UNIVERSITE DE GRENOBLE

MATHEMATIQUES APPLIQUEES

Anné 1959 - 60

PROGRAMMATION GAMMA - E T

2ème Partie :

Langage machine

L.BOLLIET

C H A P I T R E 1

L'emploi de la virgule flottante nous a permis d'exposer rapidement un mode simple de programmation. Il donne, par contre, une idée assez fausse du calculateur, comme nous allons le voir.

ECRITURE DES NOMBRES EN VIRGULE FIXE.- "SPLITTAGE" DES MEMOIRES.-

Les mémoires dont on dispose ont toujours 12 positions, mais on peut y disposer à son gré, plusieurs nombres. Il importe de bien remarquer que cette manière de faire laisse toute liberté au programmeur pour placer les nombres. Elle l'oblige, en revanche, à noter et à indiquer à la machine, non seulement la mémoire où se trouve le nombre, mais encore les positions de mémoire qu'il occupe ou mieux les limites entre lesquelles il est écrit.

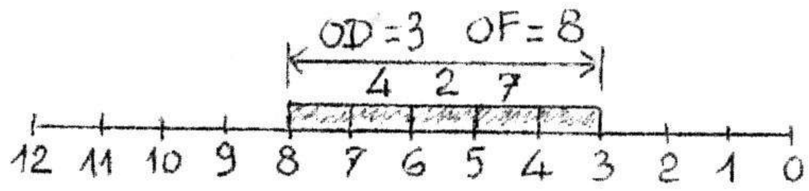
ORDRE DEBUT (OD) et ORDRE FIN (OF).-

Les codes OD et OF permettent de repérer les limites d'écriture pour chaque nombre.

Ces codes sont enregistrés en même temps que les codes TO et AD dans les 4 mémoires-programmes (TO-AD-OD-OF) à chaque ligne d'instruction.

Remarquons que OD et OF n'indiquent pas forcément la position des chiffres significatifs extrêmes, mais les positions de mémoire effectivement réservées au nombre, même si celui-ci ne les utilise pas toutes.

Exemple :



On a placé entre OD=3 et OF=8, le nombre 427 qui n'occupe que 3 positions sur les 5.

On définit de cette façon une portion de mémoire donc plus qu'un nombre.

Le calculateur ne connait pas la position de la virgule le programmeur doit donc noter celle-ci et s'arranger pour que les nombres se combinent correctement.

SIGNE DES NOMBRES.-

Cette notion de signe n'a de sens que si le calculateur est en CD.

En mémoire opérateur : Lorsqu'un nombre est appelé en mémoire opérateur

- son signe est envoyé dans une mémoire spéciale, la MS1 et sa valeur absolue est envoyée en M1

Remarque : En CD, la M1 ne peut contenir de code $\geqslant 10$

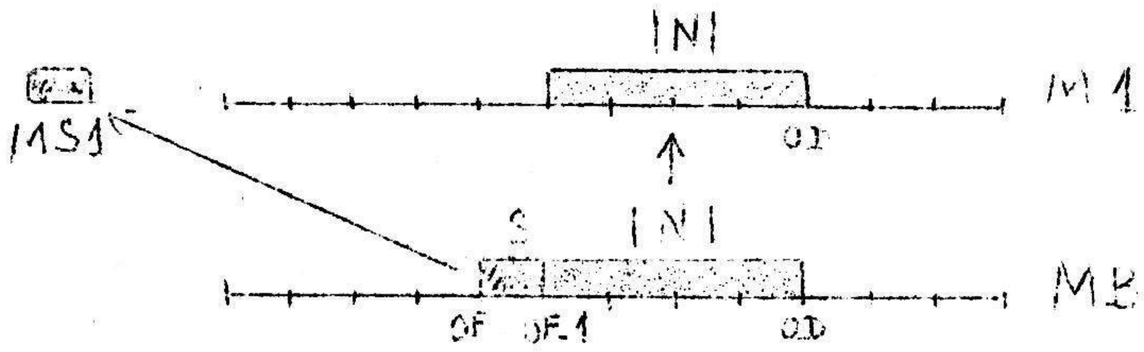
- en mémoire banale:

Les nombres positifs sont écrits sans signe.

Les nombres négatifs sont caractérisés par un code 10(8+2) en position OF - 1. Il en résulte que pour loger un nombre algébrique il faut prévoir une position de mémoire pour le signe en tête du nombre en plus de celles nécessaires pour les chiffres.

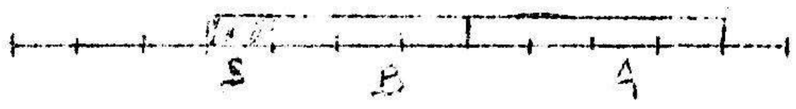
Il en résulte également qu'il ne faut pas modifier l'OF d'un nombre négatif sous peine de perdre son signe.

Exemple :



Dans le cas d'un signe plus, il n'est pas nécessaire de laisser en blanc la position définie par OF - 1; OF ; elle peut être occupée par n'importe quel code arithmétique (inférieur à 10).

Exemple :



Si A est $>$ 0, on peut inscrire à sa gauche, sans laisser de blanc, un nombre B.

MEMOIRE OPERATEUR. MEMOIRE BANALE.

La mémoire 1 joue le rôle d'organe de calcul ; elle est appelée mémoire opérateur.

Toutes les autres mémoires servent uniquement à enregistrer des données et des résultats ; elles sont appelées mémoires banales.

La mémoire 1 possède toutes les propriétés des mémoires banales et dans certaines opérations, on peut l'utiliser comme mémoire banale.

La mémoire 2 dépourvue de la fonction calcul est annexée dans certaines opérations à la mémoire 1, dont elle double ainsi la capacité.

La mémoire opérateur assure la fonction calcul grâce à un additionneur-soustracteur qui constitue sa douzième position.

L'additionneur-soustracteur comporte deux entrées sur lesquelles sont amenés les deux termes, et une sortie qui émet la somme ou la différence. L'une de ses entrées est alimentée en permanence par la sortie de la mémoire opérateur, et la sortie est systématiquement connectée à l'entrée de cette mémoire.

Il en résulte que, dans toute addition ou soustraction , l'un des facteurs doit être préalablement enregistré dans la mémoire opérateur et que le résultat se forme toujours dans la mémoire opérateur à la place du facteur qui s'y trouvait.

La deuxième entrée de l'additionneur-soustracteur est alimentée par un circuit particulier dit "canal-données". Il peut recevoir la sortie de l'une quelconque des mémoires 2 à 7 et des mémoires 8 à 15 de l'octade commutée, et transmettre son contenu pour l'ajouter, ou le soustraire, à celui de la mémoire opérateur.

Un circuit symétrique dit "canal-résultats" est alimenté par la sortie de la mémoire opérateur et permet de transférer son contenu dans l'une quelconque des mémoires précédentes.

OPERATEUR DE SIGNES. -

C'est un dispositif permettant de déterminer le signe du résultat pour les 4 opérations élémentaires.

L'opérateur de signe est constitué :

- d'une mémoire signe MS 1 qui contient à tout instant le signe du nombre situé en mémoire opérateur,

- d'une mémoire signe MS B qui reçoit le signe du second terme utilisé au cours de l'opération,

- d'un opérateur de signes proprement dit qui détermine le signe du résultat et l'envoie dans MS 1.

En addition et en soustraction, le signe du deuxième facteur transmis par le canal-donnée est comparé au signe du premier facteur contenu en mémoire opérateur. Cette comparaison, en cas de différence, inverse la commande de l'additionneur-soustracteur. On a ainsi :

$$(a) + (b) = a + b \qquad (a) + (-b) = a - b$$
$$(-a) + (-b) = -(a+b) \qquad (a) - (-b) = a + b$$

Le signe du résultat est déterminé par les signes des facteurs et par un comparateur qui détermine si le résultat apparait sous forme normale ou sous forme complémentaire.

En multiplication et en division, le signe du résultat est déterminé par la règle des signes.

En division, le quotient et le reste ont le même signe.

MEMOIRE DECALAGE. -

La mémoire décalage est un organe qui permet de repérer dans la mémoire **opérateur** une position de cette mémoire.

Dans la plupart des cas, son contenu est égal à l'OD de la portion de mémoire utilisée dans l'opérateur.

Son fonctionnement et son utilisation seront étudiés plus tard.

En résumé, l'opérateur peut se schématiser ainsi :

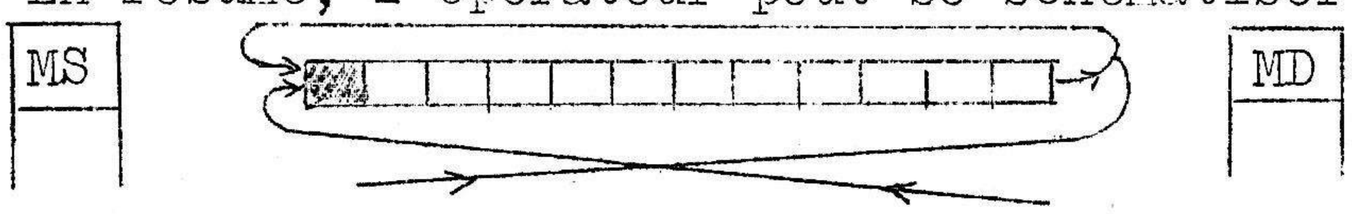
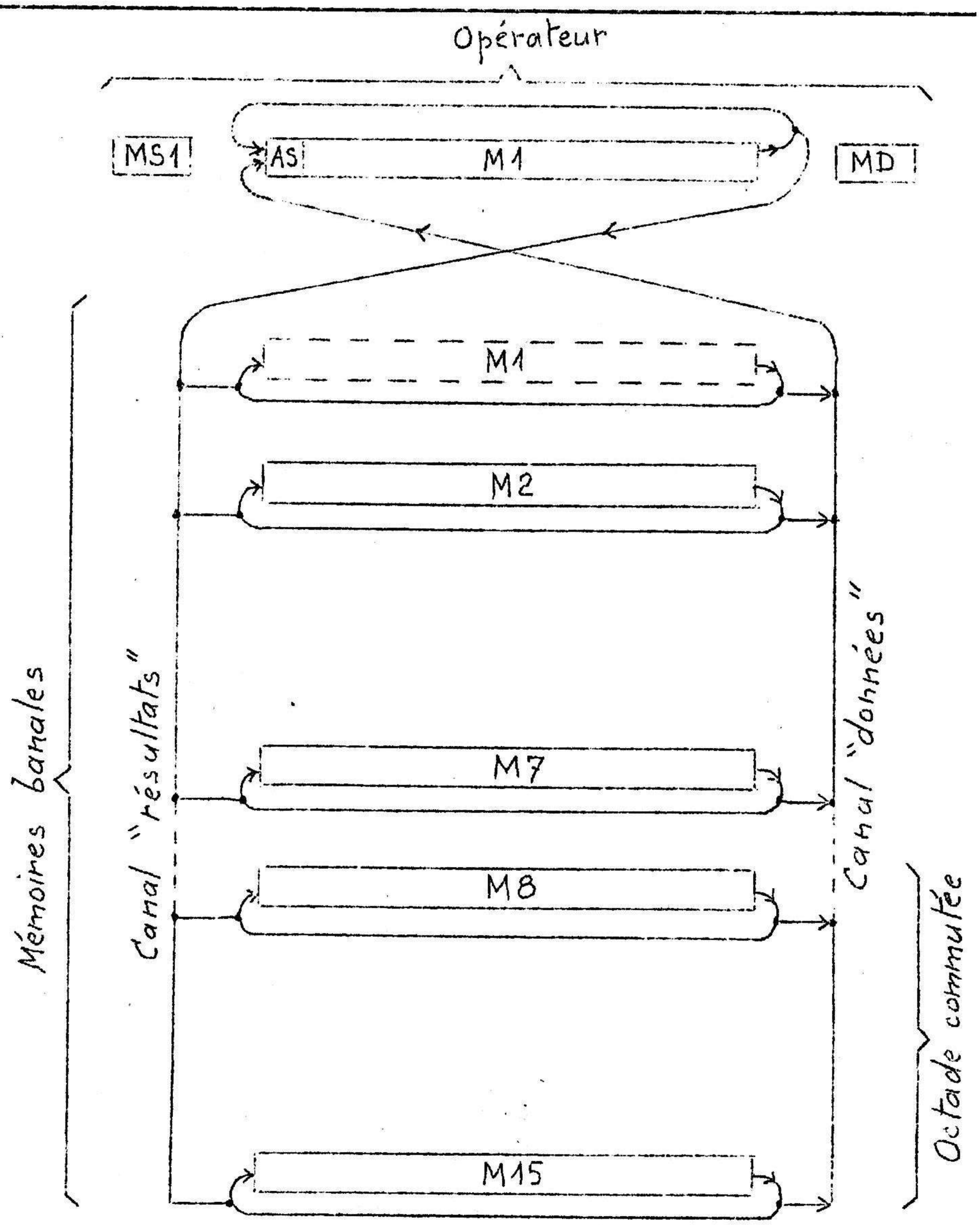


