alimentation unique de + 5 V et sa consommation varie autour de 0,25 W.
Il travaille à une fréquence maximale de 2 MHZ.

B - ORGANISATION INTERNE DU MC 6800 : Voir (fig 2 - a)

Le microprocesseur comprend essentiellement :

a) - Une unité arithmétique et logique (ALU)

C'est l'ensemble des circuits combinatoires capables d'effectuer les opérations arithmétiques et logiques nécessaires au traitement des informations.

b) - Une unité de contrôle et de decodage:

Son rôle est de décoder et d'analyser les informations présentes dans le programme et de les faire traiter par les organes éxécutifs (ALU, ACCU ..) au rythme d'une impulsion d'horloge.

c) - Les registres internes du MPU

Le MPU a trois registres de 16 bits et trois registres de 8 bits accessibles par programme Voir (fig 2 - c)

* Le compteur programme (counter program)

C'est un registre de 16 bits qui contient l'adresse courante de l'instruction à éxécuter.

* Le registre d'index (index register)

C'est un registre de 16 bits qui est utilisé comme index dans le mode d'adressage indéxé.

* Le pointeur de pile (stack pointer)

C'est un registre de 16 bits qui permet de stocker l'adresse de l'emplacement mémoire du programme initial pour un retour ultérieur ceci lors d'un passage à un sous-programme.

* Les accumulateurs A et B

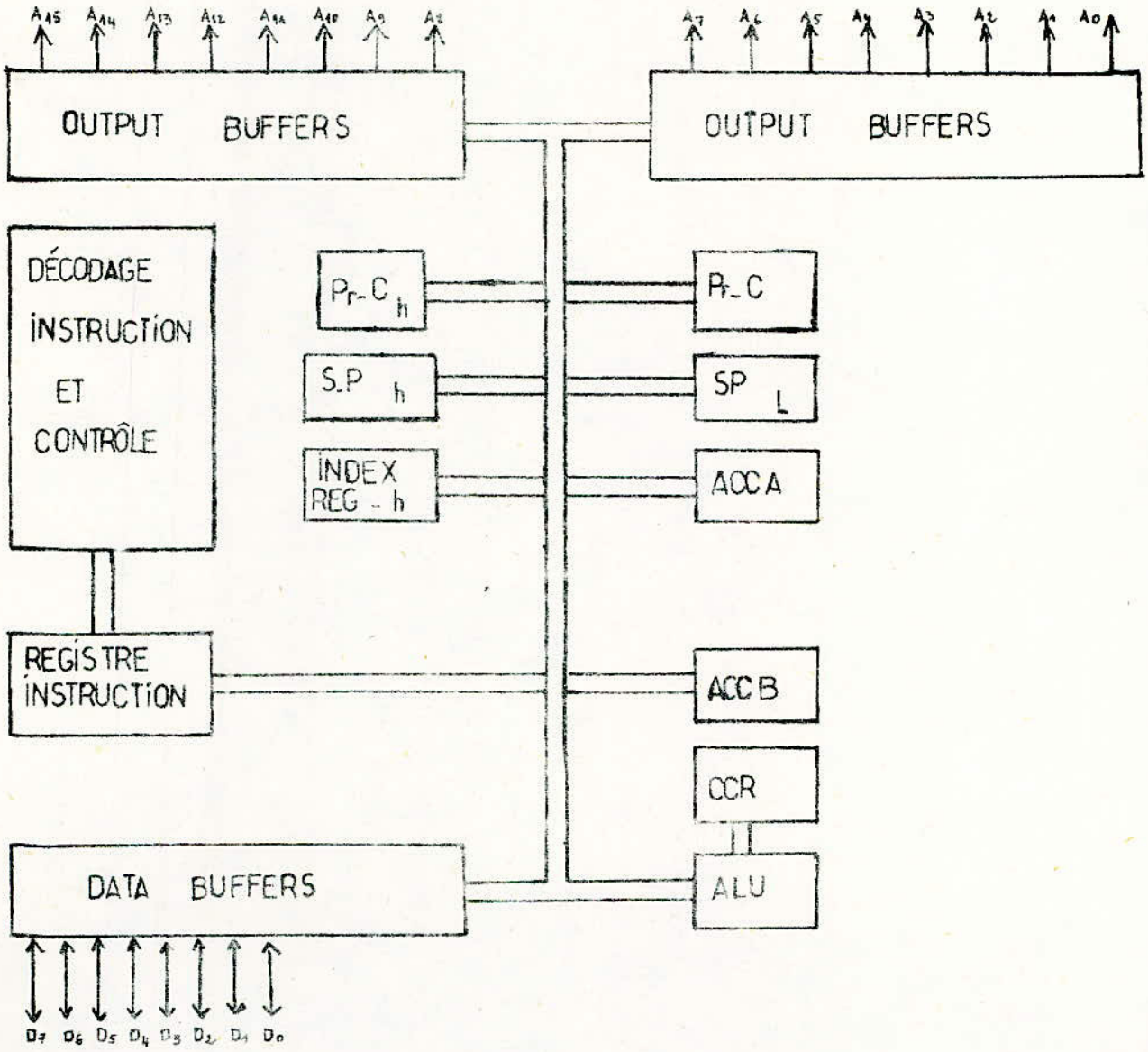
Ce sont des registres de 8 bits qui sont utilisés pour contenir des opérandes et des résultats de l'ALU.

* Le registre d'état (condition code register)

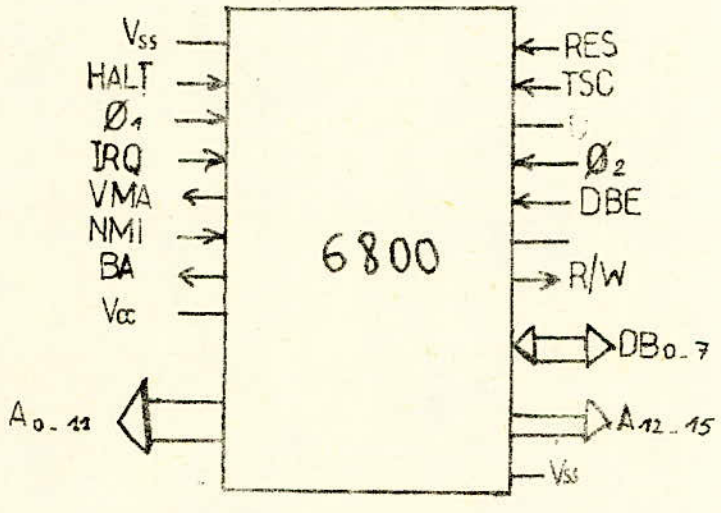
C'est un registre de 8 bits qui permet de disposer de 6 informations utiles à la gestion du programme .
5 informations concernent le résultat des opérations effectuées par l'ALU :

.../...

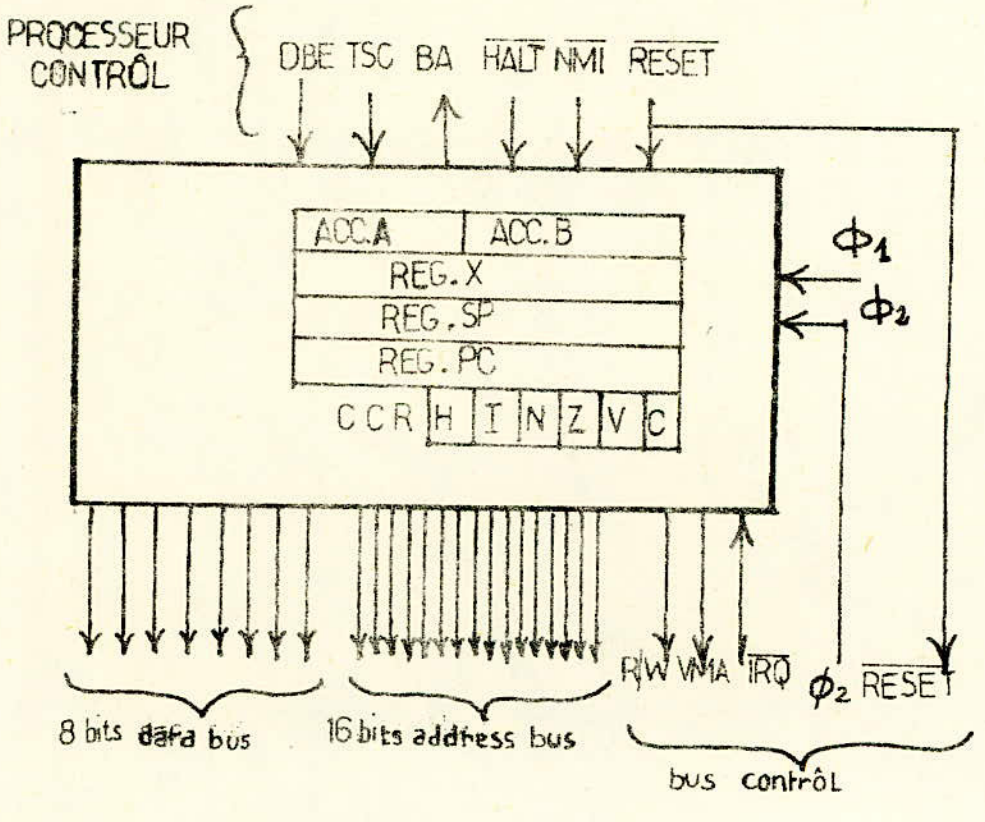
ORGANISATION INTERNE DU MC 6800



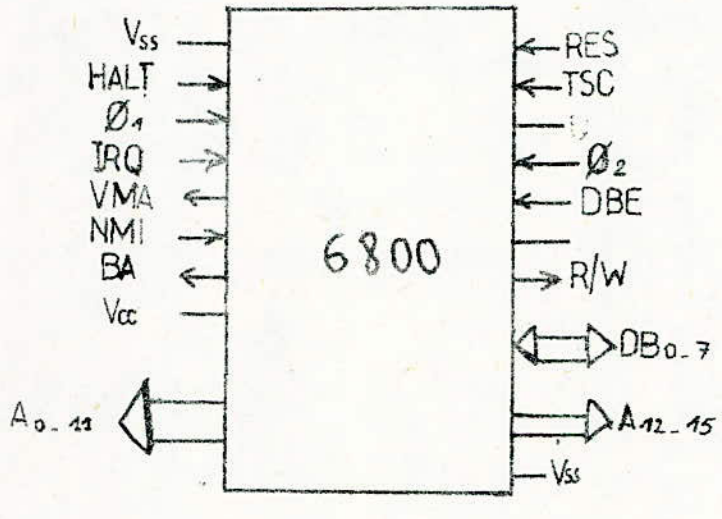
(fig-2-a)



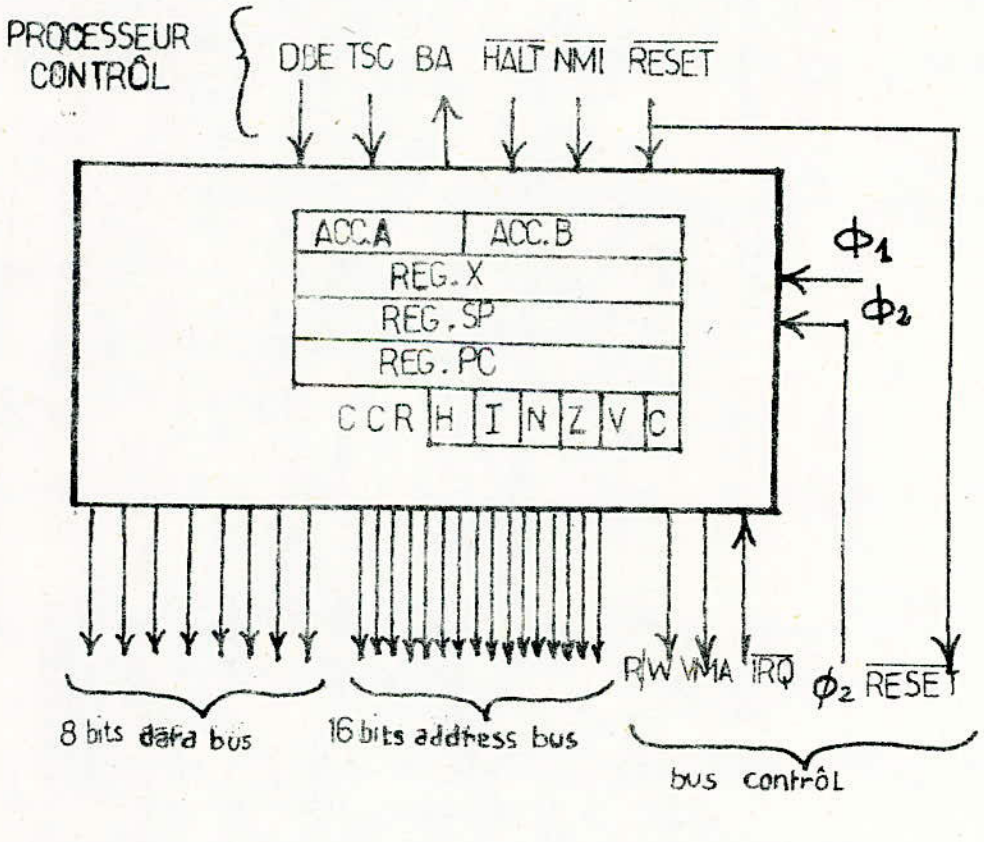
(fig.2-b) brochage du MC6800



(fig.2c) UNITÉ CENTRALE MPU MC6800



(fig.2-b) brochage du MC6800



(fig.2c) UNITÉ CENTRALE MPU MC6800

* N : résultat négatif
* Z : résultat nul
* V : dépassement de capacité (overflow)
* C : retenue (carry)
* H : demi-retenue (half carry)

La 6 ème information est le bit I : masque de l'interruption $\overline{IRQ}$. Les deux derniers bits sont constamment à "1".

C - LIGNES D'INTERFACE DU MICROPROCESSEUR

Les lignes d'entrée/sortie du microprocesseur sont divisées en trois groupes
- Lignes des bus
- Lignes de commande des bus
- Lignes de commande du UP.

a) - Lignes des bus :

* Bus d'adresse : A0 - A15 c'est un bus unidirectionnel de 16 bits, pouvant être mis en haute impédance dans des applications avec accés direct en mémoire (DMA) par activation de TSC et de $\overline{HALT}$.

* Bus de données : D0 - D7 c'est un bus bidirectionnel de 8 bits qui sert pour le transfert de données avec la mémoire et les circuits périphériques .

Il peut être mis en haute impédance par désactivation de la ligne DBE (état bas).

b) - Lignes de commande des bus :

* Entrée horloge : phase une et phase deux (ϕ_1 et ϕ_2) sur ces deux phases sont synchronisés l'adressage et le transfert de données .

* Ligne R/$\overline{W}$ (read/write) : Indique aux circuits mémoires et périphériques que le MPU est dans l'état lecture (R/$\overline{W}$ = 1) ou dans l'état écriture (R/$\overline{W}$ = 0).

* Ligne VMA : Sert à la validation d'adresse mémoire .
* Ligne DBE : Sert à l'activation du bus de données .
* Ligne BA : Sert à indiquer la disponibilité du bus d'adresse quand il est à l'état haute impédance.
* Ligne TSC : Sert à mettre le bus adresse et la ligne R/$\overline{W}$ en haute impédance pendant quelques cycles d'horloge .

c) - Lignes de commande du UP :

* L'entrée $\overline{RESET}$: met à l'état initial et fait démarrer le UP après une mise sous tension.

.../...

* L'entrée $\overline{HALT}$: permet l'arrêt du UP et toute activité d'exécution du programme est arrêtée; le bus est en haute impédance.

* L'entrée $\overline{IRQ}$: c'est une entrée de demande d'interuption masquable.

* Entrée $\overline{NMI}$: c'est une entrée d'interruption non masquable .

D - JEUX D'INSTRUCTION DU MPU

Le MPU possède 72 instructions différentes de longueur variable (un à trois octets); permettant d'éffectuer les opérations suivantes :

* Arithmétique binaire et décimale
* Logique
* Décalages
* Décalages circulaires
* Chargements
* Stockages
* Branchements conditionnels et inconditionnels
* Instructions associées aux interruptions
* Instructions de manipulation en pile

E - MODES D'ADRESSAGE DU MPU

Le MC 6800 possède sept modes d'adressages possibles.

* Adressage des accumulateurs (ACCX)

L'opérande est soit l'accumulateur A soit B . L'instruction est codée sur un seul octet.

* Adressage immédiat

L'opérande se trouve dans le 2 ème ou 3 ème octet de l'instruction selon qu'on s'adresse aux registres ou aux accumulateurs.

* Adressage direct

Dans le 2 ème octet se trouve l'adresse de l'opérande . On peut accéder aux 256 premiers octets de la mémoire.

* Adressage étendu

l'adresse de l'opérande est contenue dans les deuxième (bits de poids fort) et troisième (bits de poids faible) octets de l'instruction.
Ce qui permet de balayer toutes les mémoires de 0000 à FFFF.

* Adressage indéxé

Le contenu du deuxième octet de l'instruction (appelé aussi déplacement) est ajouté au contenu du registre d'Index pour former l'adresse absolue de l'opérande.

.../...

* Adressage implicite

L'opérande est indiqué par le code opération de l'instruction.

* Adressage relatif

Utilisé uniquement pour les instructions de branchement.
Le contenu du 2 ème octet de l'instruction est ajouté au contenu du compteur programme afin de déterminer l'adresse de branchement.
Les autres modes : le mode implicite, le mode relatif et le mode indirect sont rarement utilisés .

F - STRUCTURE DE LA PILE DU MC 6800

Le MC 6800 dispose d'une pile de registres volatiles, qui permet de mémoriser les informations et de les utiliser selon le mode LIFO (lastin, first out) : dernier entré, premier sorti . Le pointeur de pile permet d'adresser les registres de la pile.

* Gestion de la pile

Le registre pointeur de pile SP permet la gestion de la pile. Il contient d'adresse de la 1ère position libre au sommet de la pile.
Il est modifié à chaque entrée ou sortie d'information de sorte qu'il désigne toujours ce sommet de pile. Si l'on considère une pile descendante en mémoire, les adresses sont décroissantes vers le sommet de la pile, le registre est donc décrémenté à chaque entrée d'information et incrémenté à chaque sortie.
Des instructions de stockage en pile (PUSH) ou de sortie de la pile (PULL) permettent son utilisation; mais cette structure est aussi utilisée pour les appels de retour de sous-programmes .

* Sauvegarde en pile

La pile peut-être utilisée pour sauvegarger l'adresse de retour d'un sous-programme. Lorsqu'un programme est appelé, l'adresse de retour dans le programme appelant est stockée dans la pile et restituée dans le compteur programme à la fin de sous-programme;

.../...

G) - CALCUL NUMERIQUE EFFECTUE PAR LE UP MC 6800

Motorola met à notre disposition une BIBMAT qui contient plusieurs sous-programmes assurant chacun une fonction différente.
Entre autres la fonction de conversion d'un nombre binaire quelconque (positif ou négatif) en virgule flottante sur 24 bits.
Grâce à cette BIBMAT, nous avons pu dresser les tableaux ci-dessous donnant le temps d'éxécution des algorithmes d'addition, de division et de multiplication en VF sur 24 bits pour un système MC 6800.

Temps d'éxecution en US des opérations effectuées par le MC 6800

	Tmin	Tmax
ADD	810	1500
SUB	960	1320
MUL	1800	2640
DIV	2100	2280

Temps d'éxecution en cycles des opérations effectuées par l'U.A.R AM 9512

	Tmin	Typ	Tmax
ADD	58	220	512
SOUST	56	220	512
MULT	192	220	254
DIV	228	240	264

Après comparaison des 2 tableaux, nous constatons que pour l'unité arithmétique, le temps d'éxécution des opérations fondamentales sur 32 bits (simple précision) est relativement très court par rapport au temps d'éxécution des mêmes opérations effectuées par le UP et pour une précision de 24 bits seulement.
Par conséquent, il y a une nette amélioration dans la rapidité de traitement des opérations arithmétiques.

.../...

De même que la précision est largement augmentée, vu que l'U.A.R travaille sur 32 bits avec 8 bits réservés à l'exposant qui est codé par excés de $(2^7 - 1)$ ce qui permet une grande étendue du codage et la mantisse représentée sur 24 bits.

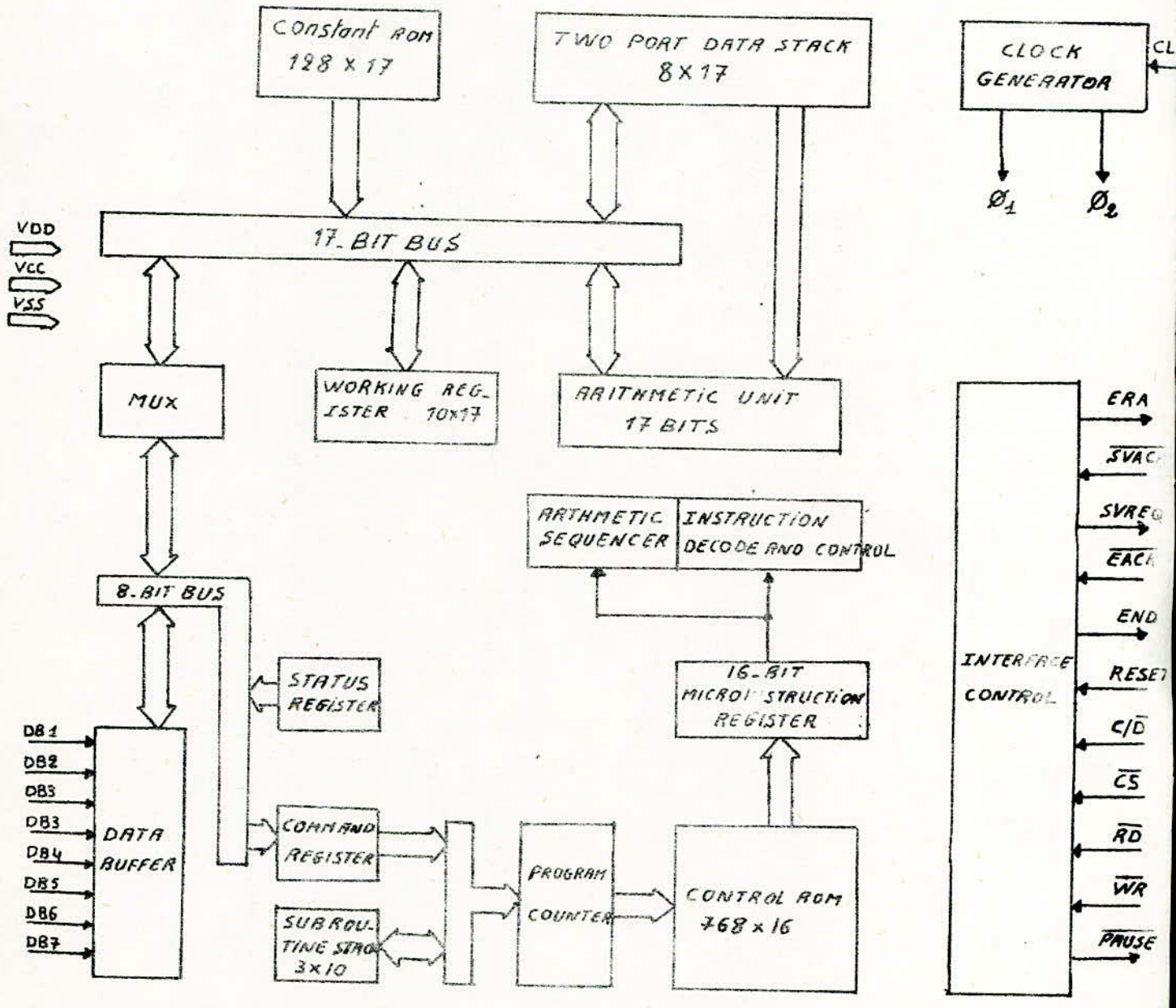
Lorsqu'il s'agit d'une utilisation intensive de ces opérations dans les problèmes scientifiques par exemple; le temps d'éxécution influe sur la rapidité de traitement du problème.