Schéma de la mémoire opérateur et des mémoires banales.-



La M 1 est figurée deux fois pour bien montrer son double rôle : opérateur et banale.

En effet, le contenu de la mémoire opérateur peut être considéré comme donnée de calcul ou comme résultat de calcul à renvoyer dans la mémoire 1.

Dans les opérations où la mémoire opérateur intervient comme mémoire banale, l'AD est égale à 1 .

FILTRAGE.-

Le filtrage est l'organe qui permet de splitter les mémoires position par position.

Le filtrage a trois actions :

- il valide le canal-donnée uniquement pendant le passage des positions décimales comprises entre OD et OF et interdit le transfert dans la mémoire opérateur des chiffres éventuellement contenus dans la mémoire banale, à droite et à gauche du nombre filtré.
- il valide le canal-résultat dans les mêmes conditions et interdit le transfert dans une mémoire banale des chiffres éventuellement contenus dans la mémoire opérateur à droite et à gauche du nombre filtré.
- il conditionne l'effacement des mémoires banales afin de n'effacer que la zone définie par le filtrage.

Remarque : Si dans une opération on affiche un OD $\geqslant$ OF , l'OF est pris égal à $\underline{0}$ (12)

C H A P I T R E II

LES OPERATIONS ELEMENTAIRES

I.- OPERATIONS DE TRANSFERT. -

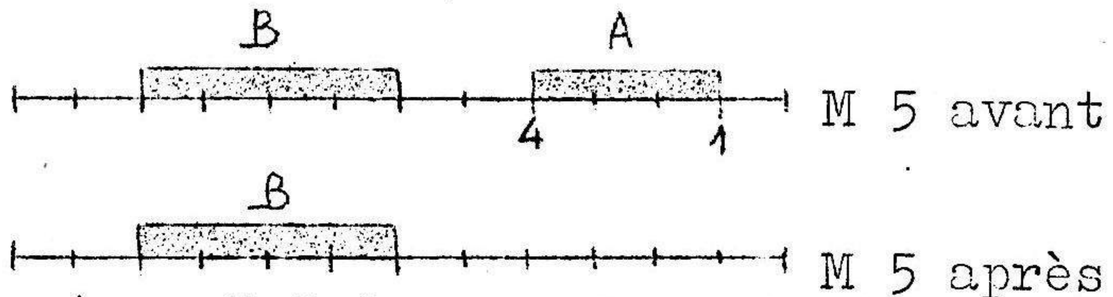
REMISE A ZERO D'UNE MEMOIRE BANALE ZB. TO = 3.

Cette opération permet d'effacer totalement ou partiellement une mémoire banale.

L'adresse définit la mémoire à effacer.

L'OD et l'OF définissent les limites des positions à effacer en cas d'effacement partiel.

Exemple :



Soient en M 5 deux nombres A et B. On veut effacer A.

TO	AD	OD	OF
3	5	1	4

ZB avec AD = 1.

La M 1 est utilisée comme mémoire banale.

La MD et la MS ne sont pas altérées.

La mémoire opérateur n'est donc pas totalement effacée par cette opération.

ZB avec AD = 0.

Opération dépourvue de sens

INTRODUCTION DE CONSTANTES EN MEMOIRE BANALE KB. TO = 4.-

Cette opération a déjà été étudiée en virgule flottante.

Elle permet d'introduire dans une mémoire définie par l'AD un code égal à l'OF (compris entre 0 et 15 même en CD s'il s'agit d'une mémoire banale) dans la position OD avec effacement préalable de cette seule position.

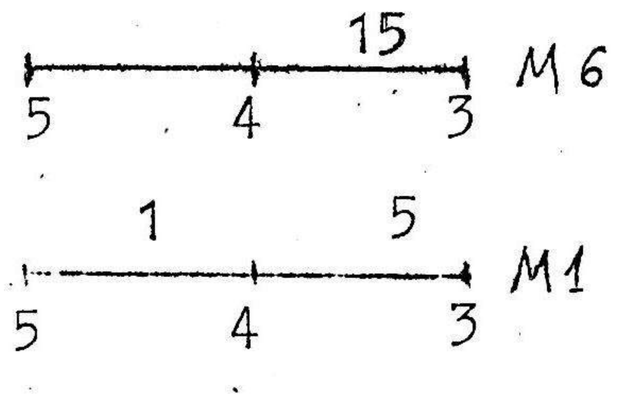
KB avec AD = 1. Introduction de constantes en M 1

Il faut distinguer deux cas :
en calcul binaire, aucune restriction sur les propriétés données par les KB,
en calcul décimal, on ne peut introduire que des codes ≤ 9, sur une seule position à cause de l'additionneur-soustracteur.

Si le code est > 9, il y a émission du report sur la position suivante.

Exemple: KB de 15

	TO	AD	OD	OF
	4	6	3	15
Opérateur en CD	4	1	3	15



Le passage du code 15 dans l'additionneur-soustracteur (12ème position) le transforme en 15 écrit sur 2 positions.

La MS et MD ne sont pas altérées.

Remarque:

KB avec AD = 0

Cette opération est dépourvue de sens pour le calculateur.
(Emission de 48 V vers la machine connectée, 3ème partie page 9.

TRANSFERT D'UNE MEMOIRE BANALE EN MEMOIRE OPERATEUR BO.TO = 6.-

L'opération BO permet de transférer dans la mémoire opérateur un nombre contenu en mémoire banale.

L'AD désigne la MB intéressée.

L'OD et l'OF définissent la portion de mémoire banale contenant le nombre à transférer.

1°) l'opérateur est intégralement effacé (M 1, MS 1, MD)

2°) le terme défini par le filtrage OD-OF prend place dans M 1 avec le même filtrage,

3°) l'OD du nombre est transféré en MD,

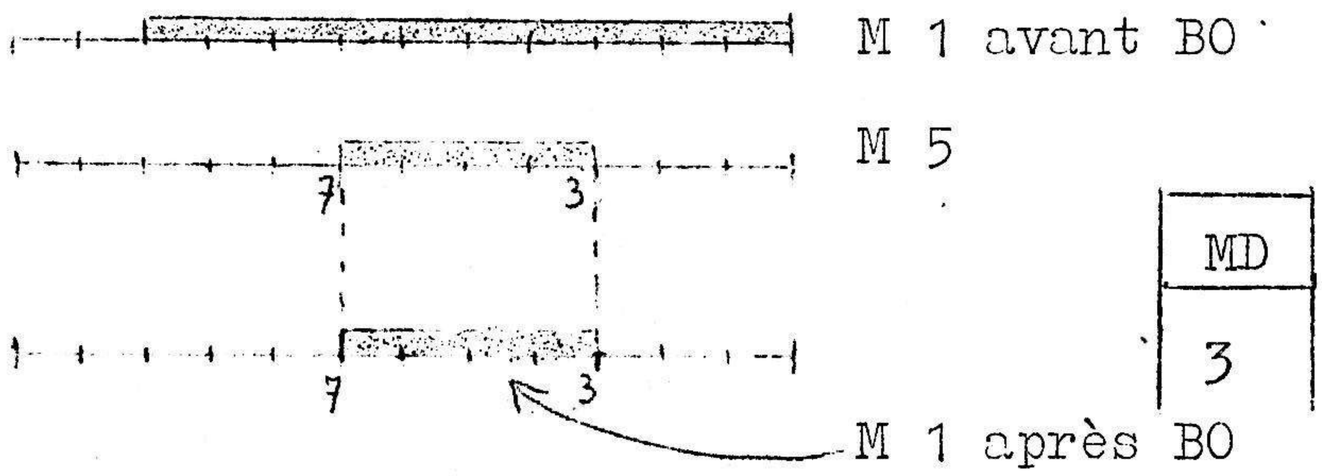
4°) le signe est transféré en MS 1 .

Remarques:- Il n'est pas possible, par cette opération, d'amener un nombre en M 1 dans des positions autres que celles qu'il occupe en MB.

Cette remarque donne la clé de la multiplication et de la division.

- Il n'est pas possible d'introduire séparement deux nombres en M 1 par deux BO successives.

Exemple : Transfert de M 5 en M 1.



TO	AD	OD	OF
6	5	3	7

BO avec AD = 1.

La mémoire émettrice est la M 1 considérée comme mémoire banale.

La mémoire réceptrice est la M 1 considérée comme mémoire opérateur.

Cette opération a pour effet :

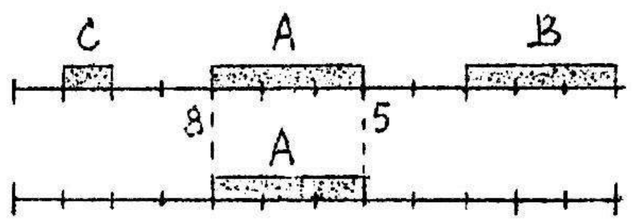
- d'effacer la M 1,
- d'y amener le nombre qui se trouvait entre OD et OF,
- de positionner MD sur OD,
- de laisser la mémoire-signe inchangée.

Autrement dit, de conserver seulement la partie de M 1 comprise entre OD et OF et de positionner MD sur OD.

On a un maintien filtré de M 1.

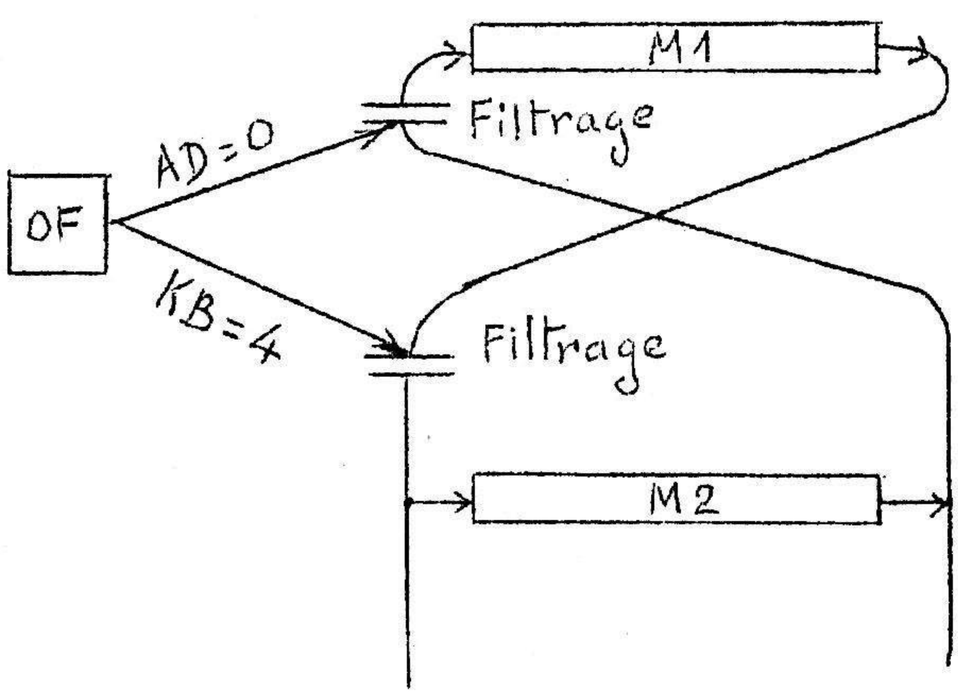
Exemple :

TO	AD	OD	OF	MS 1
6	1	5	8	+
				+



MD
0
5

Remarque : DISTRIBUTION DE CONSTANTES.-



La mémoire OF peut être utilisée comme émettrice de constantes. Cette mémoire contenant un code de 0 à 15 peut être mise en rapport avec le canal-donnée ou avec le canal-résultat. Elle permet l'introduction, l'addition, la soustraction etc... de constantes en mémoire opérateur et l'introduction de constantes en MB

L'accès de la mémoire OF sur le canal-donnée est conditionné par l'indication d'une adresse, l'adresse 0, qui est celle de la mémoire OF.

L'accès de la mémoire OF sur le canal-résultat est conditionné par l'indication d'un type d'opération qui est l'opération KB B0 avec AD=0. Transfert de la mémoire OF en mémoire opérateur.

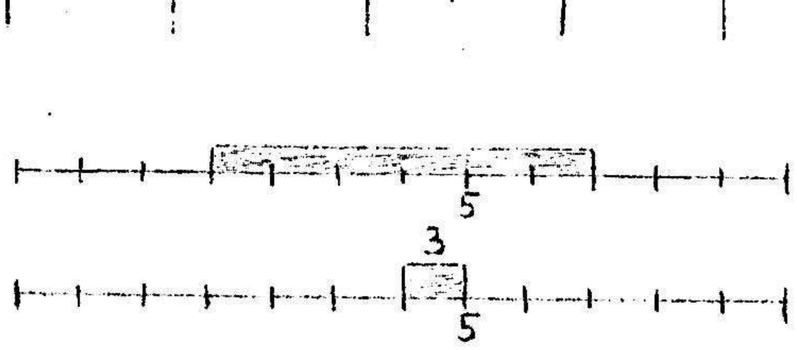
C'est une B0 particulière qui consiste à transférer le contenu de la mémoire OF dans une position de M 1 définie par OD.

Donc :

- la M 1 est intégralement effacée,
- l'OD est transféré en MD,
- le signe est transféré en MS, donc la MS est effacé (constantes positives).

Exemple :

TO	AD	OD	OF
6	←	5	3



MD	MS
3	-
5	+

Remarque : En CD, si la constante est supérieure à 9 il y a émission d'un report sur la position suivante lorsque la constante passe par l'additionneur-soustracteur.

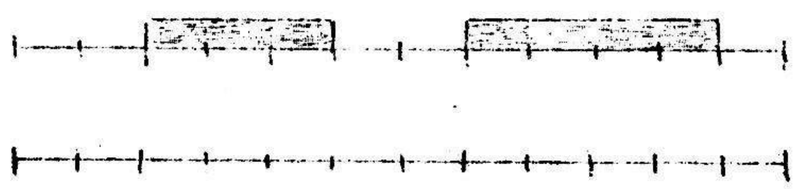
Cette opération est différente d'une KB avec AD = 1 (dans une KB, la MS et la MD ne sont pas altérées).

BO avec AD = 0 et OF = 0 . 6 - (c) -

- La M 1 est totalement effacée,
- la MD est positionnée à la valeur de l'OD,
- la MS est effacée.

Exemple :

TO	AD	OD	OF
6	-	7	←



MD	MS
1	-
7	+

BO avec AD = 0, OD = 0, OF = 0 . 6 . → . - . ←

L'opérateur est totalement effacé (M 1, MS 1, MD)

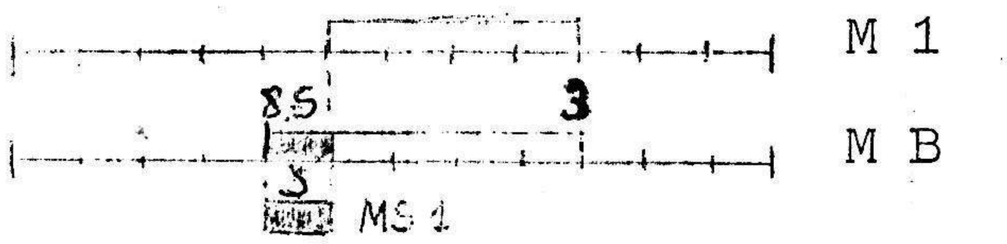
Cette opération est différente d'une ZB avec AD = 1.
OD = 0, OF = 0.

Transfert du signe de mémoire banale vers MS 1. -

Filtrage-signe : les transferts de signe sont contrôlés par un filtrage-signe défini par l'intervalle OF-1, OF.

Soit N un nombre contenu en mémoire banale, son signe S étant sur la gauche. L'opération BO transfère |N| en M 1 et le signe en MS.

TO	AD	OD	OF
6	12	3	8



De OD à OF-1, soit de 3 à 7, le transfert de MB a lieu au profit de M 1 ; de OF-1 à OF, il a lieu au profit de MS ou de M 1.
Si le code inscrit entre OF-1 et OF est un 10, ce code est éliminé à son entrée dans M 1, et aiguillé vers MS ; si ce code est différent de 10, il est transféré en M 1.

Il en résulte que si la position de OF-1 à OF comporte un chiffre arithmétique (cas d'un nombre positif ordinaire où l'on ne réserve pas de place pour le signe), la MS ne reçoit rien puisqu'il s'agit d'un code non égal à 10, alors que M 1 reçoit le chiffre arithmétique pour la même raison.

Remarque : Si le calculateur est en CD, le transfert vers M 1 d'une MB contenant des codes non arithmétiques n'introduit que leur complément à 10. Exemple : Une BO d'un code 12 dans 1 position décimale de MB transfère 2 dans laposition homologue de M 1.

OPERATIONS DE TRANSFERT PARTICULIERES. -

TRANSFERT DE MEMOIRE BANALE EN MEMOIRE DECALAGE BD. TO = 7

7 2 c -

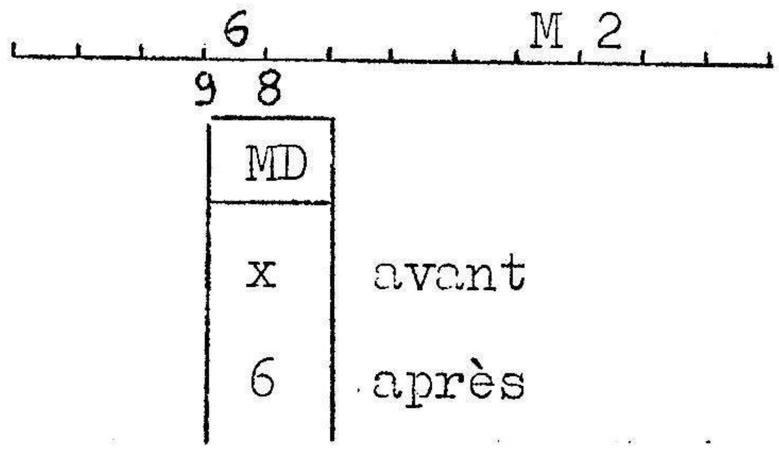
Cette opération permet de transférer le contenu d'une position de la mémoire banale 2 (M 2) dans la MD.

L'OD définit le rang de la position de M2 que l'on transfère.

L'OF est nul.

Exemple :

TO	AD	OD	OF
-	-	-	-
7	2	8	-



Cette opération est utilisée dans le tableau "Point décimal flottant".

Remarque : Si la constante est > 9, elle doit être enregistrée en binaire.

Altération de la mémoire décalage. AMD. 7 - - c

Transfert de c en MD.

Cette opération permet de changer le contenu de MD en n'altérant ni en valeur, ni en position le contenu de la mémoire opérateur.

Comme la MD est chargée de repérer le chiffre le plus à droite enregistré en M 1, cette opération permet de repérer une position quelconque d'un nombre contenu en M 1.

Cette opération peut se concevoir comme une BD avec AD=0

II.- LES OPERATIONS AVEC CADRAGE PREALABLE. -

Définition du cadrage préalable.

Le cadrage préalable est une opération qui permet de décaler d'un certain nombre de positions le contenu de la mémoire opérateur.

Dans tous les cas le cadrage préalable dépend du contenu de la MD avant l'opération et de l'OD de l'opération en cours.