L'idéal pour le UP serait que les opérations arithmétiques occupent les dimensions de quelques instructions.

Nous adoptons donc l'utilisation d'une unité arithmétique rapide en virgule flottante qui va résoudre le problème de lenteur du UP MC 6800.

-O- PRESENTATION DE L'UNITE ARITHMETIQUE RAPIDE -O-

A M 9512



(fig.2-a)

Bloc diagramme de l'Am 9512

A - CARACTERISTIQUES GENERALES DE L'UNITE AM 9512

L'A M 9512 est un monochip LSI, en technologie MOS canal N à grille de silicium, qui se présente dans un boîtier de 24 broches (Voir fig. 2 - B). Il est destiné à accroitre sensiblement la capacité de calcul arithmétique des UPS 8 bits.

Son caractère universel permet de le connecter indifféremment à n'importe quel microprocesseur.

Il travaille à une fréquence d'horloge de 2 Mhz et demande deux tensions d'alimentation : + 5V et + 12V.

L'unité arithmétique rapide A M 9512 éxécute les 4 opérations fondamentales : addition; soustraction; division et multiplication sur des mombres représentés en code binaire; en virgule flottante et sous 2 formats :

- Format simple précision 32 bits
- Format double précision 64 bits.

1°) Broches d'Interface

En dehors du bus de données DB0 - DB7 et du signal d'horloge CLK, l'A M 9512 comprend les broches d'interface suivantes :

a) - Lignes d'entrée

* RESET : Sert à initialiser le composant; l'activation de cette ligne a pour conséquences :

- l'arrêt de toute opération en cours
- La RAZ du registre d'état
- Le positionnement du composant dans un état d'attente.

* C/$\overline{D}$ (Command/Data) : Cette entrée est le bit de sélection du registre utilisé en conjonction avec $\overline{RD}$ et $\overline{WR}$; donc indique le type de transfert à effectuer.

La fig. 2 - c donne le codage des transferts

* $\overline{CS}$: Chip select (Selection du boîtier) : cette entrée valide les échanges d'informations avec l'èxterieur.

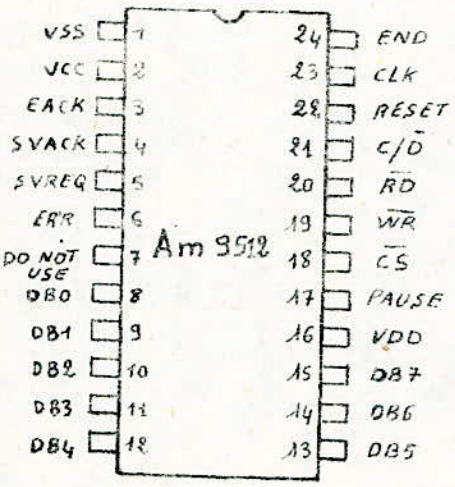
* $\overline{RD}$ Read (lecture) : c'est la commande de lecture activée par son niveau bas quand $\overline{CS}$ = 0 .

* $\overline{WR}$ write (Ecriture) : c'est la commande d'écriture activée par son niveau bas quand $\overline{CS}$ = 0 .

* $\overline{EACK}$: End Acknowledge (Acquittement de fin de traitement) : Cette commande d'entrée remet à zéro la sortie END.

Quand un niveau bas est appliqué au signal END, cette commande va aussi à zéro.

.../...



brochage de l'Am9512

(fig.2b)

C/D	RD	WR	FONCTION
L	H	L	entrée d'un byte donnée dans la pile
L	L	H	sortie d'un byte donnée de la pile
H	H	L	entrée d'une commande
H	L	H	lecture du registre d'etat
X	L	L	indefini

L : Low

H : High

X : Indéfini

commande des transferts (fig.3b)

* $\overline{SVACK}$ Service Acknowledge (Acquittement de fin de service): cette commande d'entrée remet à zéro la sortie SVREQ indiquant une demande de service.

b) - Lignes de sortie

* END end éxécution (fin d'éxécution) : Un haut niveau sur cette sortie indique que l'éxécution de la commande courante est finie.
Cette commande est mise à l'état bas par l'activation de la commande $\overline{EACK}$ à l'état bas ou par l'éxécution d'une quelconque commande de lecture ou d'écriture ou un procédé d'initialisation utilisant un reset.
Cette commande peut-être utilisée pour générer une demande d'interruption.

* SVREQ : Service request (demande de service): un haut niveau sur cette sortie, indique qu'un service post.opératoire a été demandé.
Peut-être mis à l'état bas soit par l'activation du $\overline{SVACK}$ ou par un pocédé d'initialisation utilisant un RE ET; soit de façon automatique après la fin de l'éxécution de toute commande ayant le bit 7 " SVREQ ENB " à zéro.
La sortie SVREQ sera à l'état haut à la fin de l'éxécution de la commande ayant ce bit à un .

* $\overline{PAUSE}$ pause (demande d'attente): cette sortie est activée dans les conditions suivantes :

* Une entrée de données est demandée alors que la donnée précédente n'est pas encore rangée en pile.

* Une entrée est demandée alors qu'une opération est en cours d'éxécution.

* Une sortie (de données ou de mots d'état) est demandée et l'information n'est pas encore valide sur le bus de données .

Cette sortie permet donc à l'U.A.R, de se synchroniser avec le microprocesseur maître par l'intermédiaire d'une procédure d'attente.

* ERR error (erreur) : l'activation de cette sortie indique une erreur de condition dans l'éxécution de la commande en cours . Les erreurs de condition sont :

- Une division par zéro,
- Un overflow de l'exposant ou un underflow.

La sortie ERR est mise à zéro après lecture du registre d'état ou un RESET .

.../...

B - DESCRIPTION FONCTIONNELLE

Le synoptique du composant représenté par la (fig. 2 - a) donne la majorité des unités fonctionnelles de l'AM 9512.

La totalité des transferts (données opérandes, données résultat, mot de commande ou mot d'état) s'effectuent par l'intermédiaire d'un bus bidirectionnel de 8 bits (DB0 - DB7).

Ces signaux entrent à travers des buffers de données.

La structure interne est celle d'un processeur 17 bits parallèles; sur ce bus interne de 17 bits sont connectés :

- L'unité arithmétique
- Une pile où sont stockés les opérandes et les résultats de calcul

Cette pile est organisée en 8 mots de 17 bits, avec le dernier entré, premier sorti (LIFO).

- Les registres de travail au nombre de 10, utilisés pour le stockage des valeurs intermédiaires pendant l'éxécution des opérations
- Une mémoire ROM de 128 constantes, founit les constantes éxigées pour éxécuter les opérations mathématiques .

Les opérations de l'Unité AM 9512 sont contrôlées par le microprogramme contenu dans une mémoire control .ROM.

- Le program counter fournit les adresses des microinstructions et peut être partiellement chargé par le registre de commande.
- Une pile de subroutine connectée au program counter stocke les adresses de retour, lors d'un appel de subroutine.
- Le chargement du registre de commande fournit l'adresse de départ dans la mémoire du microprogramme.
- Un registre d'état rassemble un certain nombre d'indications, indiquant l'état du composant à la suite de l'éxécution d'une opération déterminée.

a) - Fonctionnement de la pile de données :

Les opérandes et les résultats de calcul sont rangés dans une pile LIFO organisée en 4 niveaux de 32 bits ou 2 niveaux de 64 bits, en fonction du format de travail.

Avant le début d'une opération les opérandes sont rangés dans la pile, octet de poids faible en tête .

L'opération elle même s'effectue implicitement sur les opérandes situés au sommet de la pile TOS (TOP OF STACK) est à l'emplacement suivant, NOS (Next of Stack). A la fin de l'opération, le résultat se retrouve au sommet de la pile, après une éventuelle opération de rotation (Pop Stack) implicitement contenue dans le code opératoire.

L'acquisition du résultat par le UP s'effectue par lecture de la pile, octet de poids fort en tête, suivant le format choisi :

- 4 bytes pour la simple précision
- 8 bytes pour la double précision.

b) - Registre de commande :

Le registre de commande (voir fig d - 2) est chargé par l'octet de commande spécifiant le type de traitement à effectuer.
Le format de commande est de 8 bits; le bit 7 est le service request SVREQ = SR, il indique si la commande est assortie ou non d'une demande de service; les 6 premiers bits indiquent le code opération.
Les mots de commande de l'AM 9512 se divisent en 3 catégories : simple précision, double précision et manipulation de données .
Les 4 opérations de base (addition, soustraction, multiplication et division) se font en virgule flottante en simple ou double précision.
Le tableau (fig. e - 1) représente les fonctions réalisées par l'AM 9512, les notations suivantes ont été adoptées :

- A = TOS représente le sommet de la pile et (A) son contenu.
- B = NOS représente l'emplacement suivant et (B) son contenu.
- PUSH STACK : indique une opération d'enfoncement dans la pile (avec perte de l'opérande situé à la base de la pile).
- POPSTACK : indique une opération d'extraction de la pile.

c) - Registre d'Etat :

le résultat de la dernière opération éffectuée par l'AM 9512 affecte le contenu du registre d'état. Celui-ci comprend 8 bits, dont 6 indicateurs (voirfig. d-1)
Bit 0 : réservé
Bit 1 : dépassement de capacité de l'exposant (OVERFLOW).
Bit 2 : sous-dépassement de capacité de l'exposant (UNDERFLOW).
Bit 3 : division par zéro.
Bit 4 : reservé.
Bit 5 : zéro, indique si le TOS est nul.
Bit 6 : signé du TOS.
Bit 7 : Busy, quand il est à 1, indique qu'une opération est en cours d'éxécution. Les différents bits du registre d'état sont validés quand le bit busy est à 0.

.../...

registre d'état

busy	SIGN S	zero Z	reserv	div par 0	UNDER FLOW	OVER FLOW	reserv

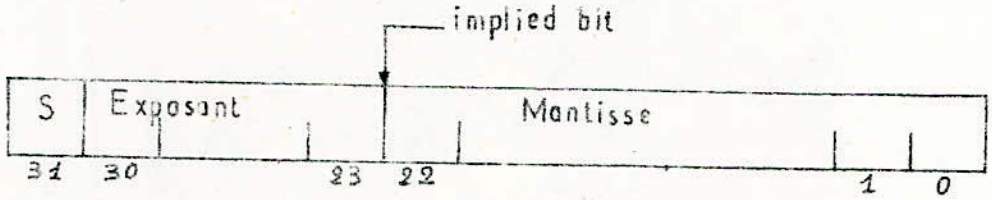
(fig.d-1)

registre de commande

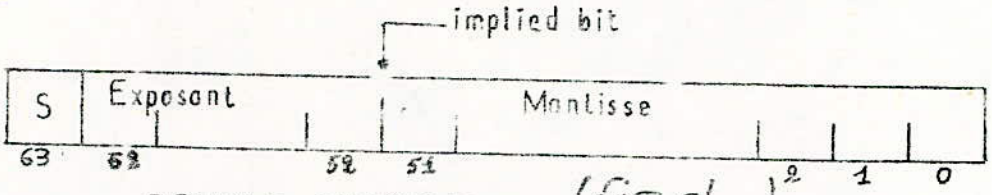
syreq enb	code operation

(fig.d-2)

format des données



SIMPLE PRECISION (fig.d-3)



DOUBLE PRECISION (fig.d-4)

temps d'exécution des opérations en cycles

	Min	Typ	Max
ADD	58	220	512
SOUST	56	220	512
MULT	192	220	254
DIV	228	240	264

SIMPLE PRECISION

	Min	Typ	Max
ADD	578	1200	3100
SOUST	578	1200	3100
MULT	1720	1770	1860
DIV	4560	4920	5120

DOUBLE PRECISION

(fig.d-5)

C - REPRESENTATION DES NOMBRES :

L'AM 9512 travaille sur des nombres binaires représentés en virgule flottante. Dans ce type de représentation, un nombre N s'écrit de la façon suivante :

$N = M. 2^E$ où M est la mantisse et E l'exposant.

Le nombre de bits de la mantisse fixe la précision limite des calculs et le nombre de bits de l'éxposant détermine l'étendue du codage.

La forme dite normalisée, permet de conserver le maximum de bits significatifs.

a) - Format simple précision

Le format simple utilisé ici, est de 32 bits (voir fig. d - 3)

- Le bit 31 = S

S : signe de la mantisse. 1 représente le négatif et 0 le positif.

- Les bits 23 à 30 = E.

Ces 8 bits représentent l'exposant codé, qui est égal à sa valeur augmenté du bias exposant.

bias exposant = $2^7 - 1 = 127$

L'exposant codé = $E + (2^7 - 1)$

- Les bits 0 à 22 = M

Ces 23 bits représentent l'amplitude de la mantisse en binaire, la virgule se trouvant à gauche de la mantisse.

Une conséquence de la normalisation fait que le 1er bit après la virgule est toujours positionné à 1 .

Ce bit appelé bit implicite est à chaque fois sous entendu; donc le premier bit significatif est le bit 22 et la valeur absolue de la mantisse est comprise entre 1 et 2.

Pendant l'éxécution interne d'une opération, le bit implicite est restauré, et à la fin de l'opération le résultat est normalisé et le bit implicite sous entendu. La virgule de la mantisse se situe entre le bit 22 et le bit implicite.

On présente le nombre codé N par la notation :

$$N = (-1)^S \, 2^{E - \underbrace{(2^7 - 1)}_{\text{bias}}} \underbrace{(1. M)}_{\text{virgule}}$$

.../...

Exemple de conversion

exemple 1 : N = 0 10000011 11000000000000000000000

SIGNE — exposant — mantisse

Exposant codé = 10000011

Exposant actuel = 10000011 - 01111111 = 00000100 = $4_{(10)}$

Mantisse = 1.11000000000000000000000

= $1 + \frac{1}{2} + \frac{1}{4} = 1{,}75_{(10)}$

Le nombre N en décimal = $2^4 \times 1{,}75 = 28_{(10)}$

exemple 2 : N 0 11110100 11000000000000000000000

SIGNE — exposant — mantisse

Exposant codé = 01111010

Exposant actuel = 01111010 - 01111111 = 11111011 = $-5_{(10)}$

Mantisse = 1.01100000000000000000000

= $1 + \frac{1}{4} + \frac{1}{8} = 1{,}375_{(10)}$

$N_{(10)} = 2^{-5} \times 1{,}375 = -0{,}429687_{(10)}$

b) - Format double précision :

Pour ce format le nombre de bits est de 64 (fig. d-4)

- Le bit 63 = S ; signe de la mantisse
- Les bits 52 à 62; ces 11 bits représentent l'exposant codé, dont la valeur est supérieure à la valeur réelle de celle du bias éxposant. Ici le bias est égal à $2^{10} - 1 = 1023$ donc l'exposant codé est $E + (2^{10} - 1)$.
- bits 0 à 51 : ces 52 bits représentent la mantisse. Le bit 51 est précédé du bit implicite égal à 1 et sous entendu .

Un nombre en double précision est réprésente par :

$$N = (-1)^S \; 2^{E - (2^{10} - 1)} \; (1 . M)$$

$E - (2^{10} - 1)$: bias exposant

$1 . M$: Virgule

.../...

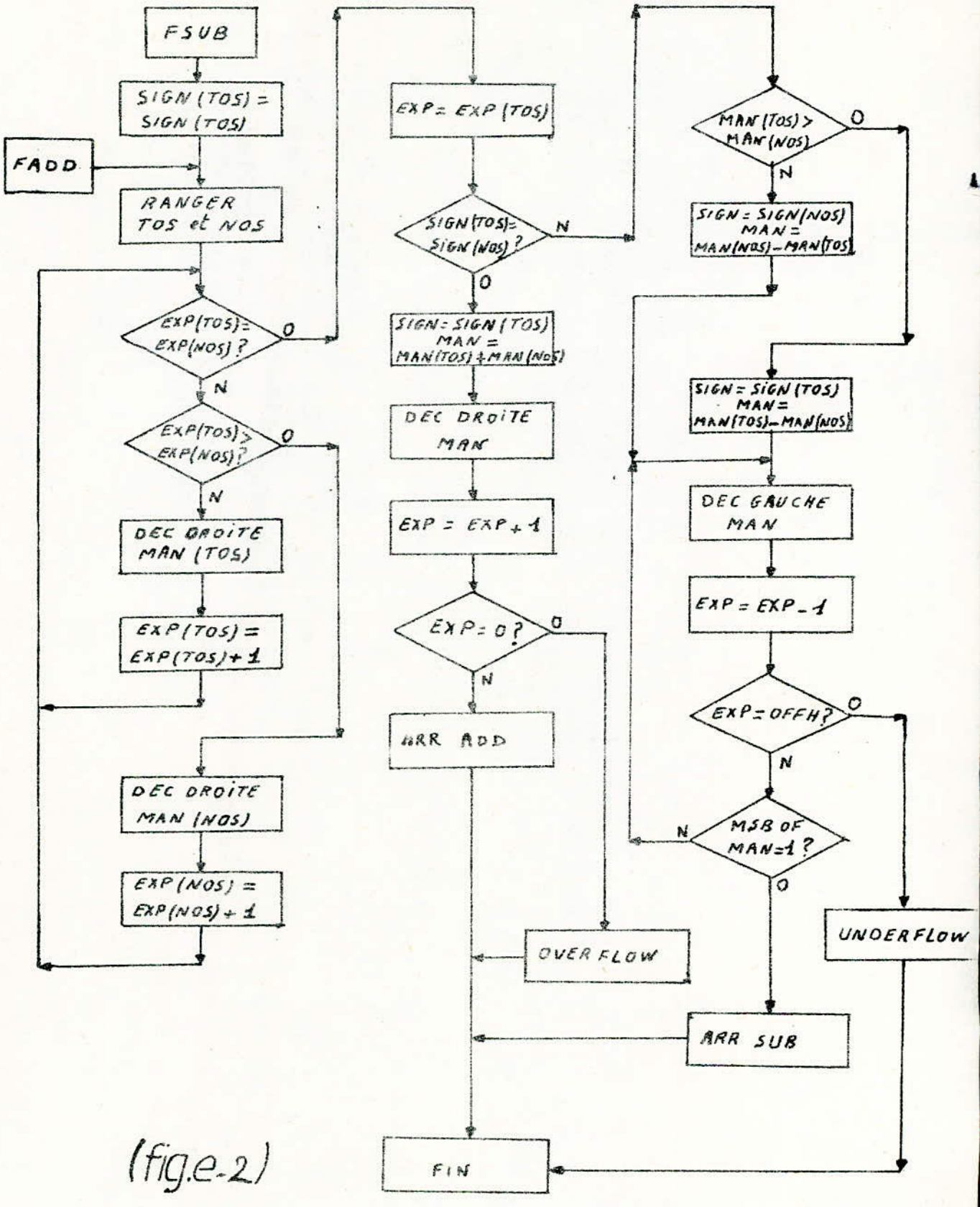
D - ALGORITHMES DES OPERATIONS EN VIRGULE FLOTTANTE :

Dès leur entrée dans l'unité, les opérandes sont rangés dans la pile, le signe et l'exposant occupent respectivement un et 8 bits pour la simple précision, et la mantisse 24 bits avec le MSB égal à 1.
Pour les différents algorithmes on utilisera les abréviations suivantes :
MSB : bit le plus significatif du nombre
Sign : signe du résultat
Exp : exposant du résultat
Man : mantisse du résultat
Sign. (TOS) : signe du sommet de la pile
Sign. (NOS) : signe du nombre dans l'emplacement suivant de la pile
Exp (TOS) : Exposant du sommet de la pile
Man (TOS) : Mantisse du sommet de la pile
Exp (NOS) : Exposant du nombre dans l'emplacement suivant de la pile
Man (NOS) : Mantisse du nombre dans l'emplacement suivant de la pile.

a) - Addition, soustraction en virgule flottante :

L'addition et le soustraction utilisent essentiellement le même algorithme. La seule différence est que pour la soustraction, le signe du TOS est chargé avant de poursuivre l'algorithme d'addition (fig. e - 2) celui-ci se résume à :
a) - ranger TOS et NOS
b) - l'exposant du TOS est comparé à l'exposant du NOS
c) - Si les exposant sont égaux, passer à f
d) - décalage à droite de la mantisse du nombre dont l'exposant est le plus petit.
e) - Incrémenter l'exposant le plus petit et revenir en b.
f) - mettre le signe du résultat égal au signe du nombre le plus grand.
g) - Mettre l'exposant du résultat égal à l'exposant du nombre le plus grand.
h) - Si les signes des 2 nombres sont différents, aller en m.
i) - additionner les mantisses des 2 nombres .
j) - décalage d'une position à droite de la mantisse du résultat et incrémentation de son exposant.
k) - si le MSB de l'exposant du résultat passe de 1 à 0, à la suite de l'incrémentation, le bit d'état overflow est mis à 1 .
l) - arrondissement de la mantisse si nécessaire et sortie du résultat .
m) - soustraction des mantisses des 2 nombres
n) - décalage à gauche de la mantisse du résultat et décrémentation de son exposant.