Le nombre de décalages vers la droite est égal à la différence MD - OD modulo 12 c'est-à-dire :

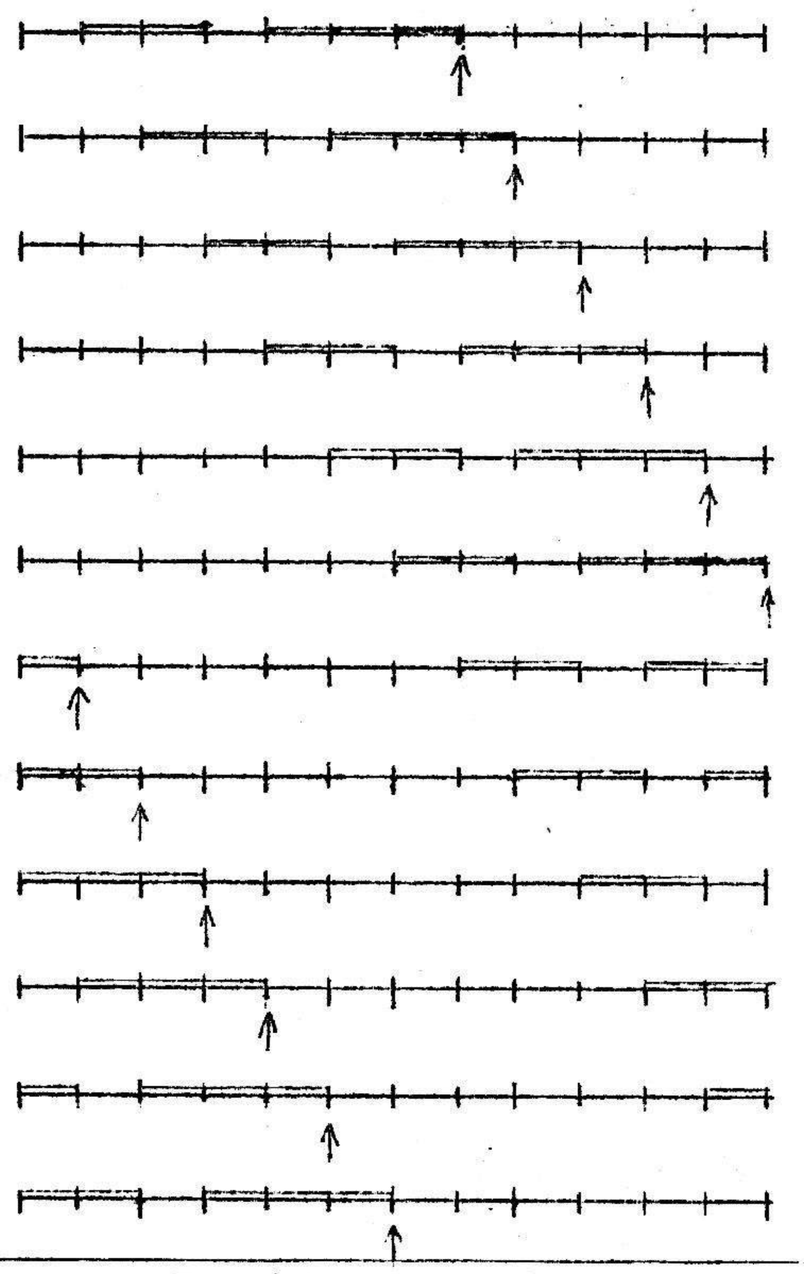
si MD - OD est > 0 il y a MD - OD décalages à droite
si MD - OD est < 0 il y a 12 + MD - OD ---"--- ,
ou OD - MD décalages à gauche

Ménanisme du cadrage préalable.

La MD se décompte de 1 unité à chaque décalage jusqu'à ce que son contenu devienne égal à OD.

Si au cours des décalages, le contenu de MD devient nul, il y a forçage à 12.

Exemple : OD de l'opération en cours = 6



MD	
5	
4	
3	
2	
1	
12	O : forçage à 12
11	
10	
9	
8	
7	
6	Fin du cadrage : MD = OD

Pratiquement, on appliquera la règle suivante :

- si MD - OD>0 décalage à droite du contenu de M1 de MD - OD positions
- si MD - OD<0 décalage à gauche du contenu de M1 de OD - MD positions

A) OPERATIONS DE TRANSFERT.

TRANSFERT DE LA MEMOIRE OPERATEUR EN MEMOIRE BANALE OB. TO = 8.

Cette opération permet de transférer dans une mémoire banale un nombre contenu dans la mémoire opérateur.

L'adresse définit le numéro de la mémoire banale, l'OD et l'OF définissent les positions de cette mémoire qui sont affectées à l'enregistrement du nombre transféré.

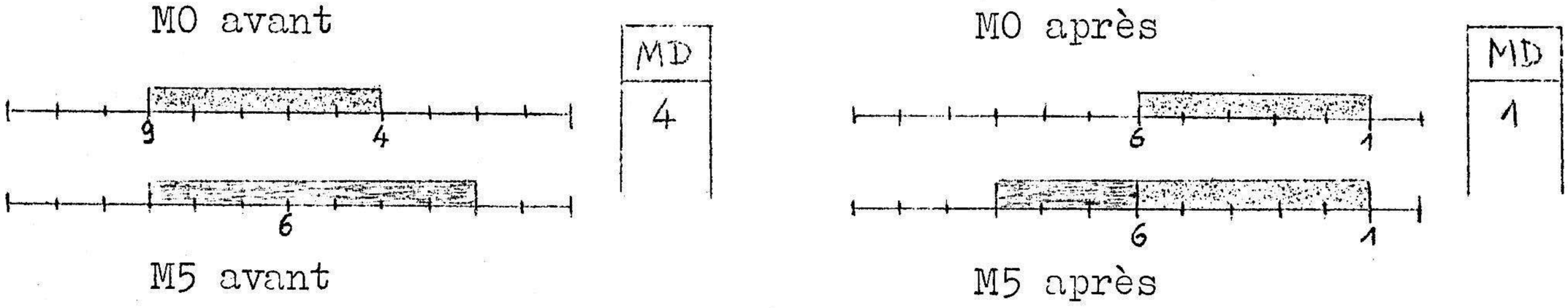
Le transfert est précédé d'un cadrage de la mémoire opérateur afin de placer le nombre à transférer dans les mêmes positions que celles qui lui sont réservées dans la mémoire banale.

(Décalage du nombre dans la M 1 jusqu'à ce que MD = OD).

A la suite de ce cadrage, le nombre est transféré position par position dans la MB dont l'effacement entre OD et OF s'effectue simultanément au transfert.

La MD contient donc toujours à la fin de l'opération l'OD affiché. La mémoire opérateur n'est pas altérée en valeur mais peut l'être en position par le cadrage.

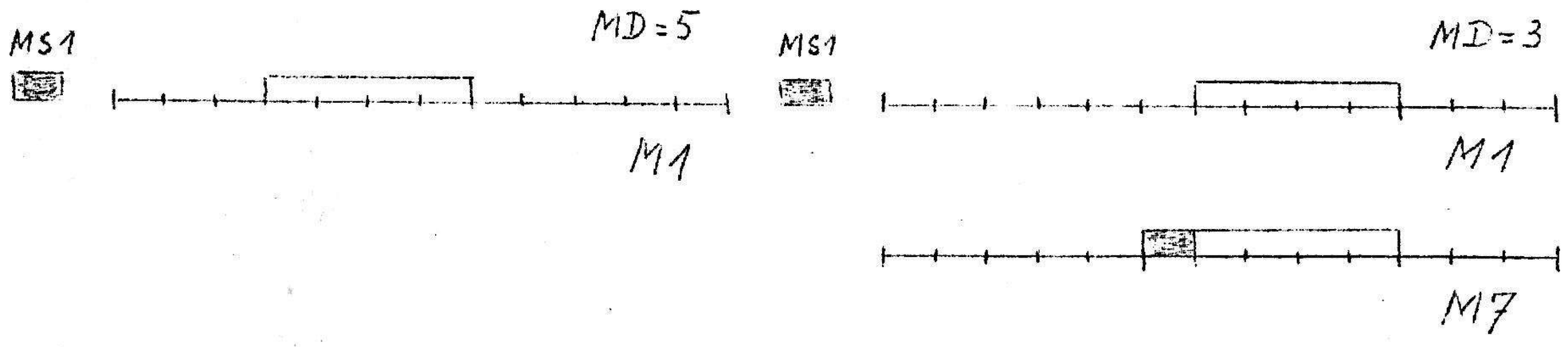
Exemple :



TO	AD	OD	OF
8	5	1	6

La M 1 se cadre suivant OD, ensuite le transfert de M 1 vers M 5 s'effectue position par position, les positions de M 5 comprises entre 1 et 6 étant effacées simultanément au transfert.

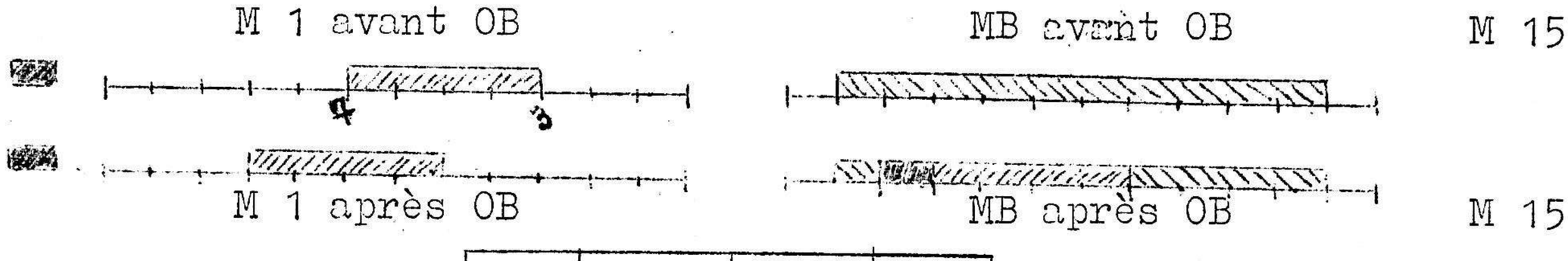
Cas d'un nombre négatif.



TO	AD	OD	OF
8	7	3	8

Il est nécessaire de prévoir une position pour le signe en OF-1, OF.

Cas où la mémoire banale contient un nombre.



TO	AD	OD	OF
8	15	5	10

En plus du nombre transféré depuis M 1, il reste en M 15, les parties du nombre précédemment contenu en dehors de l'intervalle OD-OF.

Il n'y a pas, comme dans le cas d'une BO, effacement total de la mémoire réceptrice avant le transfert, mais seulement effacement partiel entre OD et OF simultanément au transfert.

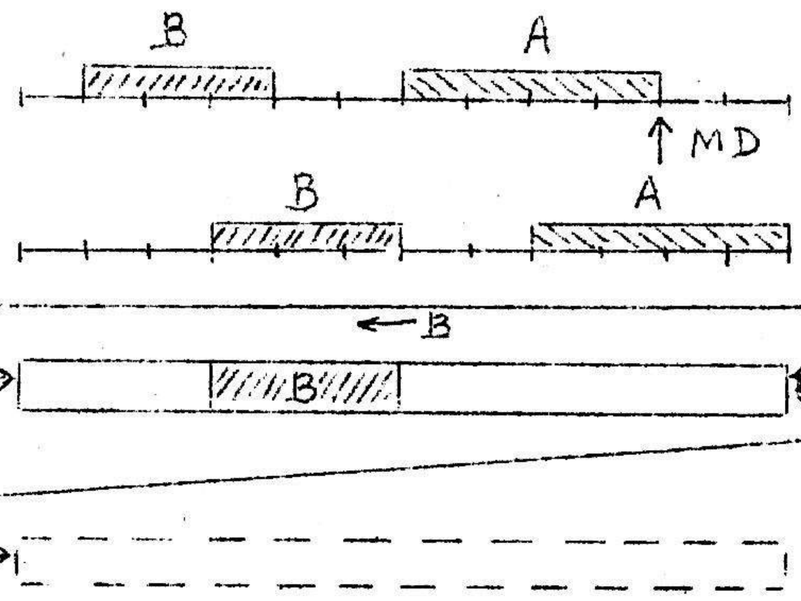
(OB avec AD = 1. Opération difficile et peu employée)

Examinons en détail l'effet de cette opération sur un exemple

TO	AD	OD	OF
8	1	-	4

MD
2

avant l'opération



M 1 avant cadrage

M 1 après cadrage

M 1 comme opérateur

M 1 comme banale

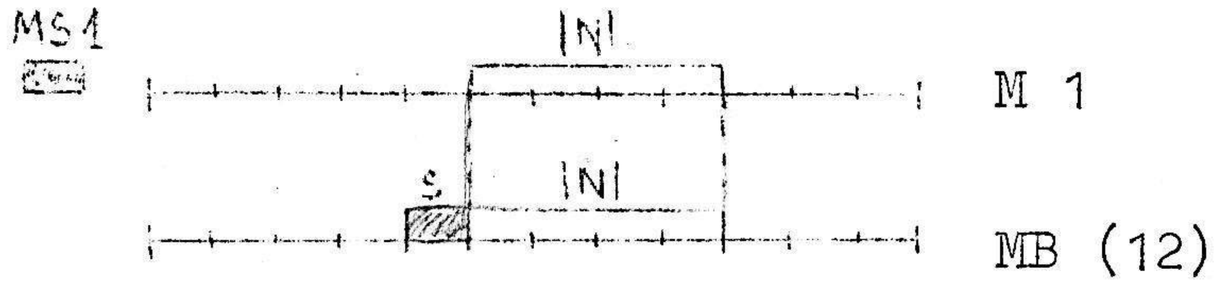
Après le cadrage, il y a effacement de M 1 comme banale entre 0 et 4 puis transfert de ce nombre nul (entre 0 et 4 de M 1 comme opérateur) dans M 1 comme banale.

Cette opération a donc permis de décaler B en effaçant A, mais la MD n'indiquë pas le début de B.

La MS est inaltérée.

C'est une remise à zéro filtrée de M 1 entre OD et OF.

Transfert du signe de MS 1 vers une mémoire banale. -



Lorsque le nombre situé en M 1 est négatif, l'opérateur de signe commande automatiquement au cours de l'opération OB, le transfert d'un code 10 dans la MB dont l'adresse est affichée et dans la position définie par le filtrage-signe entre OF - 1 et OF.

TO	AD	OD	OF
8	12	3	8

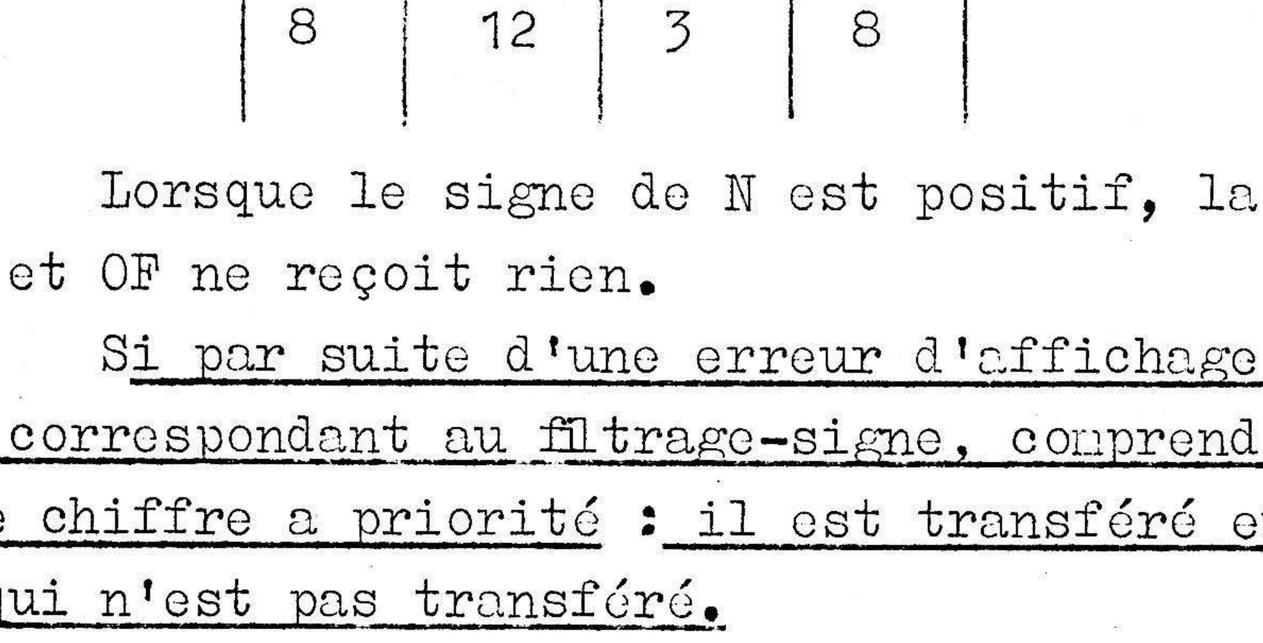
Lorsque le signe de N est positif, la position de MB entre OF - 1 et OF ne reçoit rien.

Si par suite d'une erreur d'affichage sur OF, la position de M 1 correspondant au filtrage-signe, comprend 1 chiffre significatif, ce chiffre a priorité : il est transféré en MB à la place du signe qui n'est pas transféré.

APPLICATION AUX TRANSFERTS. -

En combinant OB et BO, on peut amener un nombre d'une position quelconque, à une autre position quelconque.

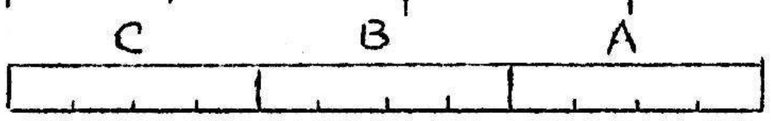
Exemple :



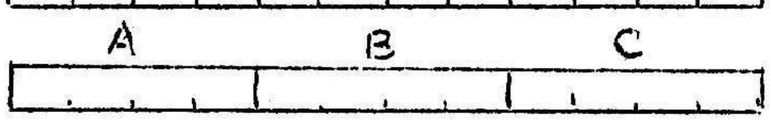
TO	AD	OD	OF
6	3	-	4
8	6	-	4
6	4	-	4
8	6	4	8
6	5	-	4
8	6	8	-

MD
-
-
-
4
-
8

M 5



M 6



TO	AD	OD	OF
6	5	-	4
8	6	8	-
6	5	4	8
8	6	4	8
6	5	8	-
8	6	-	4

MD
-
8
4
4
8
-

B) OPERATIONS LOGIQUES .-

Comparaison CN. TO=9 .-

Cette opération permet de comparer le nombre contenu en M 1 à un nombre contenu dans une mémoire banale.

La comparaison est arithmétique, les signes n'interviennent pas (comparaison en valeur absolue).

L'AD désigne la mémoire banale, l'OD et l'OF définissent la place occupée par le nombre dans la MB.

La comparaison est précédée d'un cadrage qui place le contenu de M 1 en correspondance avec le terme à comparer en mémoire banale.

Le terme contenu en MB dans les positions définies par OD et OF est comparé au contenu intégral de M 1 après cadrage et non aux positions homologues du contenu de cette mémoire. Il faut donc veiller à ce qu'il n'y ait pas plusieurs nombres écrits dans la mémoire opérateur.

Le résultat de la comparaison est enregistré dans une mémoire spéciale où il est conservé jusqu'à ce qu'une nouvelle opération CN soit effectuée.

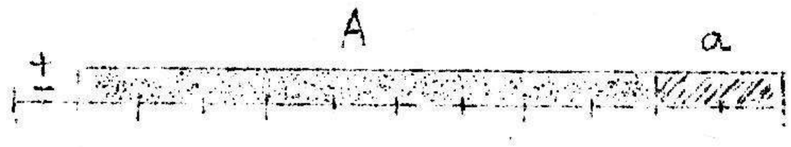
Le résultat permet de conclure :

$>$	$=$	$\geqslant$
$\leqslant$	$\neq$	$<$

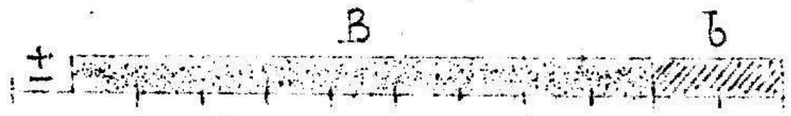
Le sens de la comparaison se lit toujours dans l'ordre suivant ; M 1 comparée à MB.

Exemple :

Soient deux nombres A et B écrits en virgule flottante. Leurs exposants sont a et b. On veut comparer a à b.



M 3



M 4

La BO amène, a en M 1 , après avoir effacé complètement la M 1. La CN compare directement a à b.

	TO	AD	OD	OF
BO	6	3	-	2
CN	9	4	-	2

Résultat $\Rightarrow$ Mémoire Comparaison

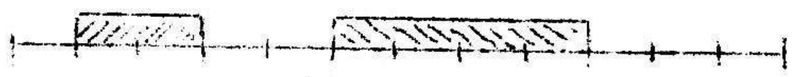
CN avec AD = 1.

Le contenu intégral de M 1 après cadrage est comparé au contenu de M 1 compris entre OD et OF.

On peut ainsi tester la présence, en dehors du filtrage, d'un code en M 1.

TO	AD	OD	OF
9	1	1	5

MD
3
1



M 1 avant cadrage

M 1 après cadrage

Si le résultat de CN est différent (cas ci-dessus), il existe dans la M 1 des codes en dehors du filtrage (1-5).

CN avec AD = 0.

Toutes les opérations avec AD = 0 suppriment le cadrage préalable.

On compare la M 1 au code OF en position OD;

Cette opération peut être utilisée pour comparer un nombre contenu en M 1 à une constante de 1 chiffre.

Exemple :

Cette instruction a été utilisée systématiquement lors du calcul d'instructions ($\leqslant 15$, $\neq 15$ en calcul binaire) pour les progressions d'adresses suivies de variantes-comparaison.

Recadrage d'un nombre en M 1 sans utiliser une mémoire banale de transfert.

On utilise l'opération comparaison avec une adresse quelconque comprise entre 1 et 15. (Tableau PdF Ligne 59).

L'inconvénient de ce procédé est l'altération de la mémoire comparaison.

C) OPERATIONS ARITHMETIQUES. -

ADDITION. AN. TO = 10.

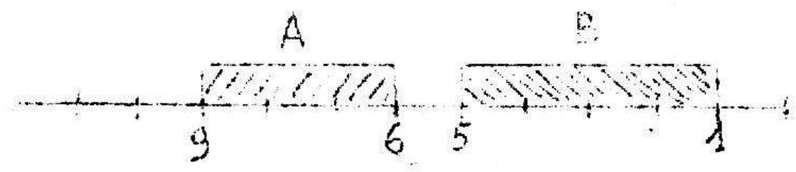
Cette opération permet d'ajouter un nombre contenu dans la mémoire opérateur et un nombre contenu dans une mémoire banale.

L'AD définit le **numéro** de la mémoire banale, l'OD et l'OF définissent la place occupée par le nombre en mémoire banale.

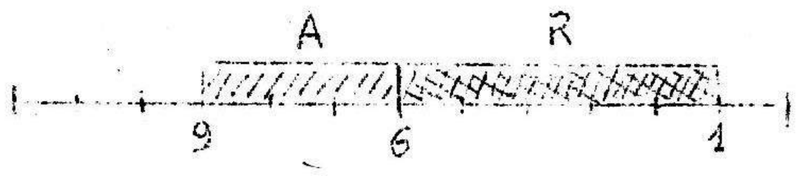
Le nombre en M 1 se recadre selon les règles du cadrage préalable et l'addition s'effectue alorsposition par position de OD à OF.