.../...



(fig.e-2)

addition - soustraction VF

o) - si le MSB de l'exposant passe de 0 à 1 à la suite de la décrémentation mise à 1 du bit d'état underflow et sortie du résultat.

p) - si le MSB de la mantisse du résultat est égal à 0, aller en n.

q) - arrondissement de la mantisse si nécessaire et sortie du résultat.

b) - Multiplication en virgule flottante :

La multiplication en VF se base sur l'addition des exposants et la multiplication des mantisses (voir fig. e - 3)

a) - Tester si TOS ou NOS est égal à 1.

b) - Si TOS ou NOS est nul, sortie du résultat égal à 0.

c) - Ranger TOS et NOS

d) - Conversion des exposants de TOS et NOS à la forme unbiased

Exp. (TOS) = Exp. (TOS) - $127_{(10)}$

Exp. (NOS) = Exp. (NOS) - $127_{(10)}$

e) - Exposant du résultat égal à la somme des exposants

Exp. = Exp (TOS) + Exp (NOS)

f) - Si le MSB de l'exposant de TOS et le MSB de l'exposant de nos sont égaux à 0, avec MSB de l'exposant du résultat égal à 1, mise à 1 du bit overflow et sortie du résultat.

g) - Si le MSB de l'exposant de TOS et le MSB de l'exposant de NOS sont égaux à 1, et si le MSB de l'exposant du résultat égal à 0, mise à 1 du bit d'underflow et sortie du résultat.

h) - Reconversion de l'exposant du résultat égal à 0, mise à 1 du bit d'underflow et sortie du résultat .

i) - Si les signes de TOS et NOS sont égaux, mettre le signe du résultat égal à 0, sinon le mettre à 1 .

j) - Multiplication des mantisses des 2 nombres.

k) - Si le MSB de la mantisse du résultat est égal à 1, décalage de celle-ci à droite d'une position et incrémentation de l'exposant du résultat.

l) - Si le MSB de l'exposant du résultat passe de 1 à 0 à la suite de l'incrémentation, mise à 1 du bit d'overflow.

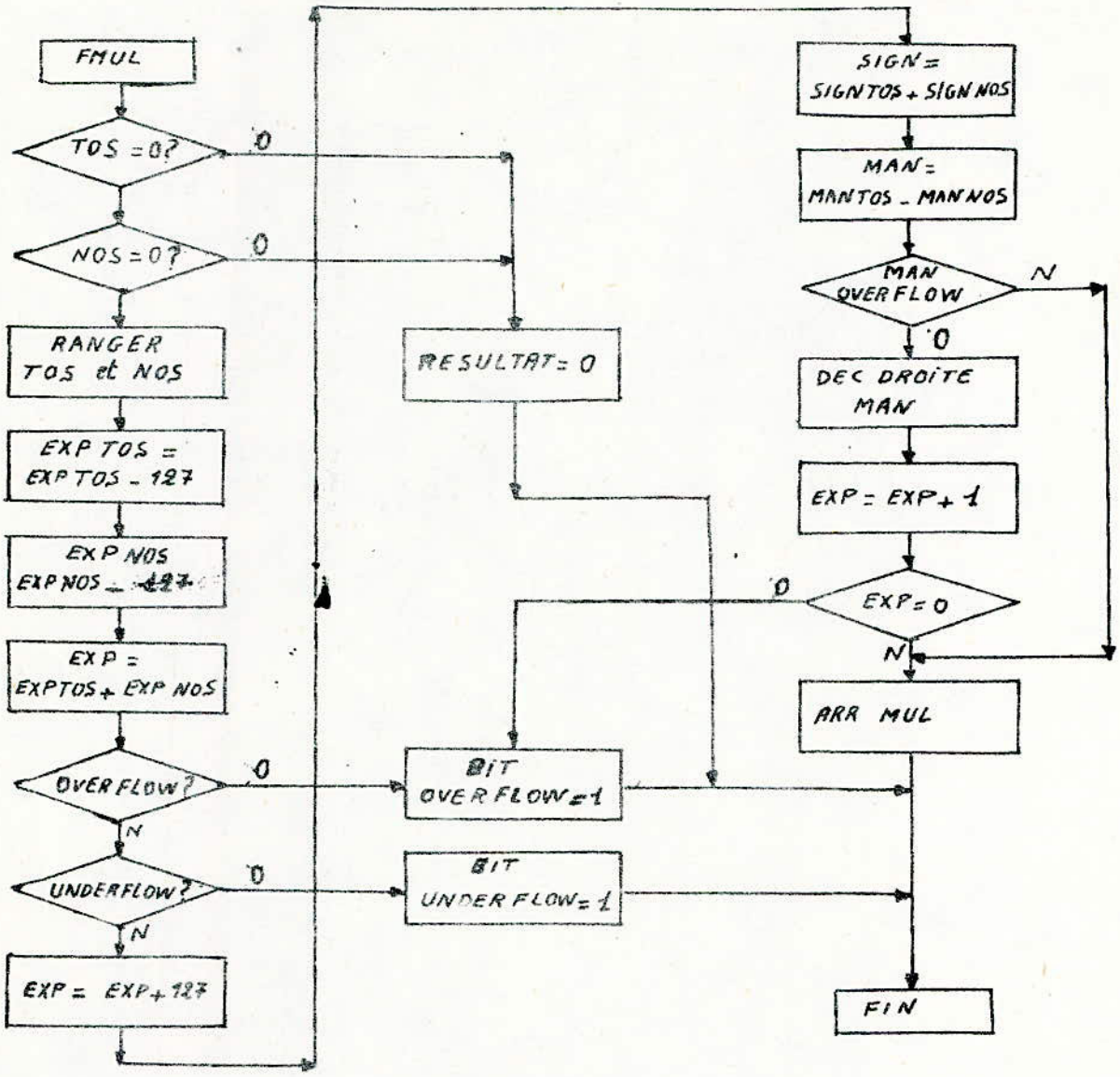
m) - arrondissement de la mantisse si nécessaire et sortie du résultat

c) - Division en VF :

Celle-ci se résume à la division des mantisses et à la soustraction des exposants (fig. e - 4)

a) - Si TOS est nul, mise à 1 du bit d'état division par zéro et sortie du résultat .

.../...



multiplication en virgule flottante *(fig. e-3)*

b) - Si NOS est nul, sortie du résultat égal à 0.

c) - Ranger TOS et NOS

d) - Conversion des exposants de TOS et NOS à la forme unbiased.

Exp (TOS)= Exp (TOS) - $127_{(10)}$

Exp (NOS)= Exp (NOS) - $127_{(10)}$

e) - Soustraction des exposants de TOS et NOS

Exp = Exp (NOS) - Exp (TOS)

f) - Si le MSB de l'exposant de NOS est nul, le MSB de l'exposant de TOS est à 1, si le MSB de l'exposant du résultat est égal à 1, mise à 1 du bit d'overflow et sortie du résultat.

g) - Si le MSB de l'exposant de NOS est égal à 1, si le MSB de l'exposant de TOS est égal à 0, et si le MSB de l'exposant du résultat est égal à 0, mise à 1 du bit d'underflow et sortie du résultat.

h) - Addition du bias à l'exposant du résultat.

exp = exp + $127_{(10)}$

i) - Si les signes de TOS et NOS sont égaux, mettre le signe du résultat égal à 0, s'ils sont différents, mettre le signe du résultat à 1.

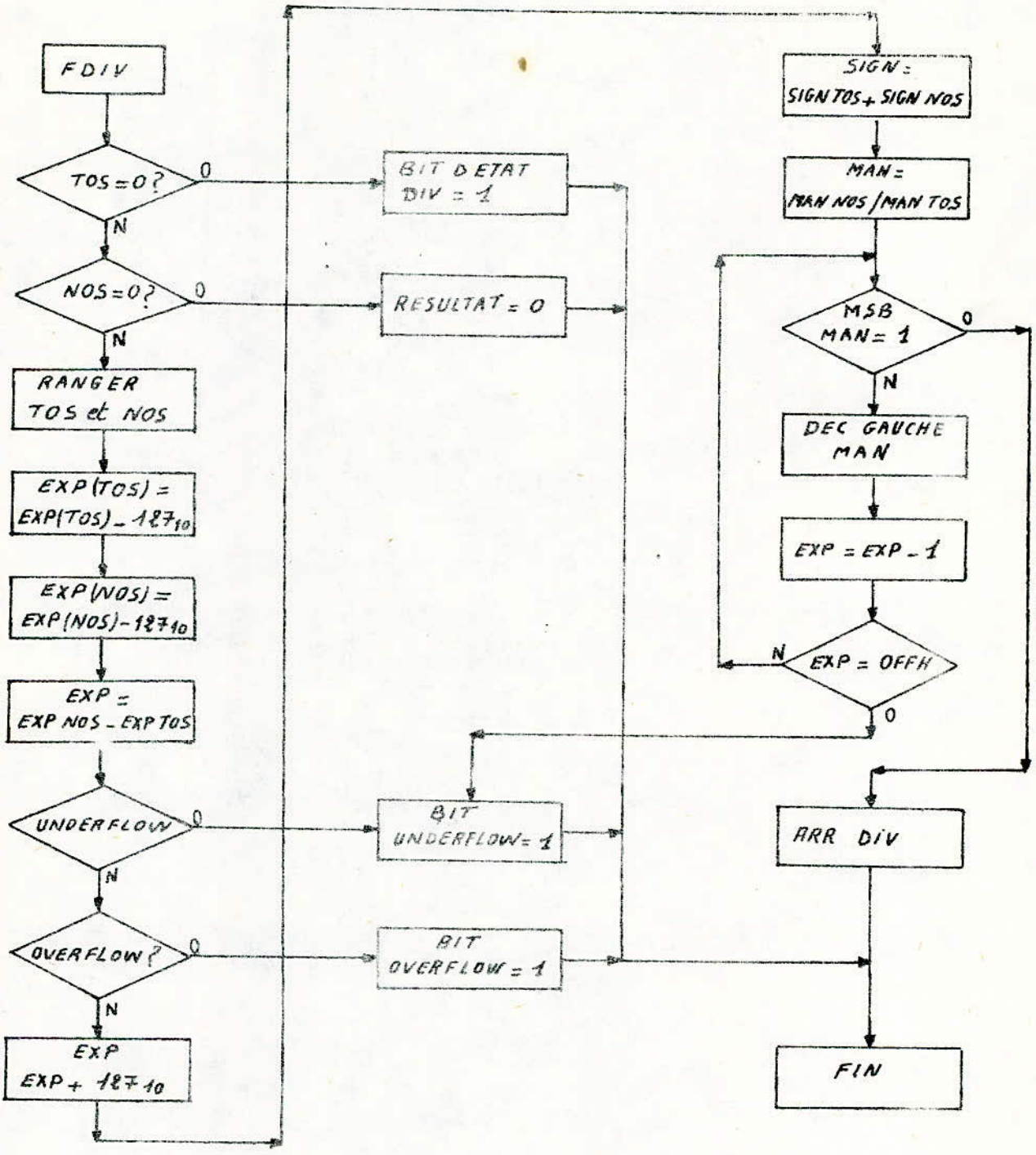
j) - Division de la mantisse de NOS par celle de TOS.

k) - Si le MSB de la mantisse du résultat est nul, décalage à gauche de celle-ci et décrémentation du résultat, sinon aller en n.

l) - Si le MSB de l'exposant du résultat passe de 0 à 1 à la suite de la décrémentation, mise à 1 du bit d'underflow.

m) - aller en k

n) - arrondissement de la mantisse si nécessaire et sortie du résultat



division en virgule flottante (fig-4)

MOT DE COMMANDE 7 6 5 4 3 2 1 0	MNEMONIQUE	FONCTION REALISEE	
X 0 0 0 0 0 0 1	SADD	(A) + (B) ⟶ B SP	POP STACK
X 0 0 0 0 0 1 0	SSUB	(B) − (A) ⟶ B SP	
X 0 0 0 0 0 1 1	SMUL	A × B = P ⟶ B SP	
X 0 0 0 0 1 0 0	SDIV	(B) : (A) = Q ⟶ B SP	
X 0 0 0 0 1 0 1	CHSS	INVERSION SIGNE (A) SP	
X 0 0 0 0 1 1 0	PTOS	PUSH STACK (A) ⟶ A SP	
X 0 0 0 0 1 1 1	POPS	POP STACK SP	
X 0 0 0 1 0 0 0	XCHS	ECHANGE entre (A) et (B) SP	
X 0 1 0 1 1 0 1	CHSD	INVERSION SIGNE (A) DP	
X 0 1 0 1 1 1 0	PTOD	PUSH STACK (A) ⟶ A DP	
X 0 1 0 1 1 1 1	POPD	POP STACK DP	
X 0 0 0 0 0 0 0 0	CLR	REGISTRE D'ETAT = 0	
X 0 1 0 1 0 0 1	DADD	(A) + (B) ⟶ B DP	POP STACK
X 0 1 0 1 0 1 0	DSUB	(B) − (A) ⟶ B DP	
X 0 1 0 1 0 1 1	DMUL	(A) × (B) = P ⟶ B DP	
X 0 1 0 1 1 0 0	DDIV	(B) : (A) = Q ⟶ B DP	

fonctions réalisées par l'Am 9512 (fig. E-1)

PTOD

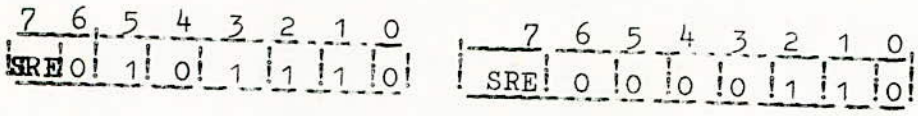
PTOS

ENFONCEMENT DANS LA PILE

DOUBLE PRECISION (D.P)

SIMPLE PRECISION (S.P)

Code binaire

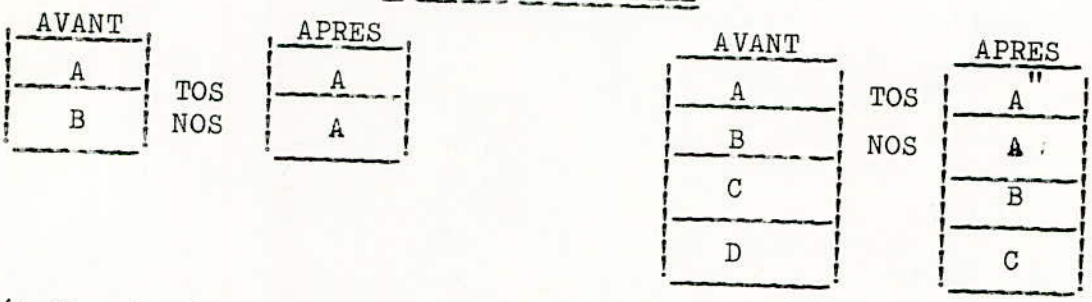


Code héxadécimal

AE SI SRE = 1
2E SI SRE = 0

86 SI SRE = 1
06 SI SRE = 0

CONTENU DE LA PILE



NOTE/: A" = A si le champ de l'exposant de A est non nul
A" = 0 si le champ de l'exposant de A est nul.

DESCRIPTION

L'opérande A de TOS (DP ou SP) est enfoncé dans la pile, ceci est effectivement une duplication de A dans deux positions consécutives dans la pile. Cependant si l'opérande (SP) de TOS qui précede la commande PTOS a seulement son champ d'exposant nul alors le nouveau TOS sera mis à 0. Le contenu de NOS sera une copie exacte de l'ancien TOS.

Les bits d'état S et Z sont affectés pour reporter le signe du nouveau TOS, et si le contenu de TOS est 0, respectivement les bits d'état U,V et D sont mis à zéro. Les bits d'état affectés sont : S,Z (U,V,D toujours à zéro.)

ECHANGER TOS ET NOS SIMPLE

code binaire

7	6	5	4	3	2	1	0
SRE	0	0	0	1	0	0	0

code hexadécimal

88 SI SRE = 1

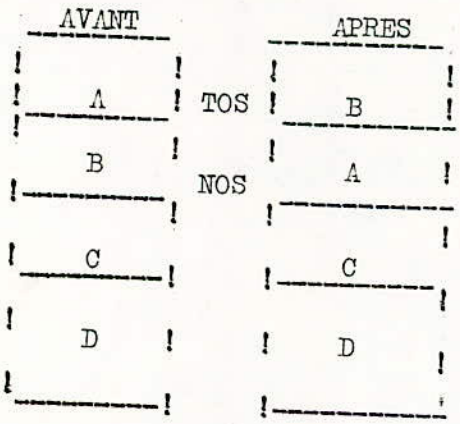
08 SI SRE = 0

DESCRIPTION

L'opérande A de TOS (SP) et l'opérande B de NOS (SP) sont échangés.

Après l'exécution, B est à TOS et A est à NOS. Tous les autres opérande sont inchangés. Les bits d'état affectés sont : S, Z, (U, V et D toujours à zéro).

CONTENU DE LA PILE



CHSD

CHSS

CHANGEMENT DE SIGNE

DOUBLE PRECISION (D.P)

SIMPLE PRECISION (S.P)

code binaire

7	6	5	4	3	2	1	0
SRE	0	1	0	1	1	0	1

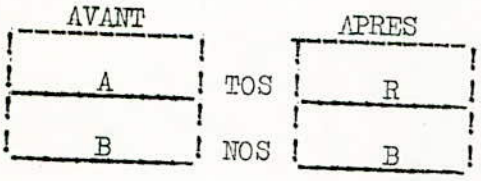
7	6	5	4	3	2	1	0
SRE	0	0	0	0	1	0	1

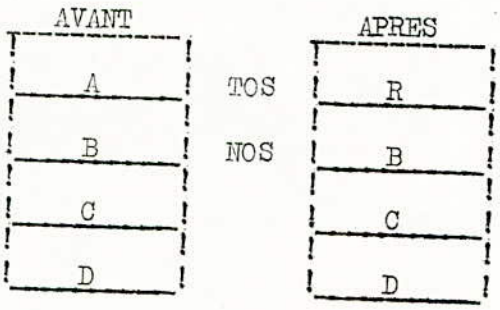
code hexadécimal

AD SI SRE = 1
2D SI SRE = 0

85 SI SRE = 1
05 SI SRE = 0

CONTENU DE LA PILE





DESCRIPTION

Le signe de l'opérande A de TOS (DP ou SP) est complément à 2; le résultat R (DP ou SP) se trouve à TOS. Si l'opérande (DP) est nul alors le signe n'est pas affecté. Si l'exposant de A (SP) est nul, tous les bits de R sont mis à zéro. Les bits d'état S et Z indiquent le signe du résultat et si le résultat est nul.

Les bits U, V et B sont toujours à zéro.

Les bits d'états affectés sont S, Z (U, V, B = 0).

/ DMUL /

/ SMUL /

MULTIPLICATION EN VIRGULE FLOTTANTE

DOUBLE PRECISION (D.P)

SIMPLE PRECISION (S.P)

code binaire

7	6	5	4	3	2	1	0
SRE	0	1	0	1	0	1	1

7	6	5	4	3	2	1	0
SRE	0	0	0	0	0	1	1

code hexadécimal

AB SI SRE = 1

2B SI SRE = 0

83 SI SRE = 1

03 SI SRE = 0

CONTENU DES LA PILE

AVANT		APRES
A	TOS	R
B	NOS	Indéfini

AVANT		APRES
A	TOS	R
B	NOS	C
C		D
D		Indéfini

DESCRIPTION

L'opérande A de TOS (DP ou SP) est multiplié par l'opérande B de NOS. Le résultat est arrondi pour obtenir le résultat final R (DP ou SP) qui est remis dans TOS.

Les bits d'états S,Z,U et V sont affectés pour reporter le signe du résulat, si le résultat est zéro, l'exposant underflow et l'exposant overflow respectivement. Le bit d'état C est mis à zéro.

/ DDIV / / SDIV /

DIVISION EN VIRGULE FLOTTANTE

DOUBLE PRECISION (D.P) SIMPLE PRECISION(S.P)

code binaire

7	6	5	4	3	2	1	0
SRE	0	1	0	1	1	0	0

7	6	5	4	3	2	1	0
SRE	0	0	0	0	1	0	0

code hexadécimal

AC SI SRE = 1 84 SI SRE = 1

2C SI SRE = 0 04 SI SRE = 0

CONTENU DE LA PILE

AVANT		APRES
A	TOS	R
B	NOS	Indéfini

AVANT		APRES
A	TOS	R
B	NOS	C
C		D
D		Indéfini

DESCRIPTION

L'opérande B de NOS (DP ou SP) est divisé par l'opérande A de TOS. Le résultat (quotient) est arrondi pour obtenir le résultat final R (SP ou DP) qui est remis à TOS.

Les bits d'état S,Z,C,U et V sont affectés pour reporter le signe du résultat si le résulat est zéro, un essai de division par zéro, l'exposant underflow et l'exposant overflow respectivement.

/ CLR /

MISE A ZERO DU REGISTRE D'ETAT

code binaire

7	6	5	4	3	2	1	0
SRE	0	0	0	0	0	0	0

code héxadécimal

80 SI SRE = 1

00 SI SRE = 0

DESCRIPTION

Les bits d'états S,Z, U,V sont mis à 0.

La pile n'est pas affectée.

Les bits d'Etats sont à zéro tant qu'il n'y a pas d'opération de commande qui met en jeu les opérandes. Les bits d'Etat affectés sont : S,Z,D,U,V, toujours à zéro.

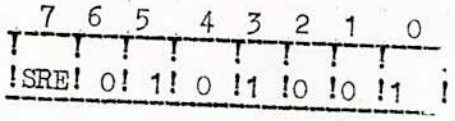
/DADD/

/SADD/

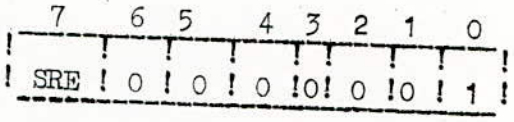
ADDITION EN VIRGULE FLOTTANTE

DOUBLE PRECISION (D.P.)

Binary coding :



SIMPLE PRECISION(S.P)



Hexadécimal coding :

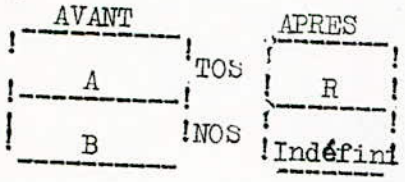
A9 SI SRE = 1

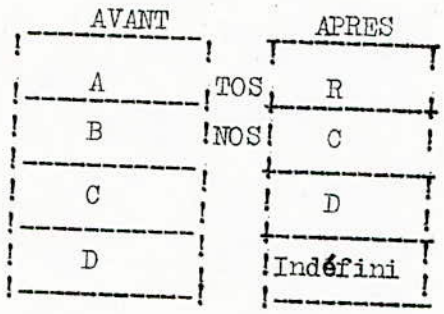
29 SI SRE = 0

81 SI SRE = 1

01 SI SRE = 0

CONTENU DE LA PILE





DESCRIPTION

L'opérande A de TOS (DP ou SP) est additionné à l'opérande B de NOS (DP ou SP). Le résultat est arrondi pour abtenir le résultat final R qui est remis dans TOS. Les bits d'état S;Z,U et V sont affectés pour reporter le signe du résultat si le résultat est zéro; l'exposant underfow et l'exposant overflow respectivement. Le bit d'état D est mis à zéro; les bits d'état affectés sont S,Z,U,V (D est toujours à zéro).

EXTRACTION DE LA PILE

DOUBLE PRECISION (D.P)

SIMPLE PRECISION (S.P)

Code binaire

7	6	5	4	3	2	1	0
SRE	0	1	0	1	1	1	1

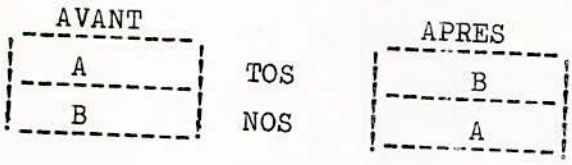
7	6	5	4	3	2	1	0
SRE	0	0	0	0	1	1	1

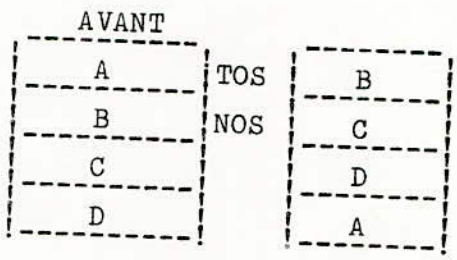
Code héxadécimal

AF SI SRE = 1
2F SI SRE = 0

87 SI SRE = 1
07 SI SRE = 0

Contenu de la pile





Cette opèration a le meme
éffet que X.C.H.S.

DESCRIPTION

l'opérande A (DP ou SP) est extrait de la pile et il est réecrit au bas de la pile.
Les bits d'état S et Z sont affectés pour reporter le signe du nouvel opérande de TOS et s'il est égal à zéro, respectivement les bits U,V,D sont mis à zéro.
Si le champ de l'éxposant du nouveau TOS (S.P) est nul alors si le bit d'état Z est zéro, il sera mis à 1; les bits d'état affectés sont : S,Z (U,V,D toujours à 0).

DSUB

SSUB

SOUSTRACTION EN VIRGULE FLOTTANTE

DOUBLE PRECISION (D.P)

code binaire

7	6	5	4	3	2	1	0
SRE	0	1	1	1	0	1	0

code hexadécimal

AA SI SRE = 1

2A SI SRE = 0

SIMPLE PRECISION

7	6	5	4	3	2	1	0
SRE	0	0	0	0	1	1	0

82 SI SRE = 1

02 SI SRE = 0

CONTENU DE LA PILE

AVANT		APRES
A	TOS	R
B	NOS	INDEFINI

AVANT		APRES
A	TOS	R
B	NOS	C
C		D
D		INDEFINI

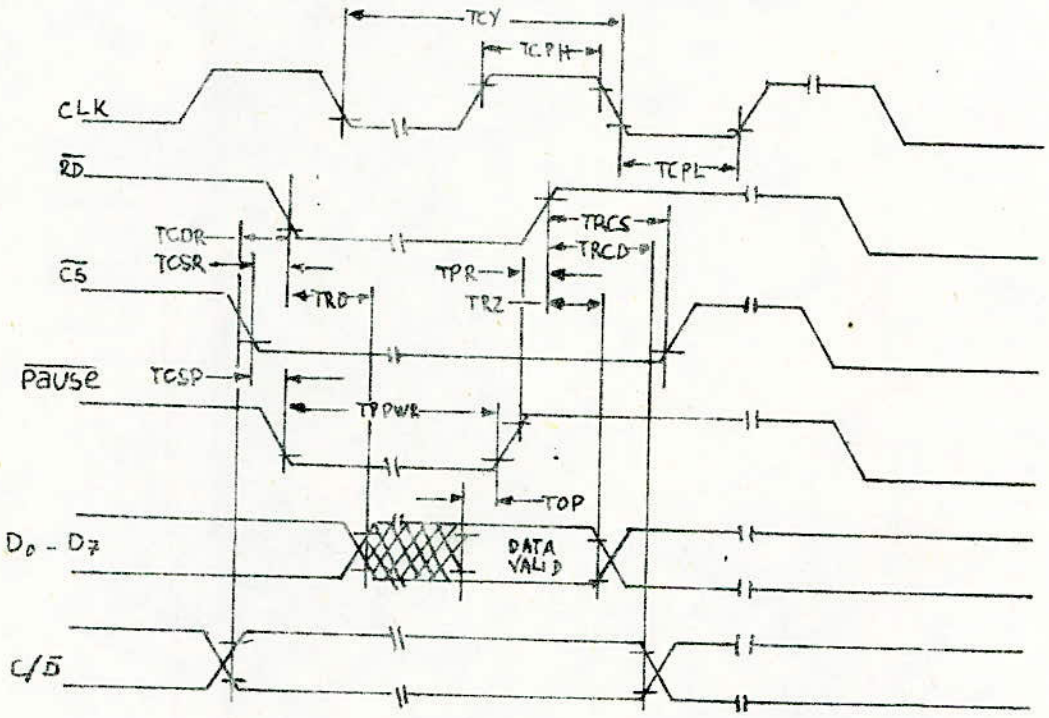
DESCRIPTION

L'opérande A de TOS (SP ou DP) est soustrait à l'opérande B de NOS. Le résultat est arrondi pour obtenir le résultat final (DP ou SP) qui est remis dans TOS.

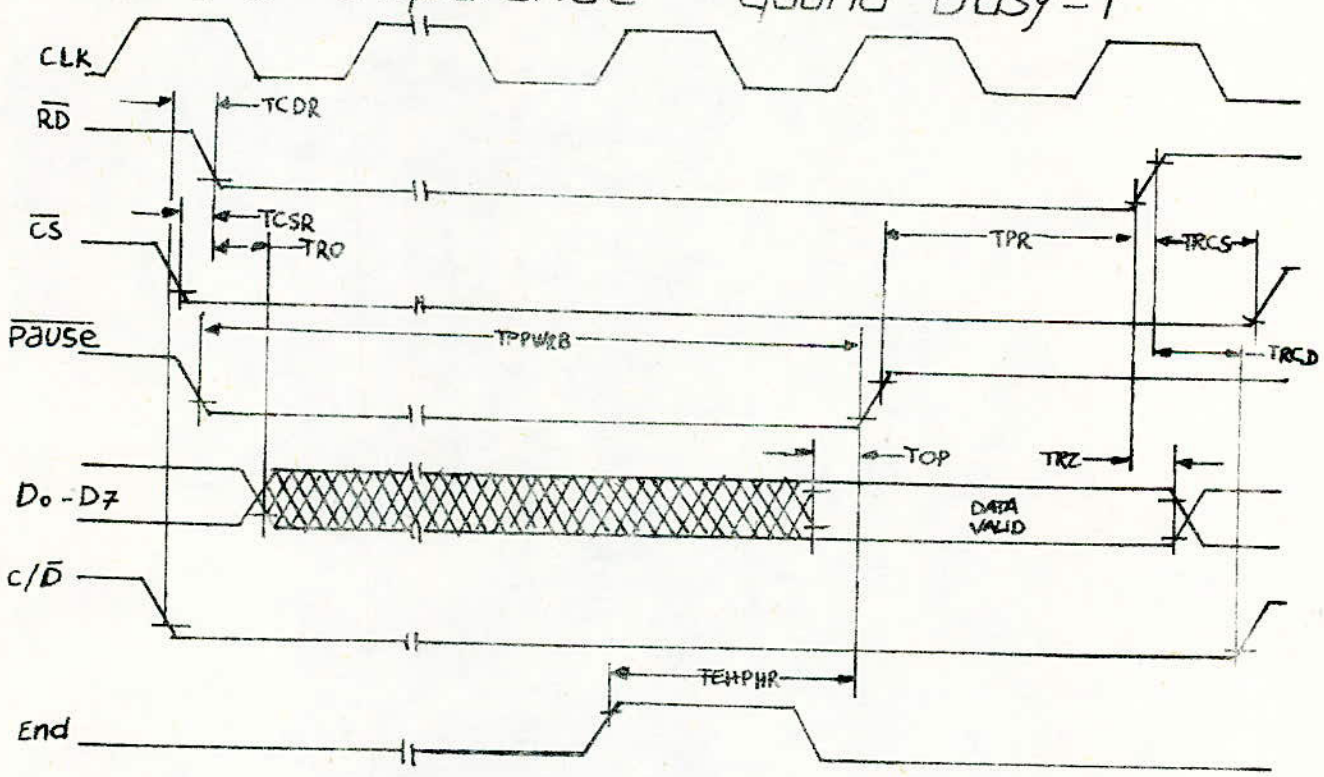
Les bits d'états S,Z,U et V sont affectés pour reporter le signe du résultat. Si le résultat est zéro, l'éxposant underflow et l'éxposant overflow respectivement le bit D remis à zéro.

Les bits d'états affectés sont : S,Z,U,V (D est toujours à zéro).

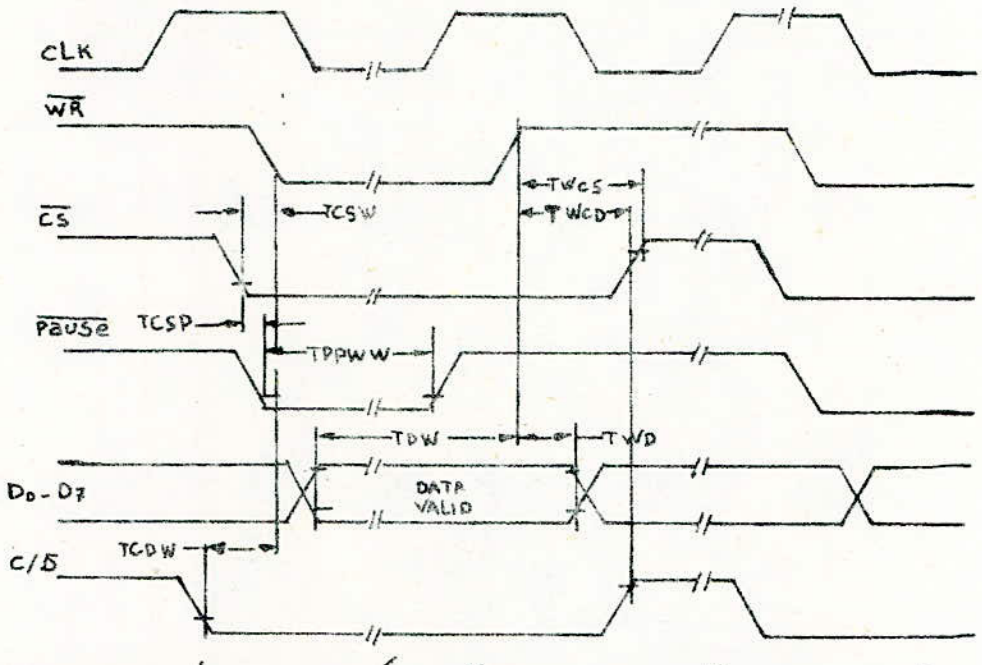
opération de lecture



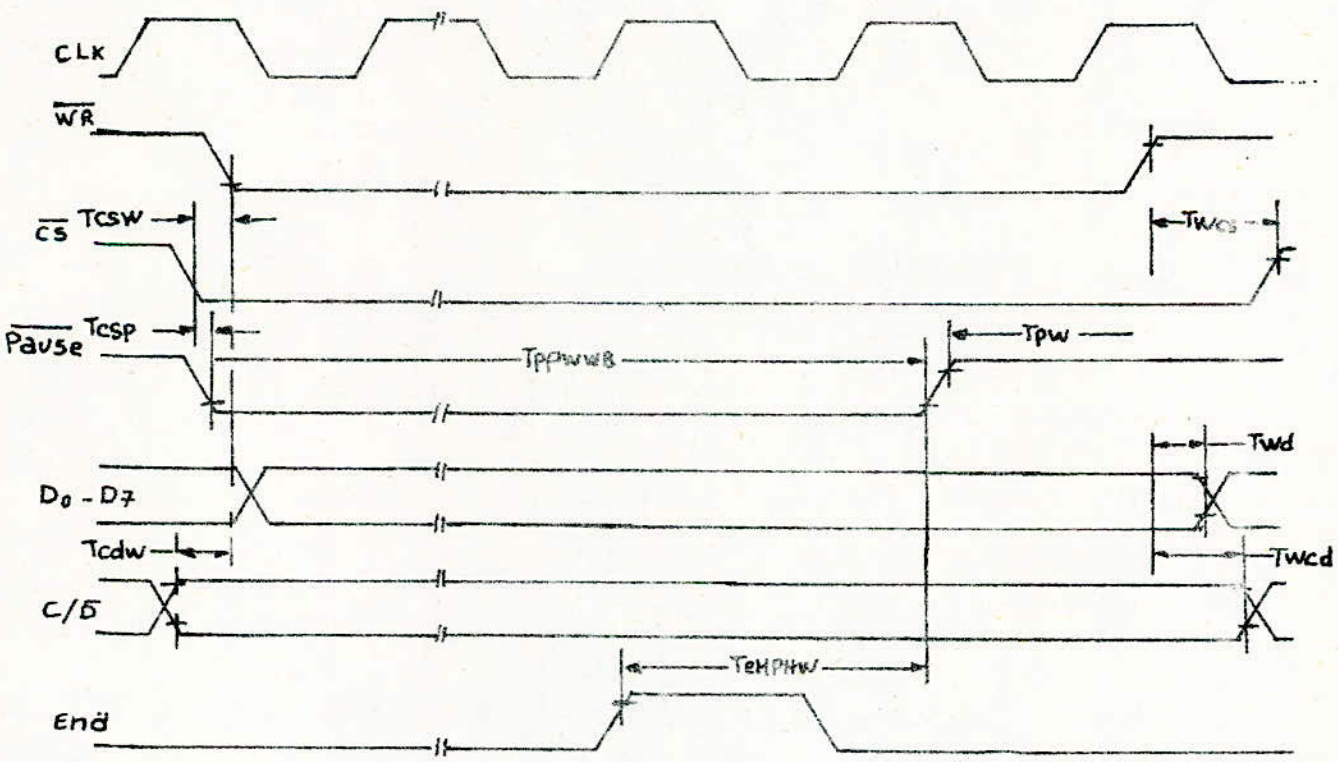
Lecture d'opérande quand busy=1

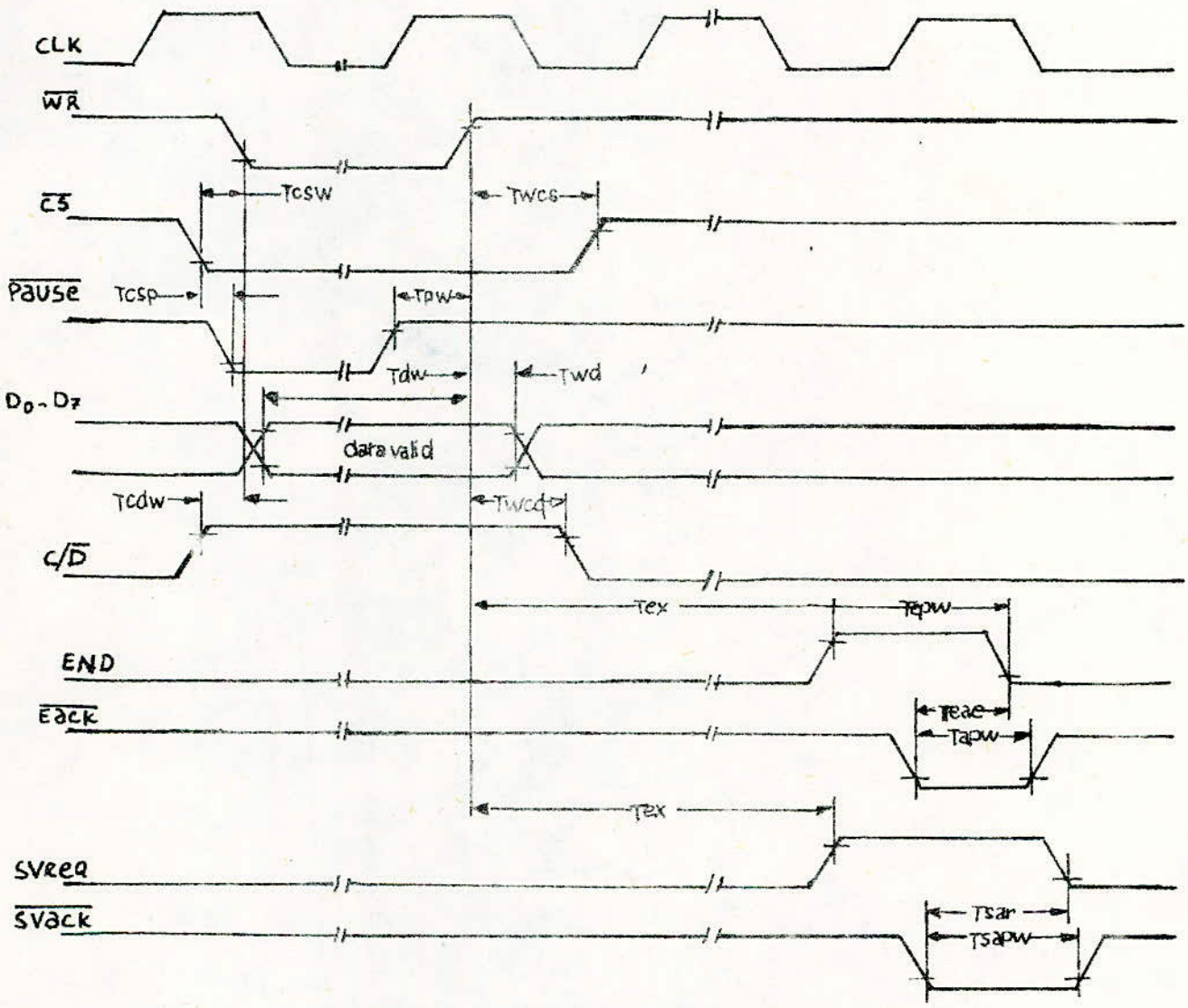


entreé d'opérandes



commande ou écriture d'une donneé quand l'Am 9512 est occupé





initialisation d'une commande

COMMANDE	TOS	NOS	RESULTAT	PERIODES
SADD	3F800000	3F800000	40000000	58
SSUB	3F800000	3F800000	00000000	56
SMUL	40400000	3FC00000	40900000	198
SDIV	40000000	3F800000	3F000000	238
CHSS	3F800000	—	BF800000	10
PTOS	3F800000	—	—	16
POPS	3F800000	—	—	14
XCHS	3F800000	4000000	—	26
CHSD	3FF0000000000000	—	BFF0000000000000	24
PTOD	3FF0000000000000	—	—	40
POPD	3FF0000000000000	—	—	26
CLR	3FF0000000000000	—	—	4
DADD	3FF0000000000000	8000000000000000	3FF00000A0000000	578
DSUB	3FF0000000000000	8000000000000000	3FF00000A0000000	578
DMUL	BFF8000000000000	3FF8000000000000	C0020000000000000	1748
DDIV	BFF8000000000000	3FF8000000000000	BFF0000000000000	4560

Execution de certains exemples

TOS, NOS et RESUL en HEXADECIMAL

PERIODES en DECIMAL

-O- COUPLAGE DE L'U.A.R (Am 9512) AU UP MC 6800 -O-

A-DECODAGE D'ADRESSE

Lorsque le CPU place une adresse de 16 bits sur le bus d'adresse,elle ne doit activer qu'une position memoire ou un interface d'E/S.
Avec 16 bits nous disposons de 64K positions mémoires,nous avons alloué à la mémoire locale et commune une capacité de 32K positions chacune.
Seulement lors de la réalisation de la carte micro-systeme,nous nous sommes limitées à 12K positions que nous avons subdivisé en des positions de 2K octets chacune :
(2K RAM ;2K ROM ET 8K POUR L'U.A.R)
Cela étant largement suffisant pour les tests de bon fonctionnement et l'extension restant toujours possible.
L'espaceemémoire adopté est schématisé par la (fig..3-b)
Le circuit de décodage depend du type de boitiers utilisés.
Les décodeurs permettent d'élaborer les signaux de selection des differents boitiers de telle façon que le boitier ou l'ensemble de boitiers qui contiennent le mot désiré soit validé..
Le décodage choisi est celui par démultiplexage, nous avons opté pour le circuit démultiplexeur SN 74LS138 (voir schéma de brochage et table de vérité fig.3 a)
Ce circuit ne peut décoder que 8 boiters vu qu'il ne possède que 8 sorties
Afin de décoder les 16 boitiers, nous avons pris deux circuits SN 74 LS 138 qui ont été différenciés par la ligne A 14 du bus d'adresse.
Ils sont validés à travers l'entréé ET par le signal:

$\overline{A15}.VMA.\emptyset 2$ TTL

Comme nous ne voulons selectionner que 6 boitiers (RAM,ROM ET 4 U.A.R), il n'y a que 6 sur les 8 sorties du décodeur qui ont été utilisées,les autres n'ont pas été connectées.
Les fils d'adresse: A0-A15 (11 fils) ont servi à adresser les 2K positions mémoire de chaque circuit (voir tableau III).

B-ACTIVATION ET DESACTIVATION DES BUFFERS DE DONNEES ET D'ADRESSE

Le micro-système que nous avons réalisé peut travailler en zone commune ou en zone locale..
L'accés en mémoire commune se faisant à travers les circuits 8T26 pour les données et 8T95 pour les adresses.
La solution envisagée a été de les valider si nous travaillons en zone commune et de les mettre en haute impédance (désactiver) sinon.