Exemples :

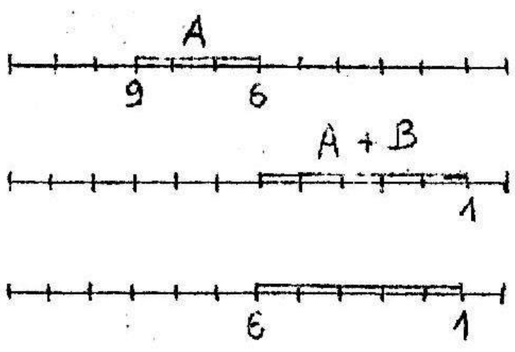
1°) On a A et B en M 4 ; on veut effectuer A + B et mettre R à la place de B.



M 4 avant



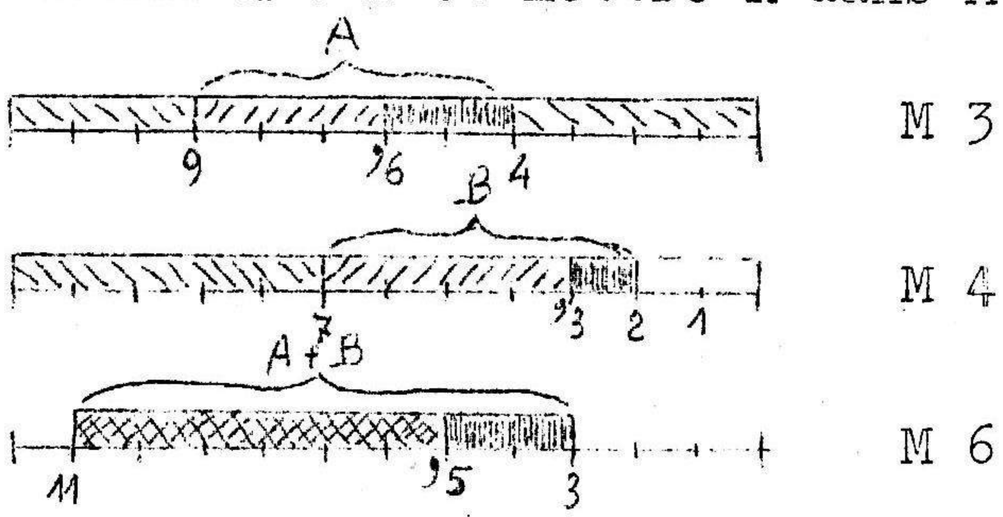
M 4 après



TO	AD	OD	OF
6	4	6	9
10	4	1	5
8	4	1	6

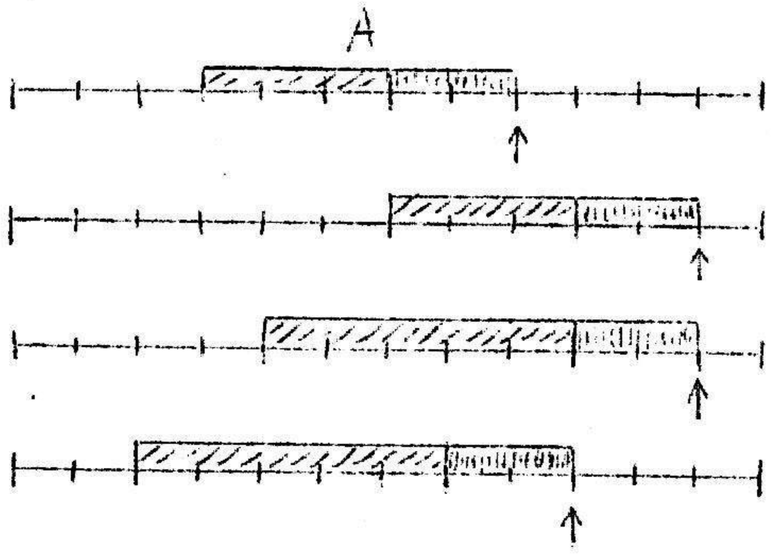
MD
6
1
1

2°) Deux nombres A et B avec partie décimale sont en M 3 et M 4 ; on veut effactuer A + B et mettre R dans M 6



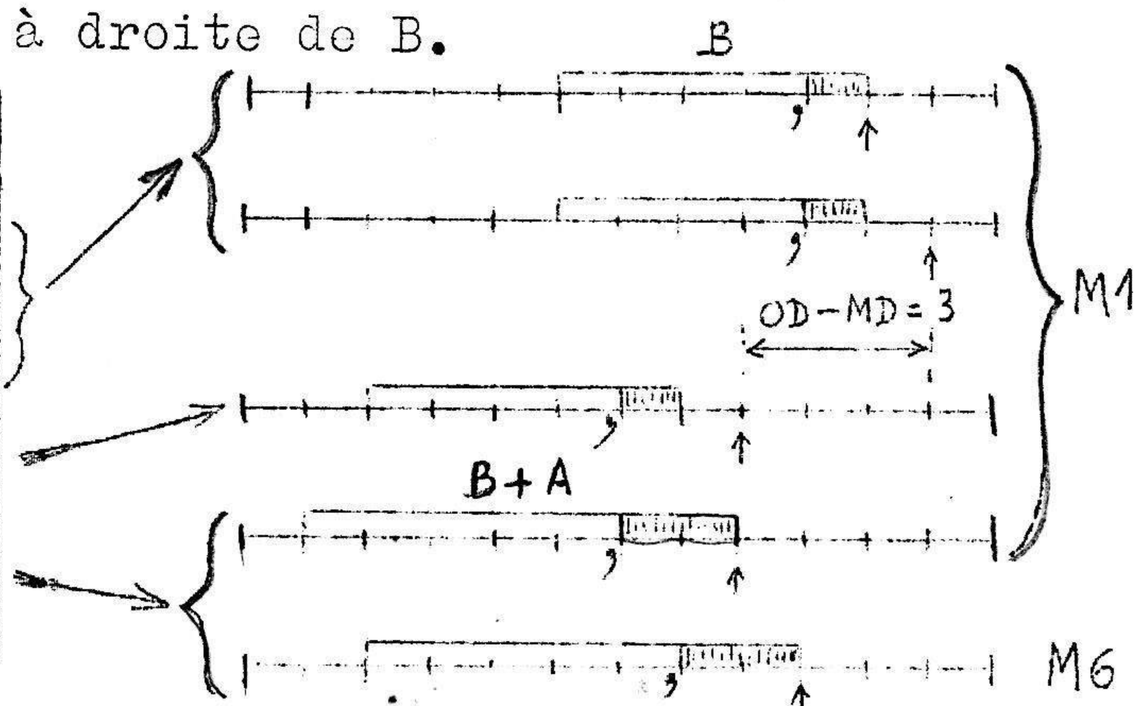
1er cas : Il n'y a rien d'écrit à droite de B dans M 4.
La séquence est la suivante :

TO	AD	OD	OF
6	3	4	9
10	4	1	7
8	6	3	11



2ème cas : Il y a un nombre écrit à la droite de B dans M 4.
Il faut changer le contenu de la MD à l'aide d'une AMD afin de ne pas ajouter le 1er chiffre à droite de B.

TO	AD	OD	OF	MD
6	4	2	7	2
7	-	-	1	1
10	3	4	9	4
8	6	3	-	3



Les flèches indiquent la position de M 1 repérée par la MD.

AN avec AD = 1.

Le contenu de la M 1 est doublé dans les positions définies par le filtrage après cadrage préalable. Il faut tenir compte du report qui peut naitre de ce doublage, de façon à ce qu'il arrive sur une position libre de M 1

AN avec AD = 0.

Opération sans cadrage préalable qui consiste à additionner au contenu de la M 1 le code affiché en OF. La position de M 1 sur laquelle le code est ajouté est définie par l'OD.

Cette opération est utilisée pour réaliser l'arrondi à l'unité la plus proche par addition de 5 sur la décimale à arrondir

(à condition que le nombre soit positif, ou si l'on considère la valeur absolue).

Opération utilisée couramment en calcul d'instructions.

SOUSTRACTION. SN. TO = 11.

Cette opération retranche du nombre contenu en M 1, un nombre contenu en MB.

SN avec AD = 1.

Equivaut à une remise à zéro filtrée de M 1 après cadrage préalable.

Il y a inversion du signe de MS, si la remise à zéro est totale.

SIGNE DE O.

D'après la remarque précédente : $\left.\begin{array}{l} +A\,|\,-A\,| = -0 \\ -A\,|\,+A\,| = +0 \end{array}\right\}$ si $A > 0$

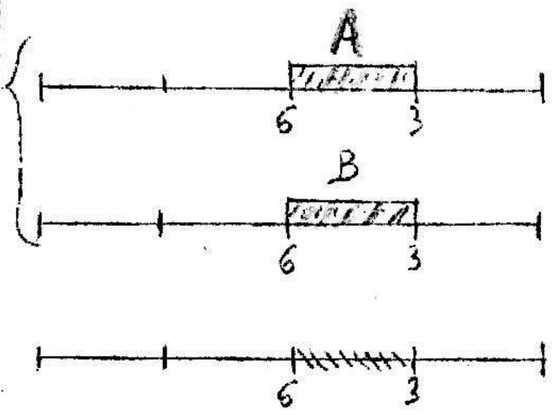
Ceci est important si l'on utilise des VMS^+ ou des VMS^-

Remarque :Le signe de O ne suit les règles précédentes que dans les opérations d'addition et de soustraction pour lesquelles aucun cadrage préalable n'a été effectué réellement avant l'opération de soustraction.

Exemple :

TO	AD	OD	OF
11	B	3	6

MS1		MD
+	M1	3
	MB	3
-		3



Si A - B = 0 le zéro obtenu a le signe moins.

Dans les autres cas le signe de zéro reste arbitraire.

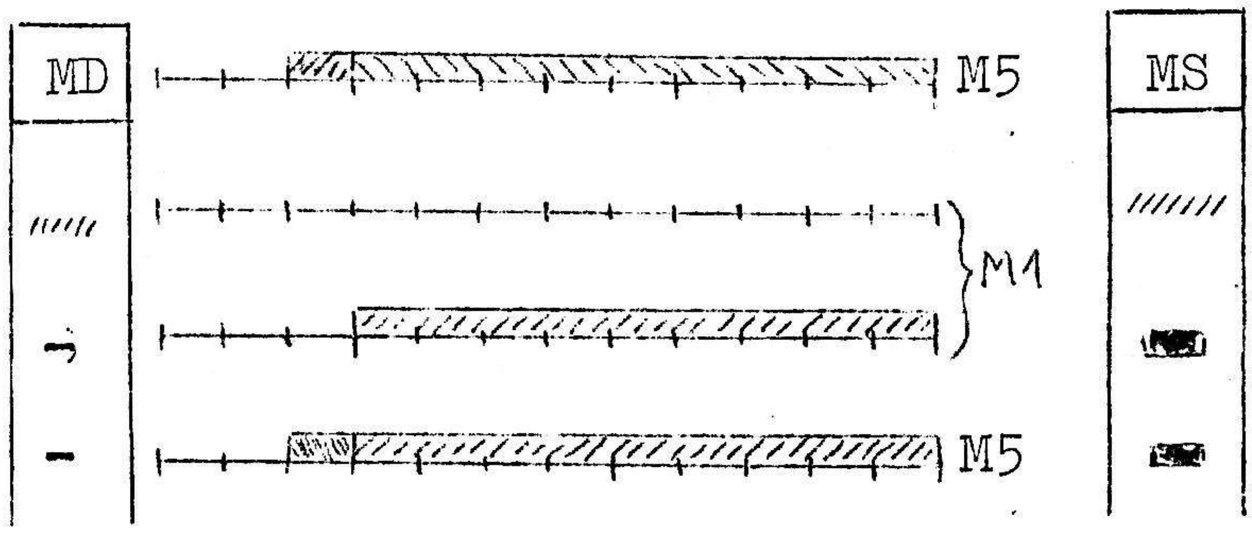
SN avec AD = 0.

Opération sans cadrage préalable qui soustrait au contenu de M 1 le code affiché en OF ; la soustraction s'effectue sur la position OD.

Opération utilisée en calcul d'instructions.

Inversion du signe d'un nombre.

TO	AD	OD	OF
6	-	-	-
11	5	-	10
8	5	-	10



C H A P I T R E III

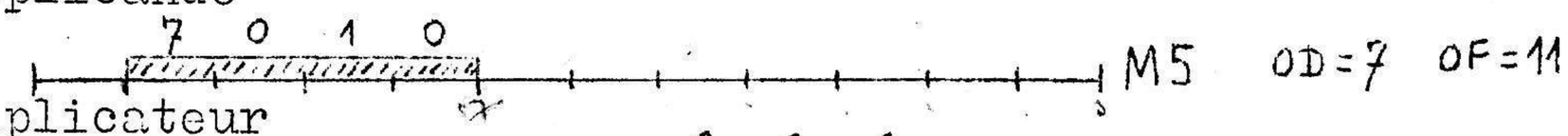
MULTIPLICATION ET DIVISION .-

Ces opérations sont réalisées par la machine sous forme d'additions ou de soustractions successives. Il existe deux types distincts pour chacune de ces deux opérations : opérations réduites, opérations complètes.

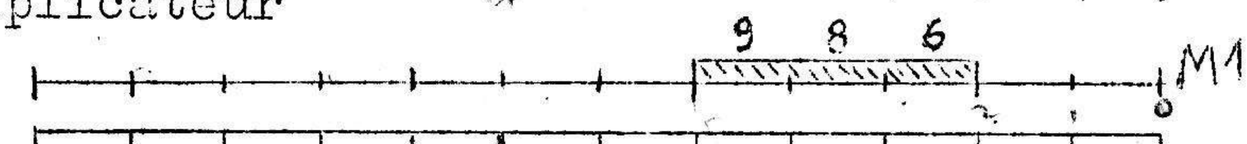
MULTIPLICATION REDUITE. MR. TO = 12. Ex. 12. 5. 7. 11

Dans cette opération, tous les calculs s'effectuent dans M 1.
Un exemple fera comprendre le mécanisme de cette opération.

Multiplicande



Multiplicateur



												MD
0	0	0	0	0	0	0	9	8	6	0	0	7
0	0	0	0	0	0	0	0	9	8	6	0	6
0	0	0	0	0	0	0	0	0	9	8	6	5
0	7	0	1	0	0	0	0	0	9	8	5	5
1	4	0	2	0	0	0	0	0	9	8	4	5
2	1	0	3	0	0	0	0	0	9	8	3	5
2	8	0	4	0	0	0	0	0	9	8	2	5
3	5	0	5	0	0	0	0	0	9	8	1	5
4	2	0	6	0	0	0	0	0	9	8	0	5
0	4	2	0	6	0	0	0	0	0	9	8	4
1	1	2	1	6	0	0	0	0	0	9	7	4
...	...	...	...	...	...	...	...	...	...	...	..	...
6	0	2	8	6	0	0	0	0	0	9	0	4
0	6	0	2	8	6	0	0	0	0	0	9	3
1	3	0	3	8	6	0	0	0	0	0	8	3
...	...	...	...	...	...	...	...	...	...	...	..	...
6	9	1	1	8	6	0	0	0	0	0	0	3
0	6	9	1	1	8	6	0	0	0	0	0	2
0	0	6	9	1	1	8	6	0	0	0	0	1
0	0	0	6	9	1	1	8	6	0	0	0	0

Autrement dit, la machine réalise une addition du multiplicande et une diminution de 1 du multiplicateur (en position 0-1), lorsque cela est possible, sinon un décalage simultané du multiplicateur et du produit vers la droite avec diminution de 1 du contenu de MD. L'opération est terminée lorsque MD = 0.

Cela exige :

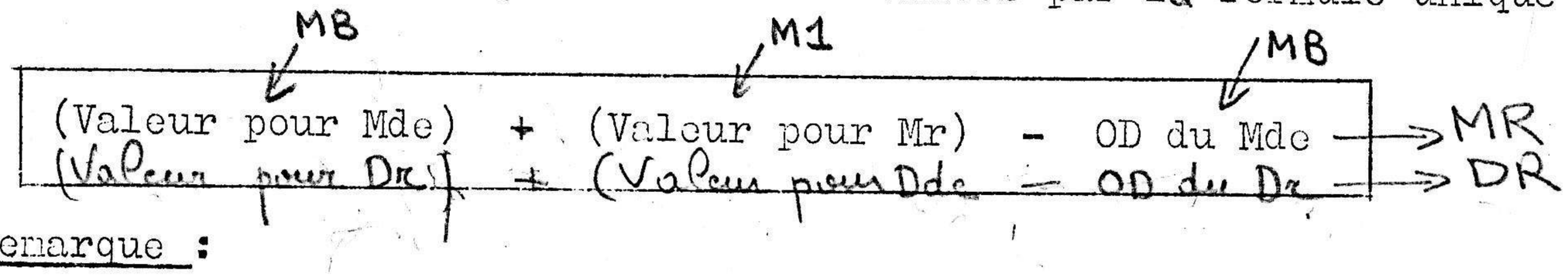
- que l'OD du multiplicande tombe à gauche du multiplicateur (abstraction faite du signe qui est en MS).
- que l'OF du multiplicande ne dépasse pas 11 (abstraction faite du signe).

Le résultat s'interprête comme le produit de deux nombres entiers dont l'un aurait des unités en (0-1) et l'autre en (OD-OD+1) le produit ayant ses unités en (0-1).

Ces remarques permettent de déterminer facilement ;

- l'Od et l'OF à donner au produit en fonction de ceux des facteurs,
- la position du chiffre extrême droit du produit si l'on connait ceux des facteurs,
- la position de la virgule du produit si l'on connait celle des facteurs.

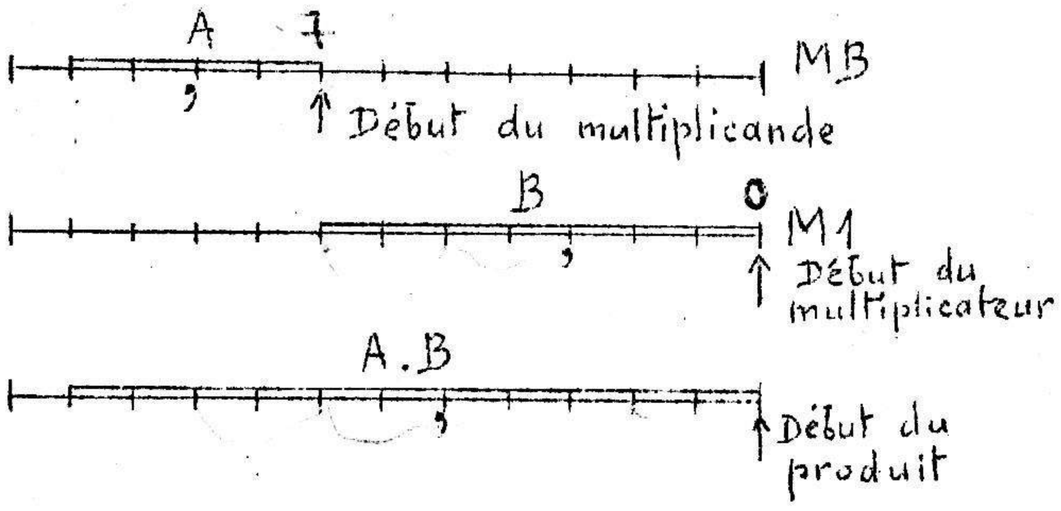
Toutes ces quantités sont données par la formule unique :



Remarque :

Par définition, à la fin de l'opération MD = 0.

Cadrage de base de la multiplication.



Ce cadrage est un cadrage de base c'est-à-dire que les facteurs et le résultat n'occupent pas forcément toutes les positions de mémoire désignées par A, B et AB et peuvent êtrebordées de zéros à droite et à gauche.

Multiplications particulières.-

Pour comprendre ces multiplications, il y alieu de remarquer que la condition de non-chevauchement du multiplicande et du multiplicateur n'est pas exigée par la machine, mais est nécessaire seulement pour empêcher les chiffres du produit de se melanger à ceux du multiplicateur.

Mr avec AD = 1, OD = 0, OF = 0. 12 1 - -

Soit A le contenu de M 1.

La mémoire décalage se positionne à 0.

La multiplication proprement dite ne commence donc pas.

Par contre, l'opérateur de signes effectue le produit des signes :

signe (a) x signe (a) = $\left[\text{signe (a)}\right]^2$ = +

On obtient donc |A| dans M 1.

Mr avec AD = 0. 12 ÷ c d

C'est la multiplication de la constante d placée en c, par le contenu de M 1.

Il faut naturellement que la position c tombre à gauche du chiffre extrême du multiplicateur.

A la fin de l'opération MD = 0.

Mr avec AD = 0 et OF = 0. 12 - c -

Le produit ainsi réalisé est nul (OF=0).

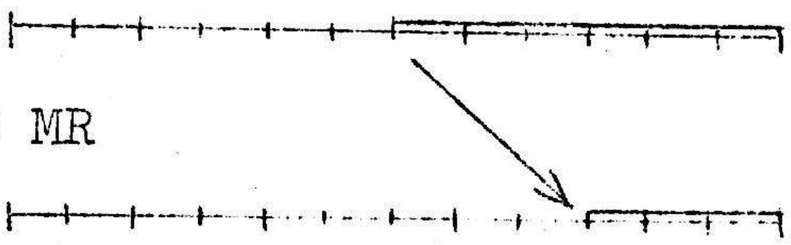
Le multiplicateur est décalé vers la droite de c positions avec perte des chiffres sortants.