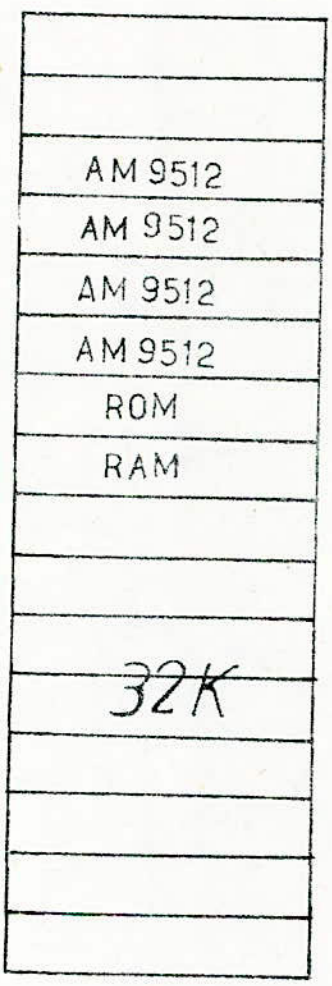
1°-Activation et désactivation des 8T26

Sachant que la mémoire locale occupe un espace mémoire de 32K positions à partir de l'adresse (0000 à 7FFF)et que la mémoire commune occupe le même

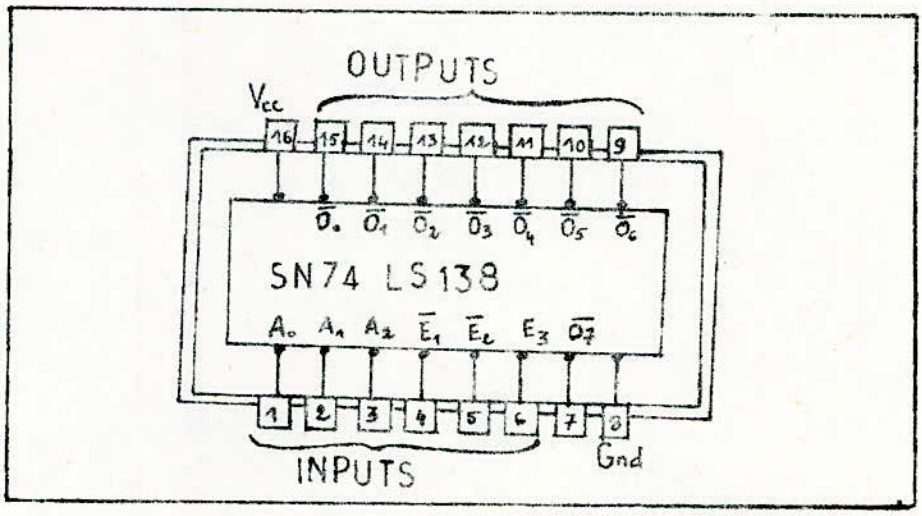
.../...

Circuit	Adresses	Combinaison $A_1\ A_2\ A_3$	Combinaison $\overline{E_1}\ \overline{E_2}\ E_3$	Sortie
RAM	4000-47FF	0 0 0	0 0 1	$\overline{O_0}$
ROM	4800-4FFF	0 0 1	0 0 1	$\overline{O_1}$
AM 9512	5000-57FF	0 1 0	0 0 1	$\overline{O_2}$
AM 9512	5800-5FFF	0 1 1	0 0 1	$\overline{O_3}$
AM 9512	6000-67FF	1 0 0	0 0 1	$\overline{O_4}$
AM 9512	6800-6FFF	1 0 1	0 0 1	$\overline{O_5}$

TABLEAU III. DE DÉCODAGE DES CIRCUITS



(fig.3-b) PAGE MEMOIRE



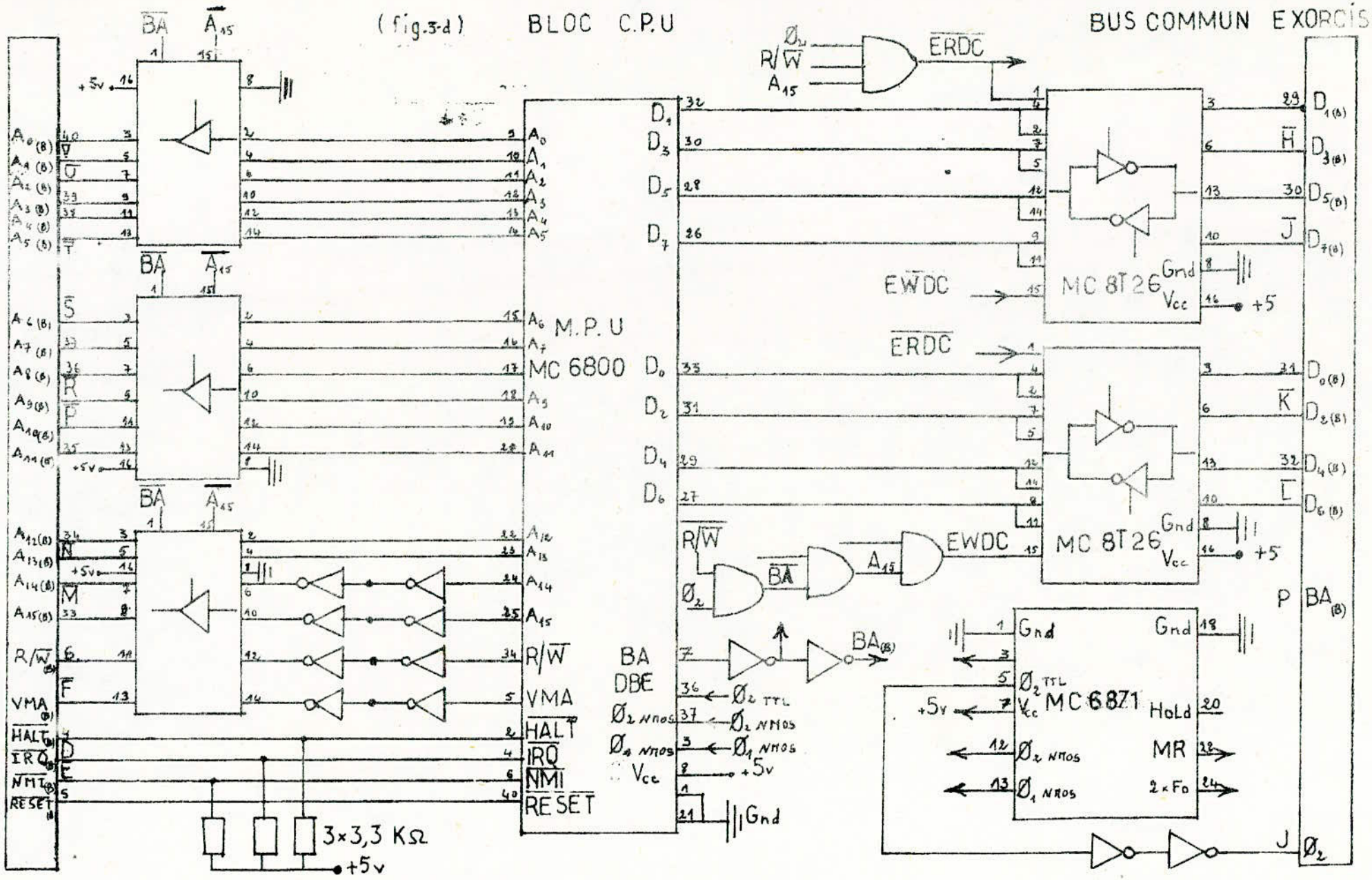
(fig.3-a)

TABLE DE VÉRITÉ

INPUTS						OUTPUTS							
E_1	E_2	E_3	A_0	A_1	A_2	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H

(fig.3-d) BLOC C.P.U

BUS COMMUN EXORCISER



espace à partir de l'adresse (8000 à FFFF).
L'etat de la ligne d'adresse A15 (0 ou 1) nous indique le passage d'une zone à l'autre.
Ainsi A15 adjointe à la logique de lecture et d'écriture décrite suivant la (fig.3-c) constitue le circuit d'aiguillonnage vers une zone mémoire ou l'autre..
D'aprés la table de vérité (fig.3-c) nous voyons que:
-En zone commune (A15=1); l'opération de lecture est donnée par l'etat 00 sur les lignes 1 et 15 des 8T26.
L'opération d'écriture est donnée par l'état 11 sur les lignes 1 et 15 des 8T26; donc ces deux états correspocdent bien à la table de vérité donnée.
-En zone locale (A15=0); dés que A15 passe à 0,nous avons la configuration 10 respectivement sur les lignes 1 et 15 des 8T26 ,les mettant en haute impédance.

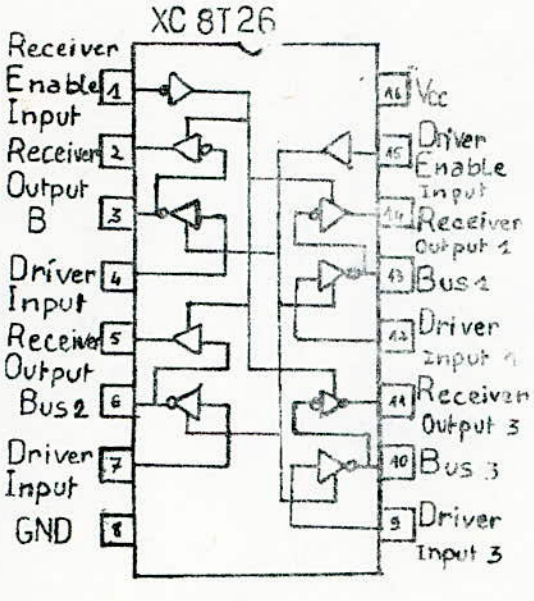
2°-Activation et désactivation des 8T95

La table de vérité des 8T95 (fig.3-d), montre que la mise à l'état bas des lignes 1et 15 permet de retrouver l'adresse d'entree en sortie; alors que leur mise à l'état haut place le buffer à l'état haute impédance.
La ligne d'adresse A15 permettant de savoir dans quelle zone le C.P.U travaille, sera utilisée comme commande des 8T95 en effet:
-En zone locale; le micro-système "esclave " travaille dans sa propre zone les lignes 1 et 15 sont à l'état haut et les 8T95 sont en haute impédance.
-En zone commune (A15 =1); le micro-système "esclave" travaille en dehors de sa zone propre, les lignes 1 et 15 sont à l'état bas et les 8T95 sont validés.

C-DECODAGE D'ADRESSE DE L'AM 9512

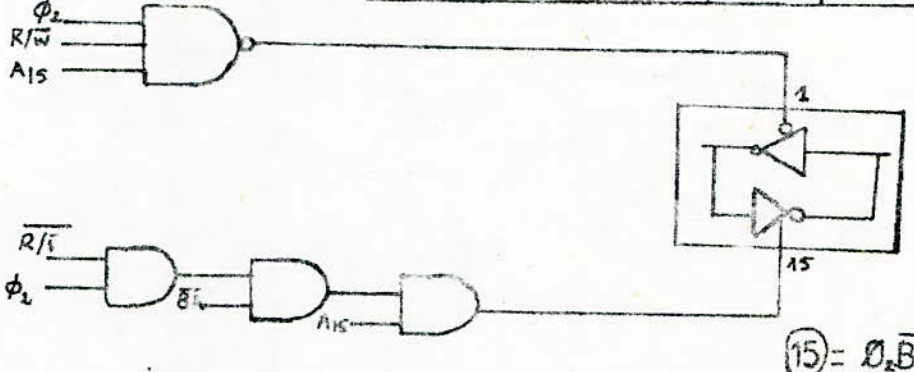
L'U.A.R (Am 9512) est vue par le microprocesseur MC 6800 comme un boitier de 2 positions mémoires validé par la sortie $\overline{O2}$ du décodeur ;le circuit SN 74 LS 138.
Cette sortie correspond à l'adresse 5000 (pour la 1ère unité câblée) et l'adresse 0101$\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi A_o$ (pour les autres unitées qui sont reliées aux autres sorties du décodeur par leur broche $\overline{CS}$.
Une fois ce circuit selectionné, la ligne A_o reliée à l'entrée C/$\overline{D}$ (commande /data); indique à l'U.A.R si le transfert concerne:
-Un octet de commande soit A_o=1; correspondant à l'adresse 5001.
Ce transfert peut-être une écriture dans le registre de commande si nous faisons un stockage à l'adresse 5001 ,comme il peut-être une lècture du registre d'état,lorsque nous chargeons l'accumulateur de l'adresse 5001.
-Un octet de données soit A_o=0; correspondant à l'adresse 5000 (accés à-la pile de l'AM 9512).

.../...



(fig.3-c)

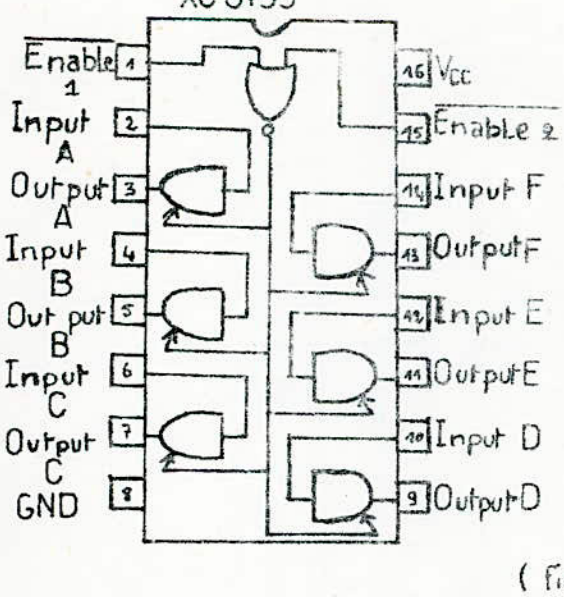
R/$\overline{W}$	$\emptyset_2$	A15	$\overline{BA}$	PIN ①	PIN ⑮
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0



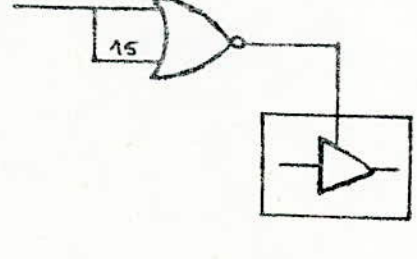
LOGIQUE DE COMMANDE DES 8T26 :

⑮ = $\emptyset_2 \overline{BA} \cdot R/\overline{W} \cdot A_{15}$

① = $\overline{\emptyset_2 R/\overline{W} \cdot A_{15}}$



A 15	état du bus
1	valide
0	HI



(fig.3-d)

Le stockage des opérandes dans la pile est effectué par une instruction d'écriture à l'adresse 5000.

La restitution du résultat de l'opération est effectué par une instruction de lecture à l'adresse 5000.

D) - LOGIQUE DE COMMANDE ET DE CONTROLE

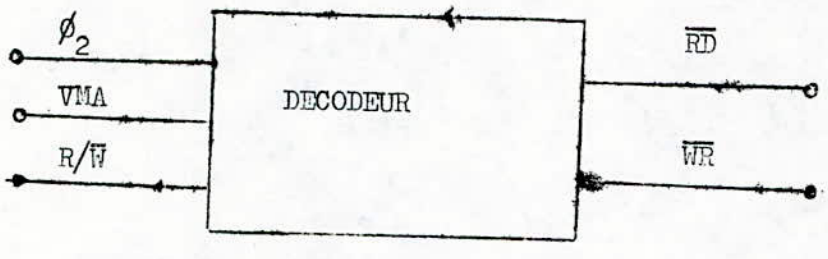
La sortie $2xf_o$ de l'horloge MC 6871 utilisée la carte, délivre un signal de 2Mhz et comme l'U.A.R travaille à cette frequence, nous relions donc directement cette sortie à l'entrée CLK de l'AM 9512.

La ligne $\overline{RESET}$ du UP est inversée et reliée à la ligne RESET de l'U.A.R.

a) circuit de lecture/écriture:

Ce circuit determine s'il faut autoriser un transfert ou une reception de données, suivant qu'il reçoit une commande de lecture ou d'écriture grâce aux signaux R/$\overline{W}$, VMA et ϕ_2 sur les interfaces de données.

Comme l'U.A.R possède deux entrées $\overline{RD}$ et $\overline{WR}$ qui valident respectivement La lecture et l'écriture,donc il est necessaire de créer ces signaux à partir des signaux VMA,ϕ_2 , et R/$\overline{W}$ du UP suivant le schéma suivant:



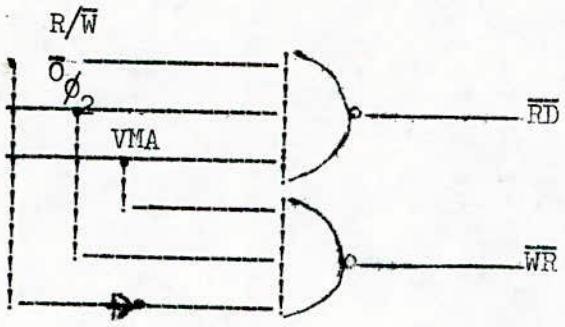
Pour cela nous avons dresse le tableau de vérité suivant:

VMA	ϕ_2	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$
0	0	0	HP	HP
0	0	1	HP	HP
0	1	0	HP	HP
0	1	1	HP	HP
1	0	0	HP	HP
1	0	1	HP	HP
1	1	0	1	0
1	1	1	0	1

HP: Haute impédance

.../...

A partir de cette table de vérité, nous déduisons le circuit logique suivant:



$$\overline{RD} = \overline{VMA.\phi_2.R/\overline{W}}$$

$$\overline{WR} = \overline{VMA.\phi_2.\overline{R/\overline{W}}}$$

Opération de lecture:

I'opération de lecture n'a lieu que si le signal de lecture $\overline{RD}$ est validé; pour cela il faut que le signal de commande $R/\overline{W}$ soit mis à l'état haut et les signaux ϕ_2 et VMA mis à l'état haut; il en résulte que le signal de lècture est donné par :

$$\overline{\overline{RD} = R/\overline{W}.\phi_2.VMA}$$

Opération d'écriture:

Cette opération n'à lieu que si le signal $R/\overline{W}$ est mis à l'état bas, la commande $R/\overline{W}$ de même et les signaux ϕ_2 et VMA mis à l'état haut.
Le signal permettant l'écriture résulte alors de la réunion de toutes ces conditions;il est donné par:

$$\overline{\overline{WR} = \overline{R/\overline{W}.\phi_2.VMA}}$$

E-INDICATEUR DE L'ETAT DE TRANSFERT

La sortie $\overline{PAUSE}$ permet l'échange de données entre le UP et l'U.A.R AM 9512. Cette sortie est activée:

-Pour toute opération de lecture ou d'écriture déclenchée et se maintiendra à cet état tant que le transfert n'est pas terminé.

-Si une entrée de données est demandée, alors que la donnée précédente n'est pas encore rangée en pile.

-Si une entrée est demandée,alors qu'une opération est en cours d'éxécution.

La sortie $\overline{PAUSE}$ permet donc de synchroniser l'U.A.R avec le UP par une procédure d'attente; qui est assurée en agissant sur le circuit horloge (MC 6871)afin de prolonger l'état "1" de ϕ_2.

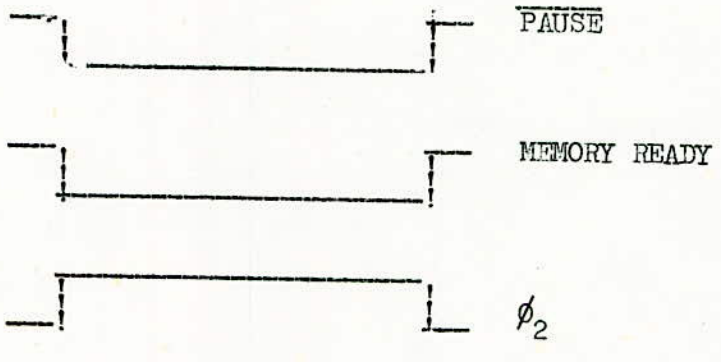
Le signal $\overline{PAUSE}$ est relié à la sortie (MEMORY READY) de l'horloge à travers une bascule et 2 inverseurs.

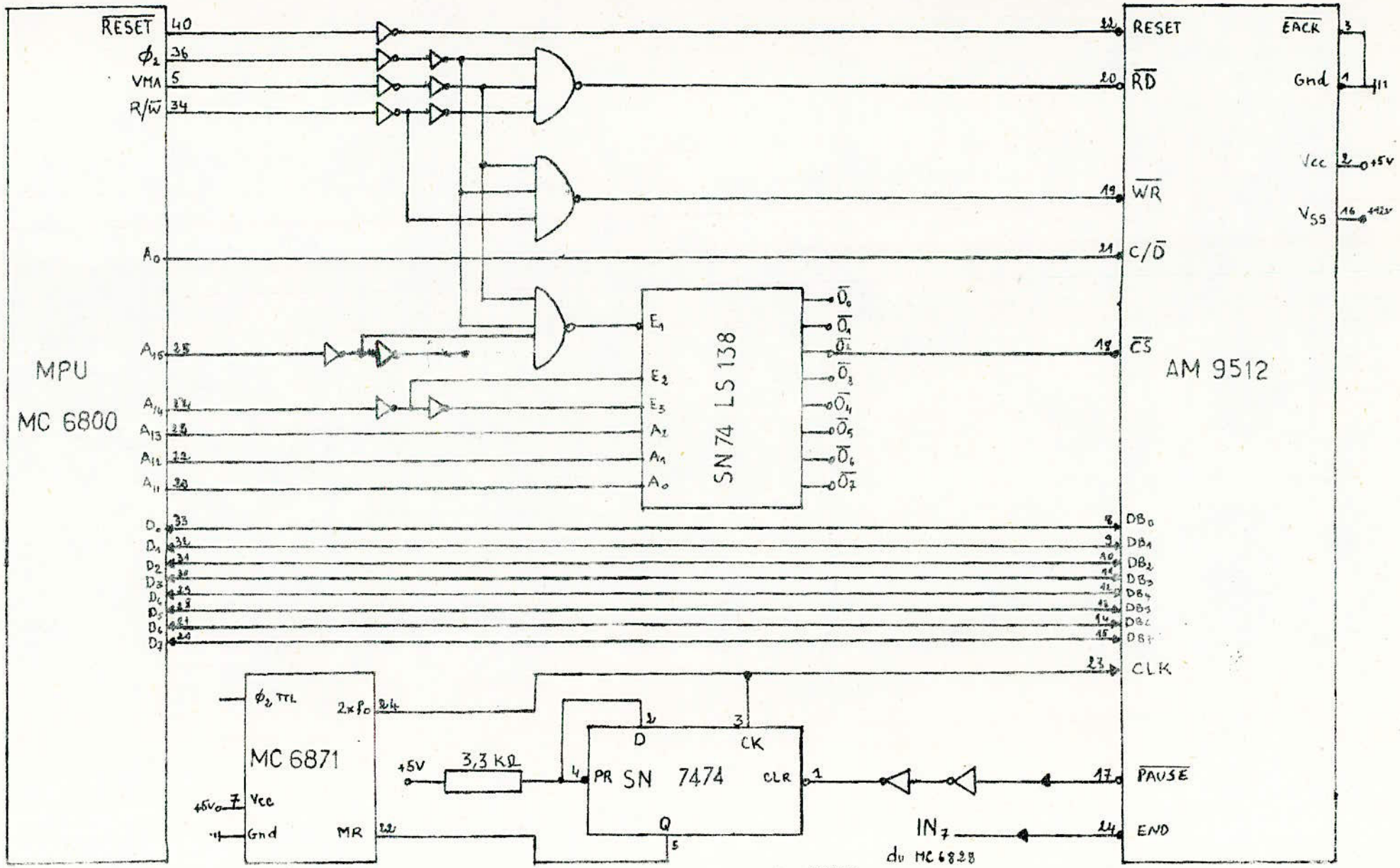
.../...

La bascule utilisée est une bascule D,à une seule entrée DATA.
Les entrées DATA et PRESET sont reliées à 5v à travers une resistance; l'entrée CLK est reliée à la sortie 2xf de l'horloge;sur lequel évolue la bascule D au front montant.
L'entrée CLEAR est reliée au signal $\overline{\text{PAUSE}}$ apres avoir été retardé par 2 inverseurs; cette entrée remet à zéro la bascule à chaque fois que le signal $\overline{\text{PAUSE}}$ est activé;

L'horloge MC 6871

MOTOROLA a concu le circuit horloge MC 6871 qui délivre deux signaux ϕ_1 et ϕ_2 et qui est capable de prolonger sur commande l'état haut de ϕ_2. Grâce à l'entrée MEMORY READY de ce circuit, l'état haut du signal ϕ_2 généré est déterminé en fonction du temps de cycle de transfert de l'AM 9512.Ce qui est schématisé ci-dessous:





(fig.3-8) SCHÉMA DE COUPLAGE DE L'U.A.R AM 9512 ET DU MC 6800

PROGRAMME D'INTRODUCTION DE DONNEES DANS L'AM 9512

Cette unité destinée a accroitre les performances des microprocesseurs par sa rapidité d'éxecution et la précision des résultats; elle sera donc solicitée pour éffectuer certains calculs de méthodes numériques.
Pour une première application nous avons éssayé d'éxecuter l'une des quatres opérations fondamentales sur l'AM 9512.
Les 2 opérandes se trouvent dans des mémoires fixes M_1 et M_2 disposées de la manière suivantes :

En M_1 : le 1er (-) significatif byte de l'opérande (LSB_1)

$M_1 + 1$: Le 2eme (-) significatif byte de l'opérande (LSB_2)

$M_1 + 2$: Le 1er (+) significatif byte de l'opérande (MSB_1)

$M_1 + 3$: Le 2eme (+) significatif byte de l'opérande (MSB_2)

Pour l'éxecution d'une opérations, les opérandes sont envoyés byte par byte, l'un à la suite de l'autre puis en dernier la commande. Un test par software permet de tester le bits busy du registre d'état, et de voir si le résultat est disponible sur le bu . Ce qui permet de le récuperer et le stocker dans la mémoire M_1.
La conception d'un programme unique permet l'introduction de données en virgule flottante dans la pile est la réception du résultat.
Ce programme s'appellera SAAD; pour une addition simple précision; SMUL : pour une multiplication simple précision et ainsi de suite.

PROGRAMME D'INTRODUCTION

```
        LDX  ≠ $ 4000
LPA     LDA A ( 0,X)
        STA A $ 5000
        INX
        CMPX ≠ $ 4007
        BLE  LPA
        LDA  B ≠  CO (CODE OPERATION)
        STA  B  $ 5001
LP      LDA  A  $ 5001
        AND  A ≠ $ 80
        BNE  LP
        LDX ≠ $ 4010
```

.../...

```
LPB     LDA  A  ø5000
        STA  A (0,X)
        INX
        CMPX ≠ ø 4013
        BLE  LPB
```

```
LPB     LDA  A  #5000
        STA  A (0,X)
        INX
        CMPX ≠ # 4013
        BLE  LPB
```

CHAPITRE IV

FONCTIONNEMENT PAR INTERRUPTION DES 4 UNITES ARITHMETIQUES RAPIDES EN PARALLELES.

A - INTRODUCTION

Les 4 U.A.R peuvent declencher une interruption au C.P.U quand elles ont des données à transmettre à ce dernier.
Pour cela nous avons utilisé un circuit spécialisé le PICU (priority interrupt contrôleur unit), qui récupere toutes les demandes d'interruption vers le C.P.U.
Comme celui-ci (le microprocesseur MC 6800) a un seul niveau d'interruption, le PICU permet d'étendre le nombre de niveaux à huit.

Lorsque l'une ou plusieurs entrées "demandes d'interruption" du contrôleur passent à l'état actif, celui-ci fournit sur ces broches de sortie le code du niveau d'interruption prioritaire.
Le code genére sur 4 bit et qui, associé à d'autres bits d'adresse figée est appelé "vecteur".
Il permet d'obtenir indirectement l'adresse du sous-programme d'interruption prioritaire c.a.d celui relatif à l'U.A.R ayant demandé une interruption et qui est considérée comme prioritaire.

B - TRAITEMENT DES INTERRUPTIONS DANS LE CAS DU 6800

Quand l'une des entrées "demande d'interruption" du UP est activée celui-ci :

- Active le signal "autorisation d'interruption" si les interruptions ont été autorisées par programme.
- Termine l'instruction en cours d'éxécution.
- Inhibe toute nouvelle demande d'interruption en mettant a"1" le masque d'interruption.
- Sauvegarde dans la pile l'adresse de retour au programme ou au sous-programme interrompu par l'arrivée de l'interruption.
- Sauvegarde le contexte machine c.a.d les accumulateurs A et B, les registres d'index, pointeur de pile et l'indicateur.
- Exécute le processus spécifique du UP en vue de se brancher au sous programme d'interruption à éxecuter, en changeant le compteur ordinal avec le contenu des positions-memoires d'adresse FFF8 (octet de poids fort) et FFF9 (octet de poids faible) dans le cas d'une interruption sur l'entrée IRQ.
Si cette demande à lieu sur l'entrée NMI, il change le compteur ordinal avec les positions memoires d'adresse FFFC (octet de poids fort) et FFFD (octet de poids faible).
- Le UP se branche alors à l'adresse indiquée par le contenu du compteur ordinal, adresse à laquelle il doit trouver l'adresse du sous-programme d'interruption.

.../...

C - TRAITEMENT DES INTERRUPTIONS PAR CONTROLEUR

1e PRESENTATION DU 6828 (VOIR BLOC DIAGRAMME INTERNE FIG 4-a)

Le MC6828 est un : circuit spécialisé utilisé pour gerer les demandes d'interruptions simultanées.

Au point de vue de l'utilisateur, il comporte deux registres

- Un registre de demandes qui reçoit les 8 entrées "demandes d'interruption" $\overline{IN0}$ à $\overline{IN7}$, il est appelé request registes par Matorola $\overline{IN7}$ a la plus haute priorité.
- Un registre masque qui inhibe la prise en compte des demandes d'interruptions de niveau inferieur au contenu de ce registre masque de 4 bits.

2e PROGRAMMATION DU 6828

Ce controleur d'interruption ne permet qu'un seul mode de fonctionnement celui-ci ou les priorités sont determinées et fixes, la plus haute ayant le niveau 7 aussi il ne possede aucun registre de commande et aucun registre d'état.

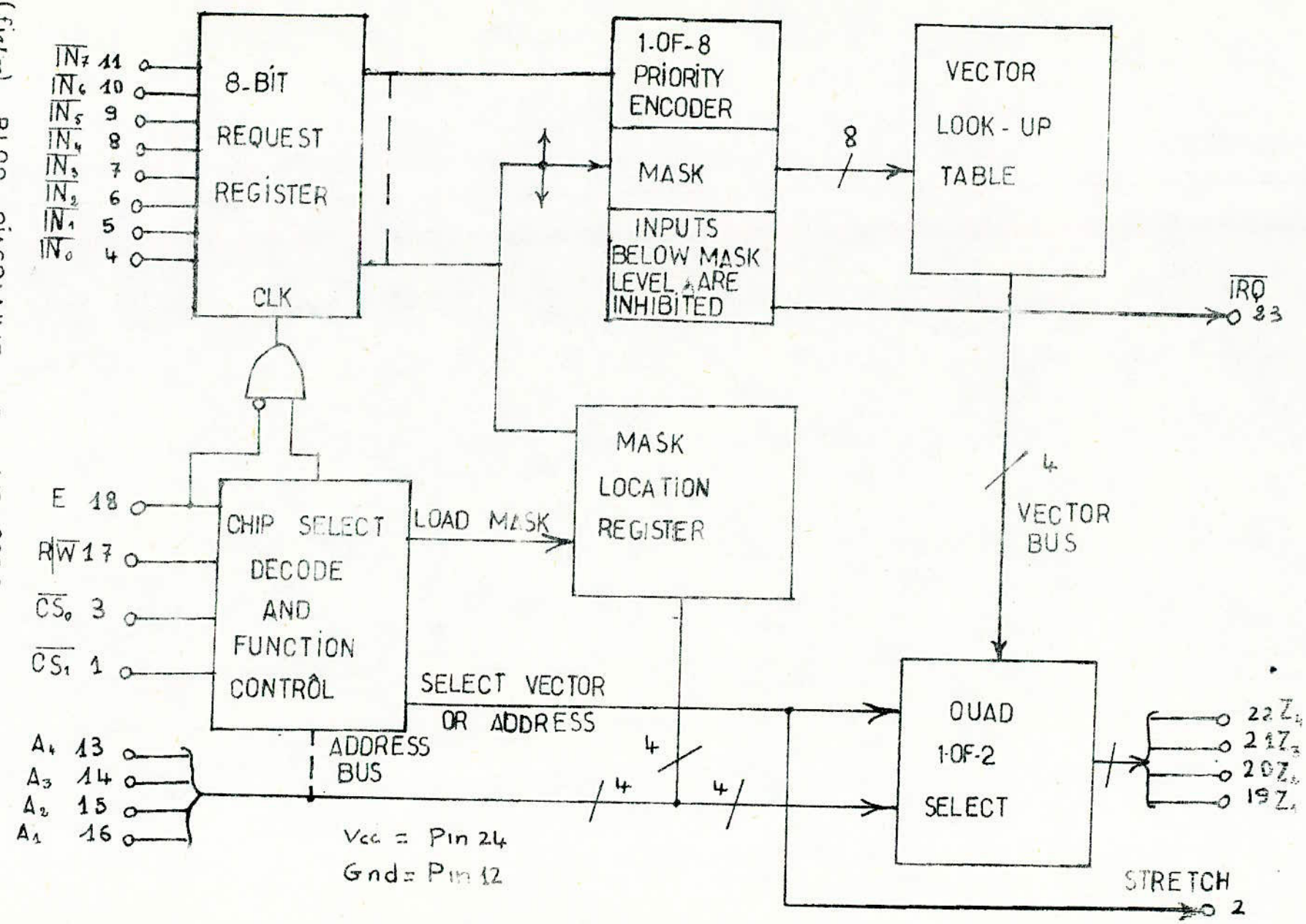
La seule programmation à faire et celle du registre masque comportant 4 bits. L'instruction d'écriture à l'adresse 1111.1111.111X XXX0 charge le registre masque avec le contenu XXXX dont la valeur va de 0000 (aucun niveau d'interruption n'est marqué) à 1000 (tous les niveaux d'interruption sont masqués). Le registre masque inhibe toute demande d'interruption de niveau inferieur au contenu du masque. La programmation de ce registre n'est donc pas faite par le bus donnée qui, pour cette programmation peut avoir n'importe quelle valeur .

3e FONCTIONNEMENT DU 6828

Le controleur d'interruption doit être insere entre les bits d'adresse A_1 A2 A3 A4 du bus adresse et les mêmes bits de la memoireROM nous savons qu'a la suite d'une demande d'interruption le UP lit le contenu des positions-memoires d'adresse FFF8 puis FFF9 pour la première lecture. Il depose la valeur FFF8 sur le bus adresse et lit le contenu du bus de donnée. L'insertion du controleur d'interruption fait que c'est ce dernier qui est adressé par FFF8.

Suite à cet adressage le controleur positionne les 4 bits d'adresse A1 A2 A3 A4 de la ROM avec la valeur d'un vecteur relatif à la demande d'interruption prioritaire. La position memoire lue a donc comme adresse: 1111.1111.111.Z4 Z3 Z2 Z1 0 (A0 = 0)

.../...



(fig4-a) BLOC DIAGRAMME DU MC 6828

Il suffit alors de mettre dans les vecteurs 1111.1111.111 Z4 Z3 Z2 Z1 0 l'octet poids fort (pour $A_0 = 0$) et l'octet poids faible (pour $A_0 = 1$) de l'adresse du sous programme d'interruption à executer

Lorsque le controleur 6828 n'est pas adressé (adresse autre que FFF8 ou FFF9) il laisse passer normalement les bits d'adresse A4 A3 A2 A1. Il jouera le rôle demi selecteur à 8 entrées et 4 sorties

4e SIGNAUX DU 6828

Les broches du 6828 sont :

- Les 8 niveaux de demande d'interruption : $\overline{IN0}$ a $\overline{IN7}$.
- Les 4 bits Z4 Z3 Z2 Z1 du vecteur transmis par le controleur d'interruption à la memoire ROM.
- Le signal R/W reuni à la sortie R/W du UP
- Le signal E reuni à Ø 2
- Les signaus $\overline{CS0}$ et CS1. Le controleur d'interruption doit être validé par FFF8 et FFF9 le bit A_1 A2 A3 A4 sont directement envoyés au controleur par le bus d'adresse. Les autres bits utilisés sont mis en ET et reunis apres inversion à $\overline{CS0}$. Une porte NAND et deux portes AND ont été utilisées pour ROM de capacité 2K octets) prevues le signal CS1 est mis à"1"
- Le signal $\overline{IRQ}$ est la demande d'interruption adressée au UP si une entrée interruption de niveau égal ou supérieur au masque est activée.

Le signal $\overline{STRETCH}$

Le controleur offre la possibilité de prolonger la période d'horloge pour permettre une lecture correcte de la ROM lorsqu'elle a un temps d'accés long.

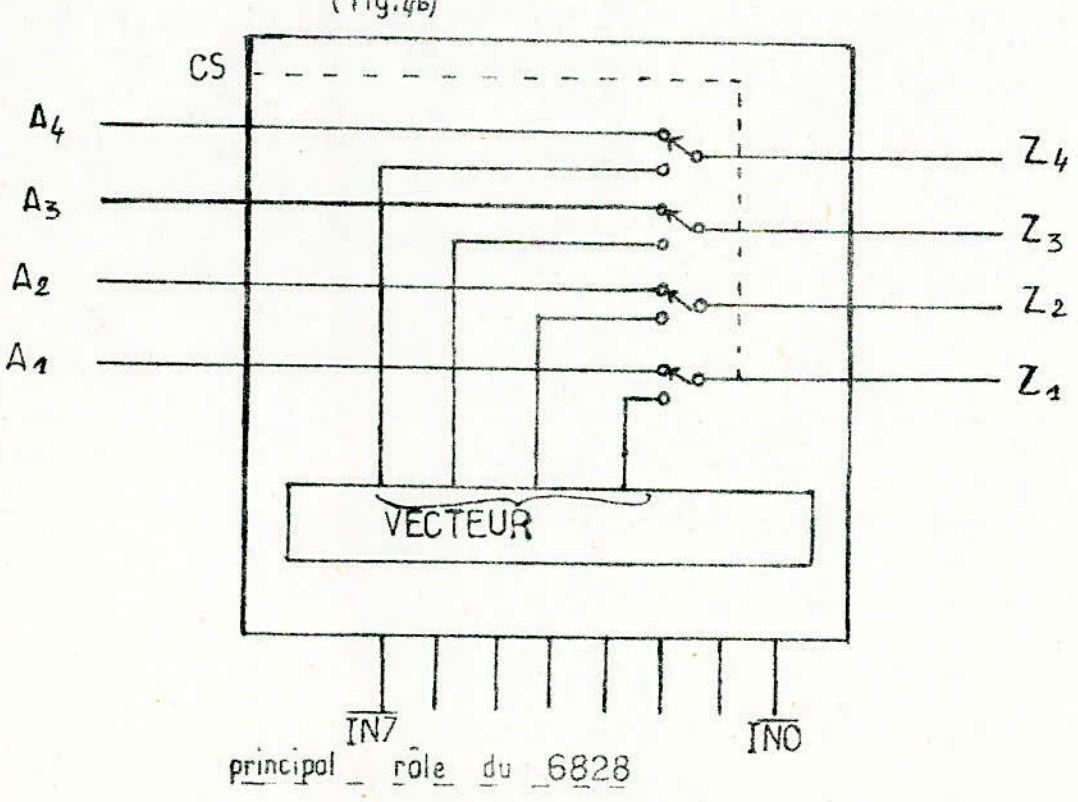
D)- ORGANIGRAMME DE TRAITEMENT D'UNE INTERRUPTION

la seule programmation à faire consiste à (faire) autoriser les interruptions et à programmer le registre masque par les 4 bits d'adresse A4 A3 A2 A1 ceci se fait par une instruction d'écriture , le contenu de l'accumulateur etant indifferent car non utilisé.

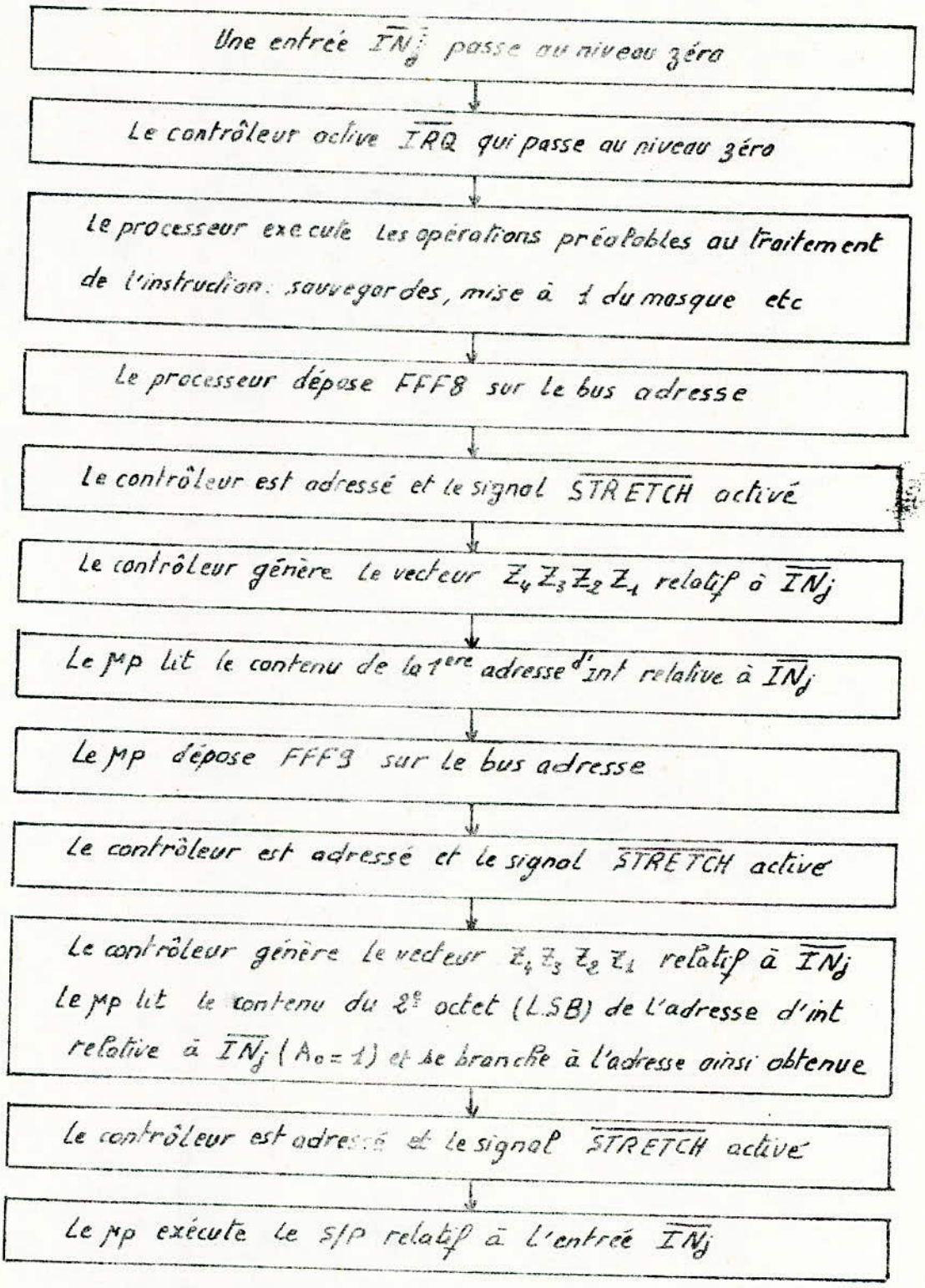
Priorité	Entrée	Vecteur $Z_4Z_3Z_2Z_1$	Adresses d'interruption (en hexadecimal)	Adresses d'interruption contenant l'adresse de :
La plus haute	IN7	1 0 1 1	FFF6 et FFF7	S/P de Priorité 7
	IN6	1 0 1 0	FFF4 et FFF5	S/P " " 6
	IN5	1 0 0 1	FFF2 et FFF3	S/P " " 5
	IN4	1 0 0 0	FFF0 et FFF1	S/P " " 4
	IN3	0 1 1 1	FFEE et FFEF	S/P " " 3
	IN2	0 1 1 0	FFEC et FFED	S/P " " 2
	IN1	0 1 0 1	FFEA et FFEB	S/P " " 1
La plus basse	IN0	0 1 0 0	FFE8 et FFE9	S/P " " 0

Tableau IV. - Adresses d'interruption du contrôleur 6828 de MOTOROLA

(fig.4b)



principal rôle du 6828



(fig.4c) ORGANIGRAMME DE TRAITEMENT D'UNE INTERRUPTION PAR LE 6828

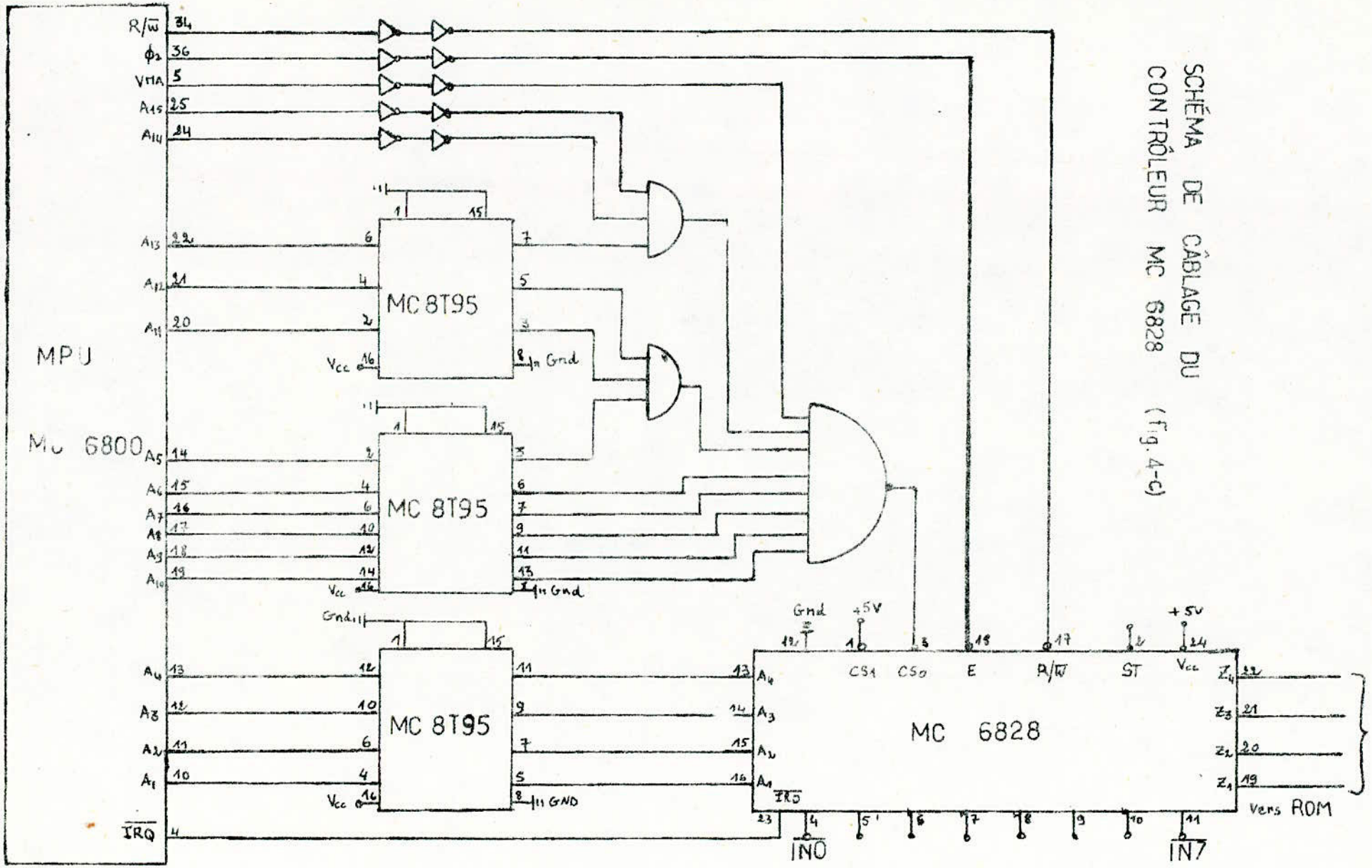


SCHÉMA DE CÂBLAGE DU CONTRÔLEUR MC 6828 (Fig. 4-c)

CHAPITRE IV

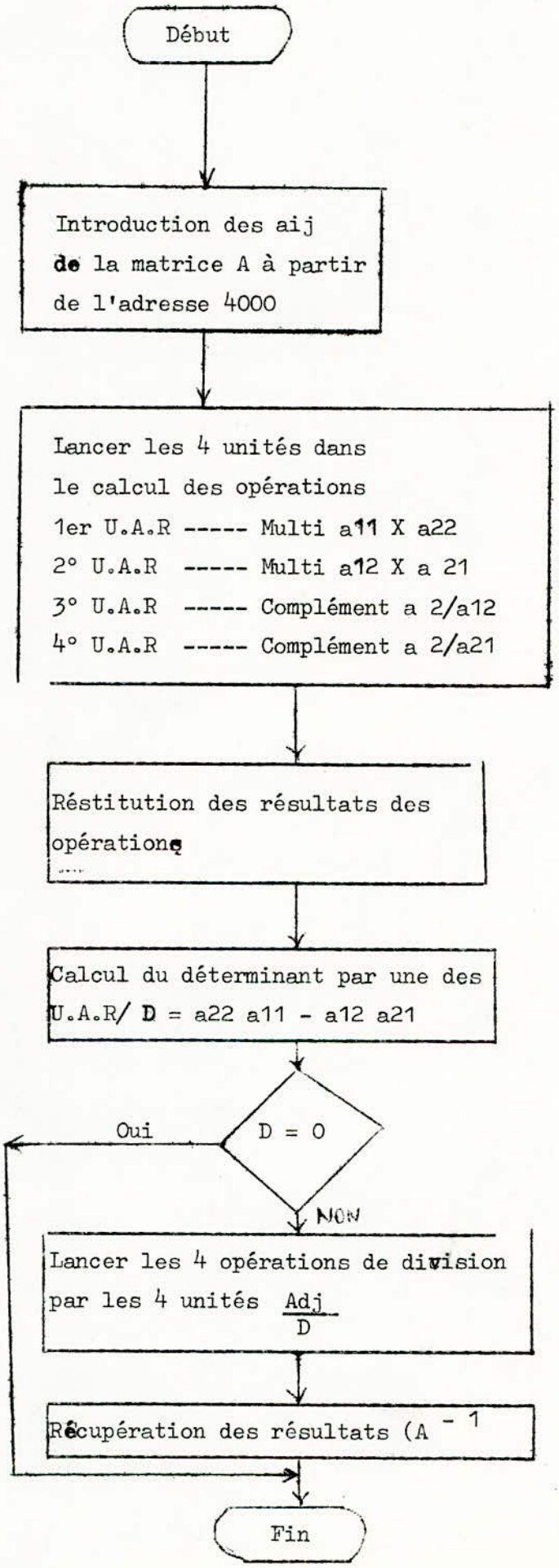
-O- APPLICATION DU MICRO-SYSTEME -O-

INTRODUCTION

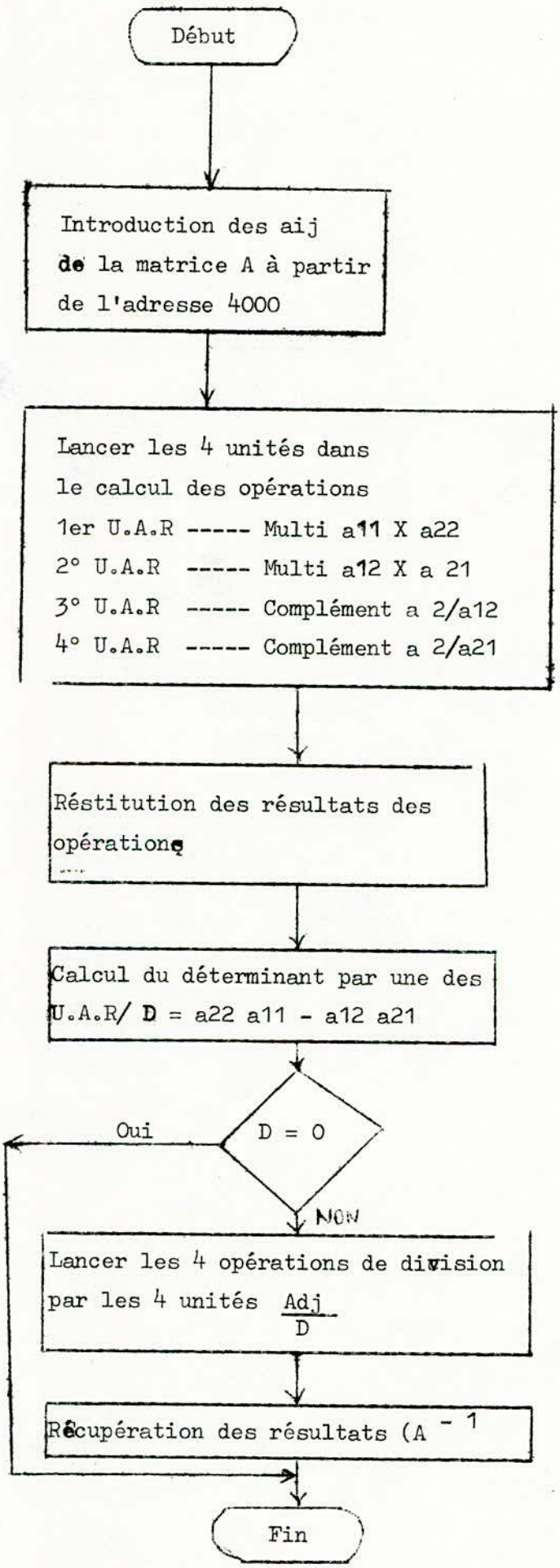
Afin de tester les capacités du micro-système réalisé avec 4 U.A.R en parallèle et la validité de la structure choisie; nous avons opté pourapplication:
L'inversion d'une matice (2x2).
Il va sans dire que l'inversion va utiliser les differentes opérations arithmétiques (addition, soustraction; multiplicatior et division) pour les calculs intermediaires.
Pour cela à chaque fois que le programme principal rencontre une de ces opérations, il fait appel aux U.A.R pour qu'elles se chargent du traitement arithmétique.
Ainsi le problème sera décomposé en tâches élémentaires et indépendantes qui seront éxecutées en parallèle et rapidement par les 4 U.A.R.
La méthode adoptée pour cette inversion, est celle des plus classiques C.A.D :

1°) Calculer le déterminant de la matrice A à inverser.
2°) Tester si ce déterminant est différent de zéro.
3°) Déterminer les cofacteurs, en déduire la comatrice et la matrice adjointe après transposition.
4°) Diviser la matrice adjointe par le déterminant non nul ainsi on obtient la matrice inverse.

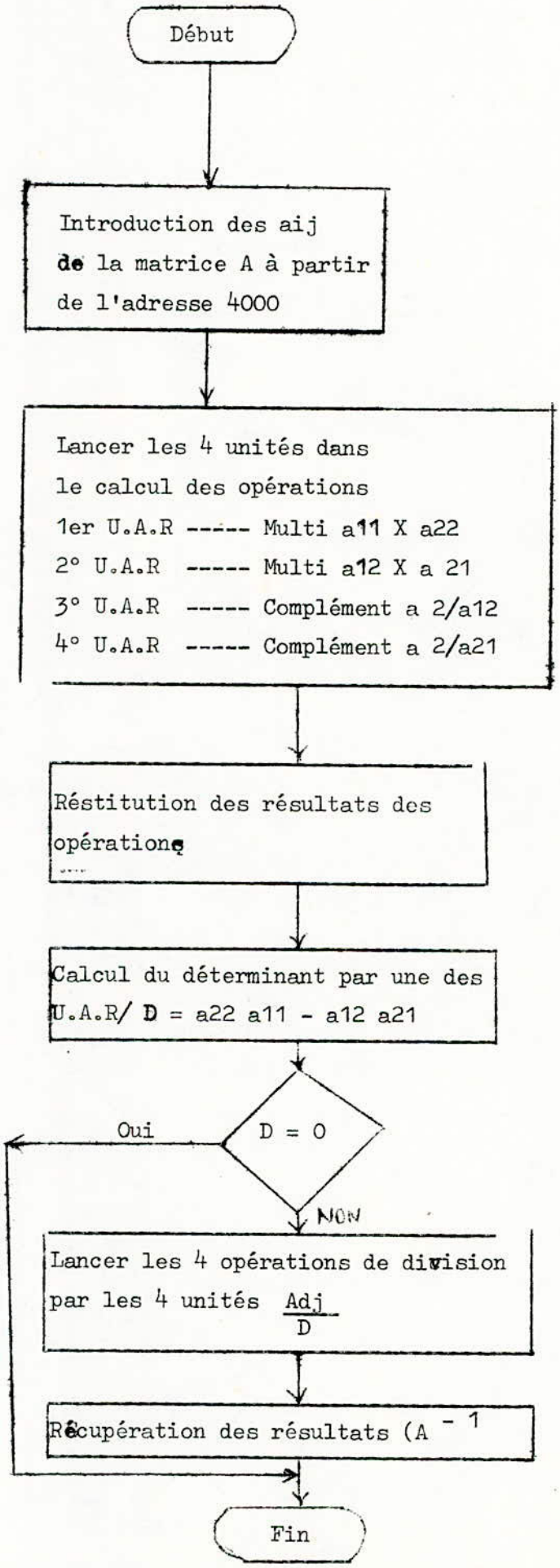
A - ALGORITHME DE LANCEMENT DES OPERATIONS



A - ALGORITHME DE LANCEMENT DES OPERATIONS



A - ALGORITHME DE LANCEMENT DES OPERATIONS



B-PROGRAMME D'INVERSION DE LA MATRICE

Les données sont introduites à partir de l'adresse 4000 jusqu'à l'adresse 400F.

a) Programme principal:

```
LDS #$400F
LDAA # MSB2
PUSHA
LDAA # MSB1
PUSHA
LDAA # LSB2
PUSHA
LDAA # LSB1
..... JUSQU'AU DERNIER OPERANDE
LDS #$4000
LDX #$5000
JSR SPCH
LDS #$400C
JSR SPCH
LDAB #$83
STAB (1,X)
LDAB = 87
STAB (1,X)
LDS #$4004
LDX #$5800
JSR SPCH
JSR SPCH
LDAB #$83
STAB (1,X)
LDX #$6000
LDS #$4004
JSR SPCH
LDAB #$85
STAB (1,X)
LDX #$6800
JSR SPCH
LDAB # $85
STAB (1,X)
LDX #$40AA
STX FFF6
LDX #$40F0
STX $FFF4
LDX #$4136
STX FFF2
LDX #$417C
STX$FFF0
NOP
CLI
Lp STAA$FFE0
   JMP Lp
Lp WAI
   LDAB # $ 3
   DEC B
   BNE Lp
   LDX #$5000
   LDS #$4020
   JSR SPCH
   LDAB # $32
   STAB (1,X)
   LDAB #$37
   STAB (1,X)
   LDX #$4162
   STX #FFF6
   NOP
   CLI
Lp STAA$4000
   JMP Lp
   WAI
   LDAA $4020
   ADDA #$7F
   STAA$4028
   LDAA$4021
   ANDA #80
   STAA$4029
   LDX$4028
   CPX #$7080
   BNE Lp
Lp LDX #$4003
   LDS #$401F
   LDAA (0,X)
   PUSHA
   DEC
   LDX #$400F
   LDS #$4013
   LDAA (0,X)
   PUSHA
   DECX
   LDAA #/$50
   STAA$4030
   LDAA # 00
   STAA$4031
   LDS #4010
   JSR$4400
   LDAA #$58
   STAA$4030
   LDAA # 00
   STAA$4031
   LDS #$4004
   JSR$4400
   LDAA # 60
   STAA$4030
   LDAA # 00
   STAA$4031
   LDS #$4008
   JSR$4400
   LDAA #$80
   STAA$4030
   LDAA # 00
   STAA$4031
   LDS #400C
```

.../...

```
    JSR 4400
    LDX #40AA
    STX FFF6
    LDX #40F0
    STX FFF4
    LDX # 4136
    STX FFF2
    LDX ##417C
    STX FFF0
    NOP
    CLI
Lp  STAA FFE0
    JMP Lp
Lp  WAI
    LDAB # $ 3
    DEC B
    BNE Lp
```

SOUS-PROGRAMMES DE LECTURE DES RESULTATS

SPMUL 1 / sous-programme de multiplication 1

SPMUL2 / " " " " 2

SPSOUST / " " " soustraction

SPCOMP / " " " complémentation

SPDIV / " " " division

SPMUL1 / logé à partir de 40AA

```
LDX # 5000
LDAA (OX)
STAA $ 4020
LDAA (0,X )
STAA $ 4021
LDAA (0,X)
STAA $ 4022
LDAA (0,X)
STAA $ 4023
RTI
```

SPMUL 2/logé à partir de 40F0

```
LDX # $ 5800
LDAA (0,X)
STAA $ 4024
LDAA (0,X)
STAA $ 4025
LDAA (0,X)
STAA $ 4026
LDAA (0,X)
STAA $ 4027
RTI
```

SPSOUST /logé à partir de 4162

```
LDX # $ 5000
LDAA (0,X)
STAA $ 4020
LDAA (0,X)
STAA $ 4021
LDAA (0,X)
STAA $ 4022
LDAA (0,X)
STAA $ 4023
RTI
```

SPCOMP 1 / logé à partir de 4136

```
LDX # $ 6000
LDAA (0,X)
STAA $ 4014
LDAA (0,X)
STAA $ 4015
LDAA (0,X)
STAA $ 4016
LDAA (0,X)
STAA $ 4017
RTI
```

.../...

SPCOMP2 / Logé à partir de 417C

```
LDX ≠ ø 4800
LDAA   (0,X)
STAA   ø 4018
LDAA   (0,X)
STAA   ø 4019
LDAA   (0,X)
STAA   ø 401A
LDAA   (0,X)
STAA   ø 401B
RTI
```

SPDIV 1/ Logé à partir de 40AA

```
LDX ≠ ø 5000
LDAA   (0,X)
STAA   ø 4000
LDAA   (0,X)
STAA   ø 4001
LDAA   (0,X)
STAA   ø 4002
LDAA   (0,X)
STAA   ø 4003
RTI
```

SPDIV2 / Logé à partir de 4070

```
LDX ≠ ø 5800
LDAA   (0,X)
STAA   ø 4004
LDAA   (0,X)
STAA   ø 4005
LDAA   (0,X)
STAA   ø 4006
LDAA   (0,X)
STAA   ø 4007
RTI
```

SPDIV3 / Logé à partir de 4136

```
LDX ≠ ø 6000
LDAA (0,X)
STAA ø 4008
LDAA (0,X)
STAA ø 4009
LDAA (0,X)
STAA ø 400A
LDAA (0,X)
STAA ø 400B
RTI
```

SPDIV4 / Logé à partir de 417C

```
LDX ≠ ø 6800
LDAA (0,X)
STAA ø 400C
LDAA (0,X)
STAA ø 400D
LDAA (0,X)
STAA ø 400E
LDAA (0,X)
STAA ø 400F
RTI
```

SPCH/Sous programme de chargement

```
   CLR B
Lp PULLA
   STAA (0,X)
   INC B
   CMP B ≠ ø 4
   BMI Lp
   RTS
```

SPDIV/ Logé à partir de 4500

```
LDX ø 4030
CLR B
PULLA
STAA (0,X)
INC B
CMP B ≠ ø 4
BMI Lp
LDAA ø 4020
STAA (0,X)
LDAA ø 4021
STAA (0,X)
LDAA ø 4022
STAA (0,X)
LDAA ø 4023
STAA (0,X)
LDA B ≠ ø 84
STAB (1X)
LDA B ≠ ø 87
STAB (1,X)
RTS
```

CONCLUSION

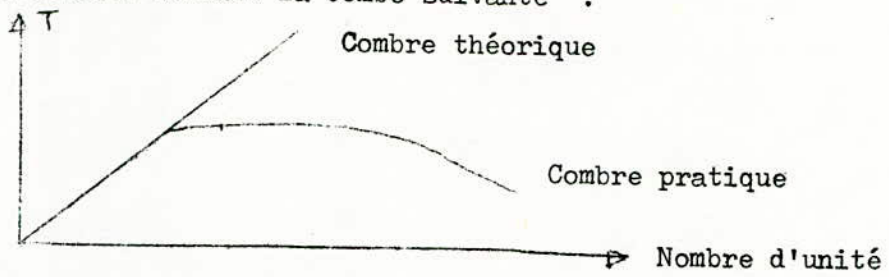
Tout le long de ce projet, notre travail a consisté en l'étude d'un point de vu architecture d'un système multiprocesseur et par la suite la réalisation d'un micro-système avec 4 U.A.R en parallèle.

Au départ nous ne nous sommes pas intéressées directement à la performance d'une telle architecture bien que c'est un des critères de test des capacités d'un système multiprocesseur.

La performance est définie par le temps requis pour éxecuter un certain nombre d'algorithmes donnés et qui est mesurée par le nombre d'opérations par unité de temps.

Idéalement la performance augmente proportionnellement avec le nombre d'unités ajoutées au micro-système, car théoriquement la performance idéale et souhaitée d'un système multiprocesseur est égale à la somme des performances optimums de chaque unité.

Seulement en pratique ce n'est pas le cas, vu que l'éffet de saturation dégrade la performance suivant la combe suivante :



Cette effet peut être attribué au fait que le temps mis par la 1ere U.A.R afin d'éffectuer l'opération arithmétique qui lui est confiée n'est pas supérieur ou égal au temps pris par le lancement des autres opérations aux U.A.R restantes jusqu'à la dernière.

Une fois la dernière instruction éffectuée par la dernière U.A.R.

La 1ere U.A.R doit être disponible sans qu'il yait de temps mort entre le 1er et le dernier lancement des opérations.

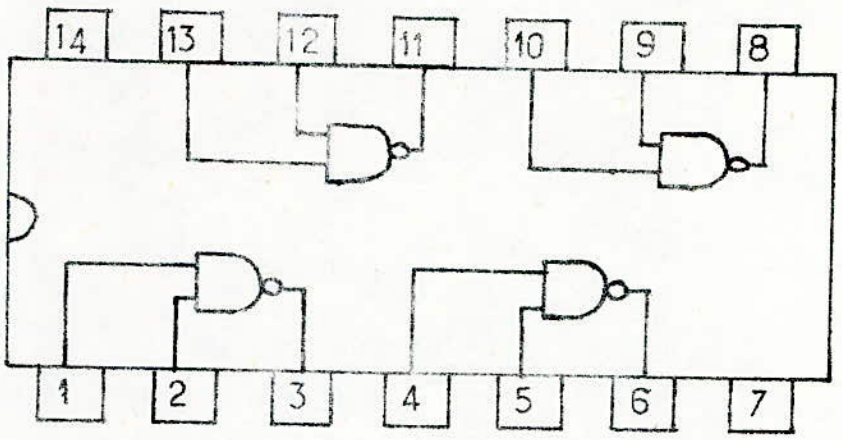
Or nous constatons que ce n'est pas le cas, vu la longueur du programme de lancement.

Nous pouvons y remédier en décomposant le travail principal en petites tâches indépendantes nécessitant de petits lancements, ce qui a été fait dans le c cadre de notre application.

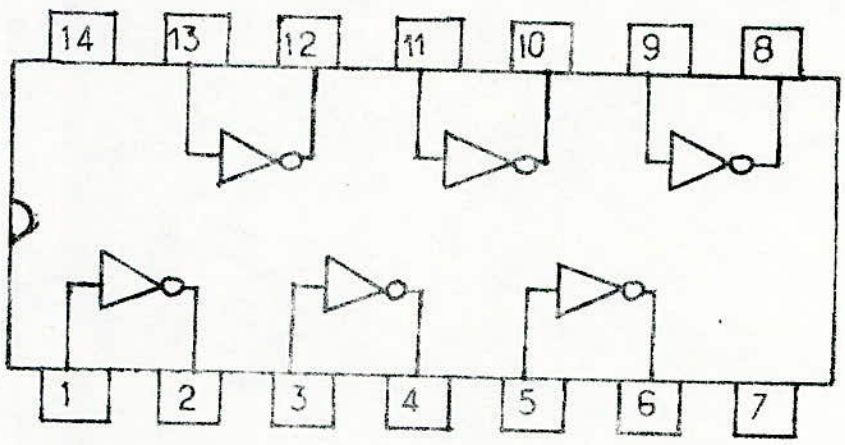
Comme l'avenir appartient aux microprocesseurs 16 bits exemple le 68000, le 8086, le 78000 ..... qui sont plus adaptés pour le calcul numérique, les constructeurs prévoient des processeurs numériques exemple le 8087 d'Intel.

ANNEXE

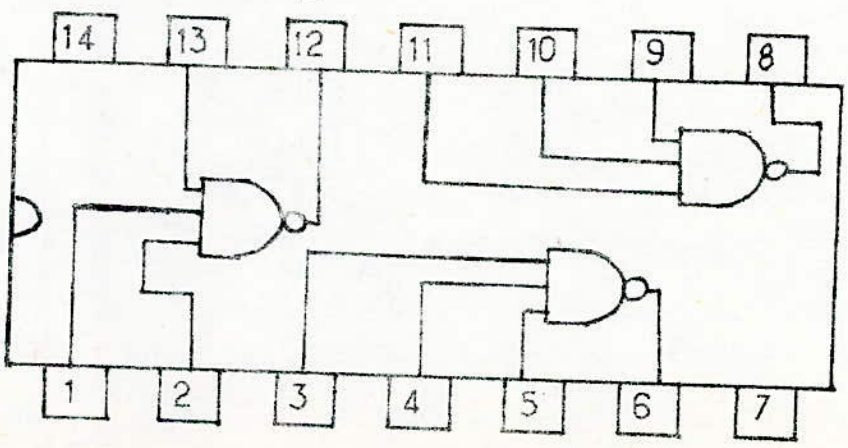
SN74 LS00



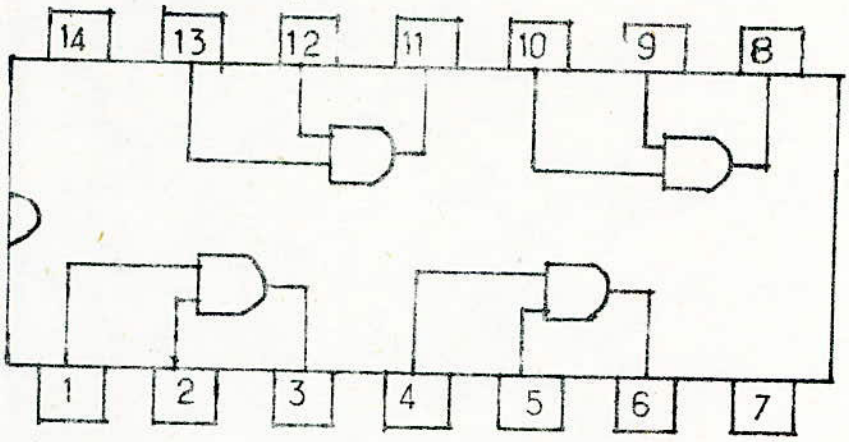
SN74 LS04



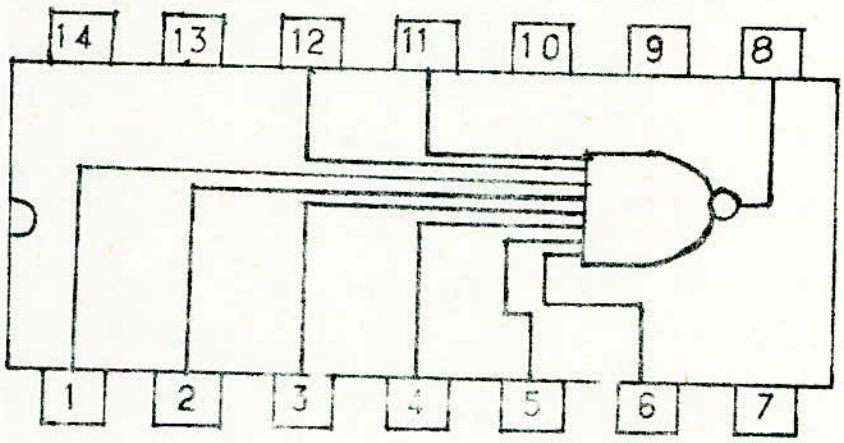
SN74 LS10



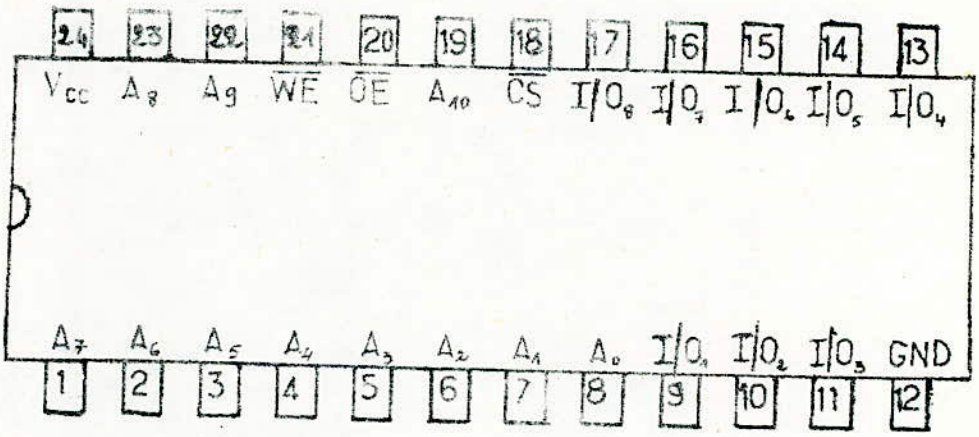
SN 74 LS08



SN 74 LS 30



MC 2016



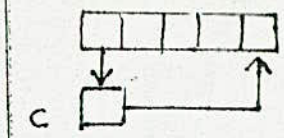
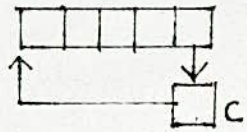
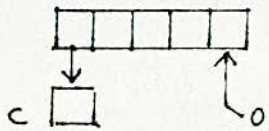
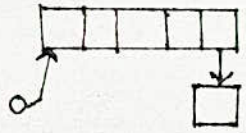
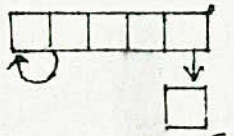
Jeu D'instruction Du Microprocesseur MC 6800

Type d'operations	Mnemonique	Registre concerné	Type d'adressage de la memoire	Fonction realisée	~	#	registres de condition						Code machine
							H	I	N	Z	V	C	
Addition Sans retenue	ADD A		IMM, DIR, ET, INX	A+M → A	2,3,4,5	2,2,3,2	X	·	X	X	X	X	8B, 9B, BB, AB
	ADD B		IMM, DIR, ET, INX	B+M → B	2,3,4,5	2,2,3,2	X	·	X	X	X	X	CB, DB, FB, EB
	ABA	A, B	IMP	A+B → A	2	1	X	·	X	X	X	X	1B
Addition avec retenue	ADC A		IMM, DIR, ET, INX	A+M+C → A	2,3,4,5	2,2,3,2	X	·	X	X	X	X	89, 99, B9, A9
	ADD B		IMM, DIR, ET, INX	B+M+C → B	2,3,4,5	2,2,3,2	X	·	X	X	X	X	C9, D9, F9, E9
Ajustement	DAA				2	1	·	·	X	X	X	X	19
Soustraction Sans retenue	SUB A		IMM, DIR, ET, INX	A − M → A	2,3,4,5	2,2,3,2	·	·	X	X	X	X	80, 90, B0, A0
	SUB B		IMM, DIR, ET, INX	B − M → B	2,3,4,5	2,2,3,2	·	·	X	X	X	X	C0, D0, F0, E0
	SBA	A, B	IMP	A − B → A	2	4	·	·	X	X	X	X	10
Soustraction avec retenue	SBC A		IMM, DIR, ET, INX	A − M − C → A	2,3,4,5	2,2,3,2	·	·	X	X	X	X	82, 92, B2, A2
	SBC B		IMM, DIR, ET, INX	B − M − C → B	2,3,4,5	2,2,3,2	·	·	X	X	X	X	C2, D2, F2, E2
Incrementer	INCA	A		A+1 → A	2	1	·	·	X	X	·	·	4C
	INCB	B		B+1 → B	2	1	·	·	X	X	·	·	5C
	INX	X		X+1 → X	4	1	·	·	·	X	·	·	08
	INS	SP		SP+1 → SP	4	1	·	·	·	·	·	·	31
	INC		INX; ET	M+1 → M	7.6	2.3	·	·	X	X	·	·	6C, 7C
Decrementer	DECA	A		A − 1 → A	2	1	·	·	X	X	·	·	4A
	DECB	B		B − 1 → B	2	1	·	·	X	X	·	·	5A
	DEX	X		X − 1 → X	4	1	·	·	·	X	·	·	09
	DES	SP		SP − 1 → SP	4	1	·	·	·	·	·	·	34
	DEC		INX; ET	M − 1 → M	7.6	2.3	·	·	X	X	·	·	6A, 7A

TYPE D'OPERATIONS	MNEMONIQUE	REGISTRE CONCERNE	TYPE D'ADRESSAGE	FONCTION REALISEE	~	#	REGISTRE DE CONDITIONS						CODE OPERATION
							H	I	N	Z	V	C	
REMISE à ZERO	CLR		ET, INX	$00 \rightarrow M$	6,7	3,2	.	.	R	X	R	R	7F, 6F
	CLRA	A	Implicite	$00 \rightarrow A$	2	1	.	.	R	X	R	R	4F
	CLR B	B	"	$00 \rightarrow B$	2	1	.	.	R	X	R	R	5F
Complement à 1	COM		ET, INX	$\overline{M} \rightarrow M$	6,7	3,2	.	.	X	X	R	X	73, 63
	COM A	A		$\overline{A} \rightarrow A$	2	1	.	.	X	X	R	X	43
	COM B	B		$\overline{B} \rightarrow B$	2	1	.	.	X	X	R	X	53
Mise à 1 de la retenue	SEC		IMP	$1 \rightarrow C$	2	1	.	.	.	.	.	.	0D
Complement à 2	NEG		ET, INX	$00 \rightarrow M \rightarrow M$	6,7	3,2	.	.	X	X	X	X	70, 60
	NEG A	A		$00 \rightarrow A \rightarrow A$	2	1	.	.	X	X	X	X	40
	NEG B	B		$00 \rightarrow B \rightarrow B$	2	1	.	.	X	X	X	X	50
Mise à 0 du Bit de retenue	CLC		IMP	$0 \rightarrow C$	2	1	.	.	.	.	.	.	0C
Mise à 0 du masque d'interrup	CLI		IMP	$0 \rightarrow I$	2	1	.	.	.	.	.	.	0E
	CLV		IMP	$0 \rightarrow V$	2	1	.	.	.	.	.	.	0A

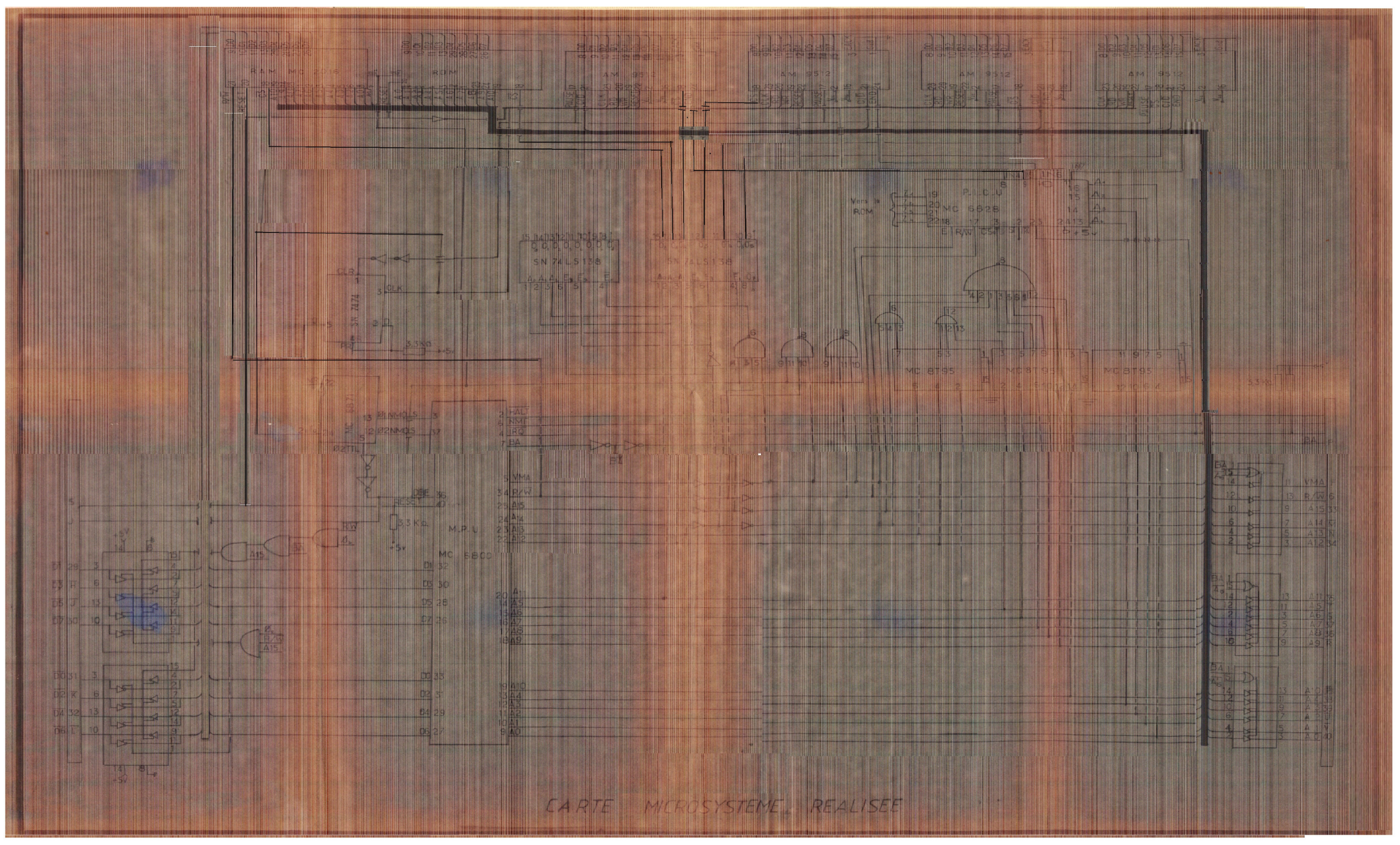
Type d'operations	Mnémonique	Registre concerné	Type d'adressage de la mémoire	Fonction réalisée	~	≠	Registre de conditions						code machine
							H	I	N	Z	V	C	
AND [ET]	AND A	A, B	IMM, DIR, ET, INX	A · M → A	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$84, 94, B4, A4$
	AND B		IMM, DIR, ET, INX	B · M → A	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$C4, D4, F4, E4$
OU [OR]	ORA A	A, B	IMM, DIR, ET, INX	A + M → A	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$8A, 9A, BA, AA$
	ORA B		IMM, DIR, ET, INX	B + M → B	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	CA, DA, FA, EA
OU EXCL [EXCL OR]	EOR A	A, B	IMM, DIR, ET, INX	A ⊕ M → A	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$88, 98, B8, A8$
	EOR B		IMM, DIR, ET, INX	B ⊕ M → B	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$C8, D8, F8, E8$
Par masque TEST DE BIT	BIT A	A, B	IMM, DIR, ET INX	A . M	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R*	•	$85, 95, B5, A5$
	BIT B		IMM, DIR et INX	B . M	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	R	•	$C5, D5, F5, E5$
Avec mémoire Comparaison	CMP A	A, B	IMM, DIR ET INX	A M	2, 3, 4, 5	2, 2, 3, 2	•	•	X	X	X	X	$81, 91, B1, A1$
	CMP B		IMM, DIR ET INX	B M	2, 3, 4, 5	2, 2, 3, 2			X	X	X	X	$C1, D1, F1, E1$
	CPX	X	IMM, DIR ET INX	X M (M sur deux bytes)	3, 4, 5, 6	3, 2, 3, 2	•	•	X	X	X	•	$8C, 9C, AC, BC$
Avec Registre Comparaison	CBA	A, B		A, B	2	1	•	•	X	X	X	X	11
Avec zero TES	TST		ET, INX	M - 00	6, 7	3, 2	•	•	X	X	R	R	7D, 6D
	TST A	A		A - 00	2	1	•	•	X	X	R	R	4D
	TST B	B		B - 00	2	1	•	•	X	X	R	R	5D

TYPE D'opération	MNÉMONIQUE	Registres concernés	Type d'Adressage	Fonction realisée	~	#	Registre de conditions						code machine
							H	I	N	Z	V	C	
changement des registres à partir de la memoire	LDAA LDAB LDX LDSP	A B X P	IMM, DIR, ET, INX IMM, DIR, ET, INX IMM, DIR, ET, INX IMM, DIR, ET, INX	M → A M → B M → X M → SP	2, 3, 4, 5 2, 3, 4, 5 3, 4, 5, 6 3, 4, 5, 6	2, 2, 3, 2 2, 2, 3, 2 3, 2, 3, 2 3, 2, 3, 2	.	.	X	X	R	.	86, 96, B6, A6 C6, D6, F6, E6 8A, 9A, BA, AA CA, DA, FA, EA
transfert du contenu des registres vers la mémoire	STAA STAB STX STS	A B X SP	DIR, ET, INX DIR, ET, INX DIR, ET, INX DIR, ET, INX	A → M B → M X → M SP → M	4, 5, 6 4, 5, 6 5, 6, 7 5, 6, 7	2, 3, 2 2, 3, 2 2, 3, 2 2, 3, 2	.	.	X	X	R	.	97, B7, A7 D7, F7, E7 DF, FF, EF 9F, BF, AF
transfert de registre à registre	TAB TBA TXS TSX TAP TPA	A, B A, B X, SP X, SP A, registre cond. A, registre cond.	IMP " " " " "	A → B B → A X − 1 → SP SP + 1 → X CCR → A A → CCR	2 2 4 4 2 2	1 1 1 1 1 1	.	.	X	X	R	.	16 17 35 30 06 07
transfert d'un contenu d'un registre dans la pile	PSHA PSHB	A B	IND	A → Msp B → Msp	4 4	1 1	.	.	.	.	.	.	36 37
Transfert de la pile vers un registre	PULA PULB	A B	IND	Msp → A Msp → B	4 4	1 1	.	.	.	.	.	.	32 33
Mise à un du masque	SEI		IMP	1 → I	2	1	.	1	.	.	.	.	0F

TYPES D'OPERATIONS	MNEMONIQUE	REGISTRE CONCERNÉ	TYPE D'ADRESSAGE	FONCTION REALISEE	~	#	Registre de conditions H	I	N	Z	V	C	code machine
Rotation à gauche sur 9 bits	ROL		ET, INX		6,7	3,2	°	°	X	X	X	X	79, 69
	ROLA	A			2	1	.	.	X	X	X	X	49
	ROL B	B			2	1	.	.	X	X	X	X	59
Rotation à DROITE SUR 9 bits	ROR		ET, INX		6,7	3.2	.	.	X	X	X	X	76, 66
	ROR A	A			2	1	.	.	X	X	X	X	46
	ROR B	B			2	1	.	.	X	X	X	X	56
DECALAGE à gauche sur 8 bits avec insertion du zero arithmetique	ASL		ET, INX		6,7	3.2	,	°	X	X	X	X	78, 68
	ASL A	A			2	1	.	.	X	X	X	X	48
	ASL B	B			2	1	°	.	X	X	X	X	58
DECALAGE à DROITE SUR 8 bits avec insertion d'1 zero	LSR		ET, INX		6,7	3.2	°	°	R	X	X	X	74, 64
	LSR A	A			2	1	.	.	R	X	X	X	44
	LSR B	B			2	1	.	.	R	X	X	X	54
Decalage à gauche sur 8 bits avec conservation du dernier bit arithmetique	ASR		Et, INX		6,7	3,2	.	°	X	X	X	.	77, 67
	ASR A	A			2	1	°	.	X	X	X	.	47
	ASR B	B			2	1	°	.	X	X	X	.	57

B I B L I O G R A P H I E

- Les systèmes à UP)
- l'emploi des UP) M. AUMIAUX Edition masson.
- Rendez-vous avec le UP Phanson et P. Bellier
- Microprocesseur SF.F 96800)
  et circuits associés 1978) Thomson - CSF
- Am. 9512 Flocting point processeur
  Advanced Micro devices
  Advances MOS / LSI
- Les super ordinateurs par RONALD LEVINE (MARS 1982)
- Bibliothèque mathématique fournie par MOTOROLA.
- Data book Motorola
- Data book TTL.


RAM
ROM
AM 9112
AM 9112
AM 9112
AM 9112
P.I.C.U
MC 6828
SN 74LS138
SN 74LS138
MC 8T95
MC 8T95
MC 8T95
M.P.U
MC 6800
3.3 KΩ
+5V
CARTE MICROSYSTEME REALISEE