Ici, l'OD peut être plus petit que l'OF multiplicateur

Exemple :

M 1 avant MR

M 1 après MR



MD
3
0

TO	AD	OD	OF
12	-	3	-

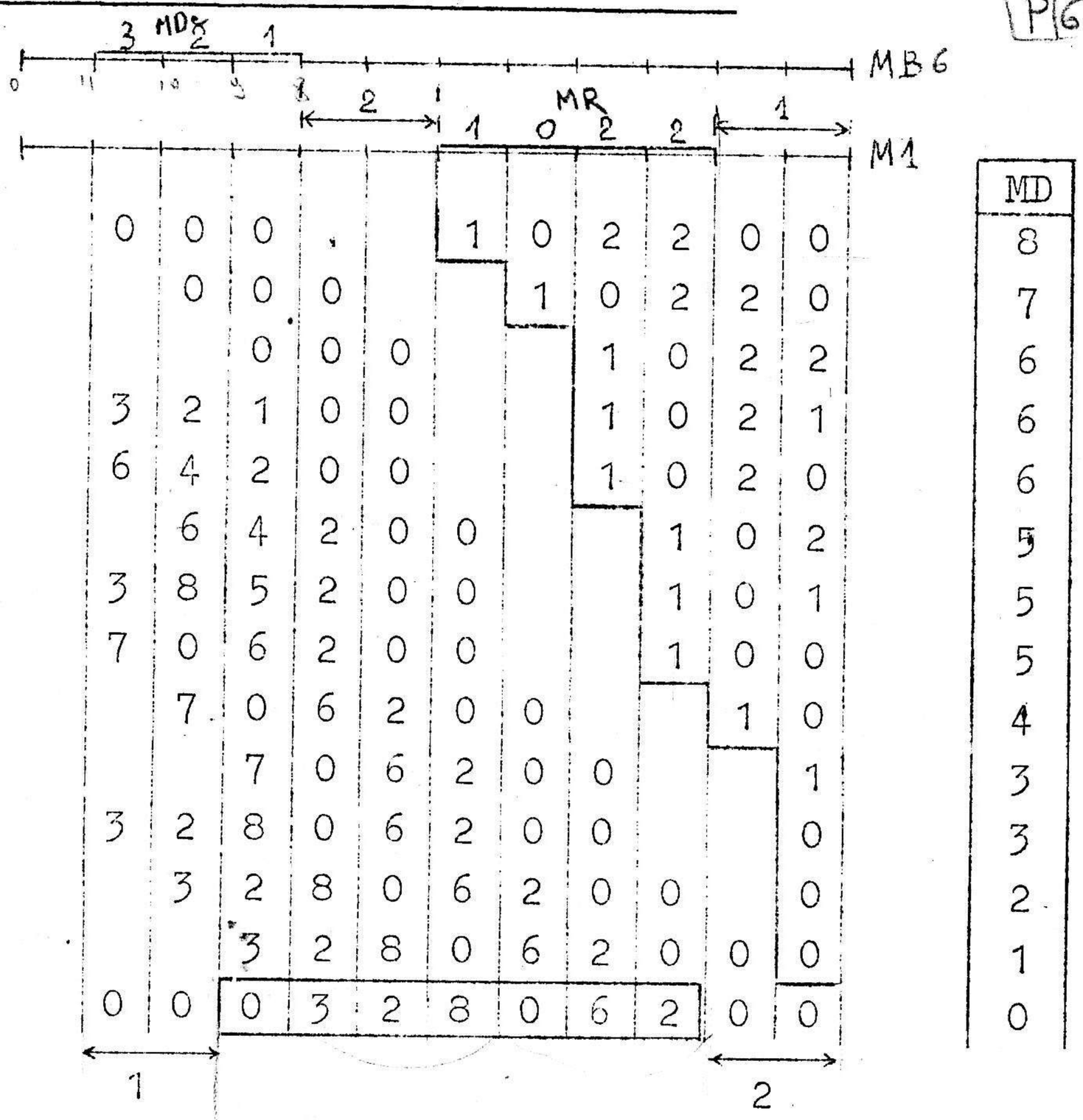
On a ainsi décalé la M 1 en la remettant particllement à zéro. A la fin de l'opération MD = 0

Cette opération est employée :

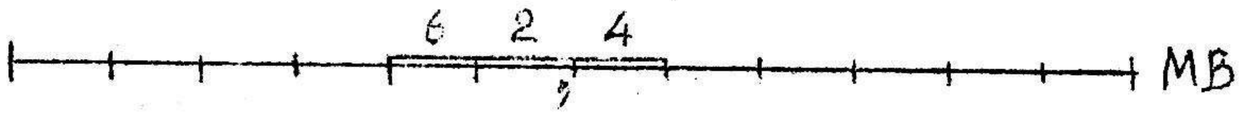
- pour décaler un nombre à droite dans M 1 (cadrage à droite)
- pour abandonner les décinales ou une partie d'un nombre.

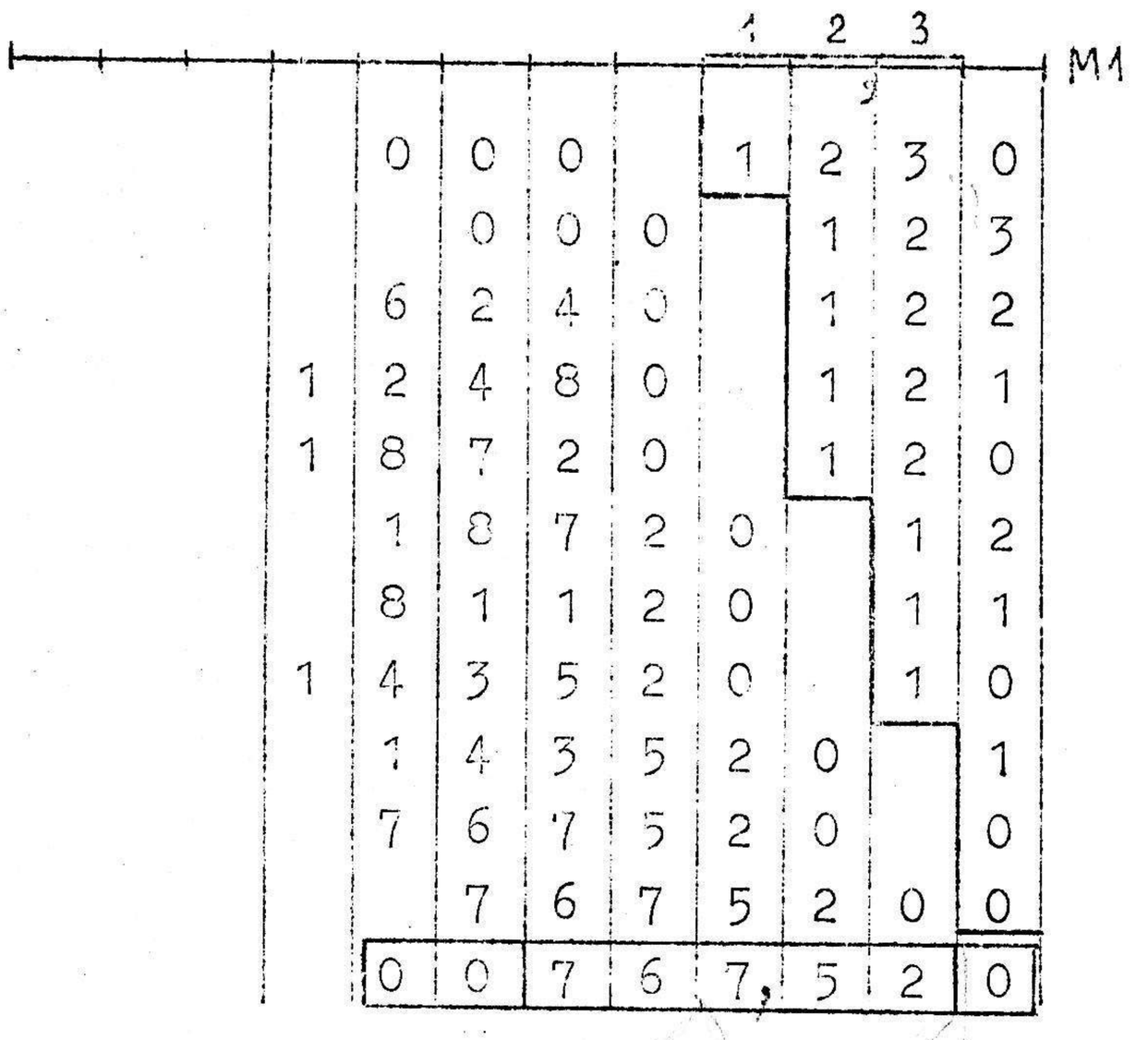
EXEMPLES DE MULTIPLICATIONS REDUITES.

P 6 8 H 1



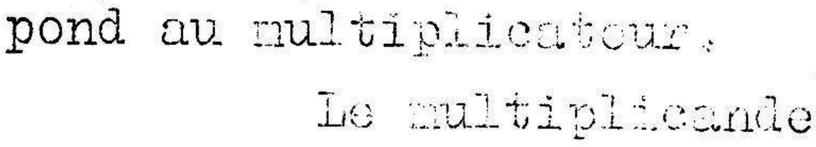
MD
8
7
6
6
6
5
5
5
4
3
3
2
1
0





									MD
		0	0	0		1	2	3	0	5
			0	0	0		1	2	3	4
		6	2	4	0		1	2	2	4
	1	2	4	8	0		1	2	1	4
	1	8	7	2	0		1	2	0	4
		1	8	7	2	0		1	2	3
		8	1	1	2	0		1	1	3
	1	4	3	5	2	0		1	0	3
		1	4	3	5	2	0		1	2
		7	6	7	5	2	0		0	2
			7	6	7	5	2	0	0	1
		0	0	7	6	7,	5	2	0	0

Multiplication par 2^n



On place A dans M 1 de 1 à 11, et n en position 0-1.

L'opération 12 . 1 . 1 . - donne $A.2^n$

MULTIPLICATION COMPLETE MC. TO = 14.

La multiplication précédente ne permet de traiter que des nombres assez courts. La multiplication complète remédie à cet inconvénient. Elle utilise un couplage de M 1 et de M 2 qui réalisent une seule mémoire de 24 positions.

M 1 correspond aux poids forts et M 2 aux poids faibles. Le contenu δ de la mémoire décalage repère une position de M 2 qui partage les 24 positions en 2 groupes, le groupe de droite correspond au multiplicateur.

Le multiplicande qui est dans une mémoire banale s'ajoute dans M 1 en même temps que l'on enlève 1 en position (0-1) de M 2.

Lorsqu'il y a 0 dans cette position, on réalise une translation de l'ensemble M 1, M 2 vers la droite et un déconptage de 1 de la MD.

Les capacités permises sont donc :

- 12 pour le multiplicateur (M 2),
- 11 pour le multiplicande (abstraction faite du signe).

Le produit s'interprête comme celui de 2 nombres entiers ayant les positions positions unités en δ de M 2 (valeur de MD) pour le multiplicande et en 0-1 de M 2 pour le multiplicateur.

Le résultat occupe l'ensemble M 1 - M 2 avec ses unités en (0-1) de M 2.

A partir de là, on détermine finalement :

- les positons de la virgule du produit connaissant celles des facteurs,
- le chiffre extrême droit du produit connaissant ceux des facteurs

En fait, la multiplication complète est une opération qui nécessite une préparation en raison des particularités suivantes :

a) le positionnement initial de MD n'est plus donné par l'OD du multiplicande comme en multiplication réduite.

Il faut au préalable positionner MD (bien entendu, en fin d'opération MD est nulle).

b) en ce qui concerne le signe du multiplicateur, il doit être en MS et il ne faut pas qu'il risque d'être décompté en M 2 comme un 10, c'est-à-dire qu'il ne doit pas être en M 2, ou, s'il y est, que la MD soit positionnée à sa droite (mais cependant à la gauche des chiffres).

On satisfait à ces conditions en même temps qu'on modifie dans la mesure du possible le positionnement du produit par la séquence suivante :

TO	AD	OD	OF
6	b	d	f
8	2	d'	f'
6	2	f'-1	f'

b désigne la mémoire banale contenant le multiplicateur au départ; d et f, l'OD et l'OF du multiplicateur dans b.

La seule restriction est que $f' - d' \geqslant f - d$

Le positionnement du produit est d'ailleurs lié uniquement à f' - d'.

Il y a à la gauche des chiffres correspondants à l'OD du multiplicateur et du multiplicande exactement

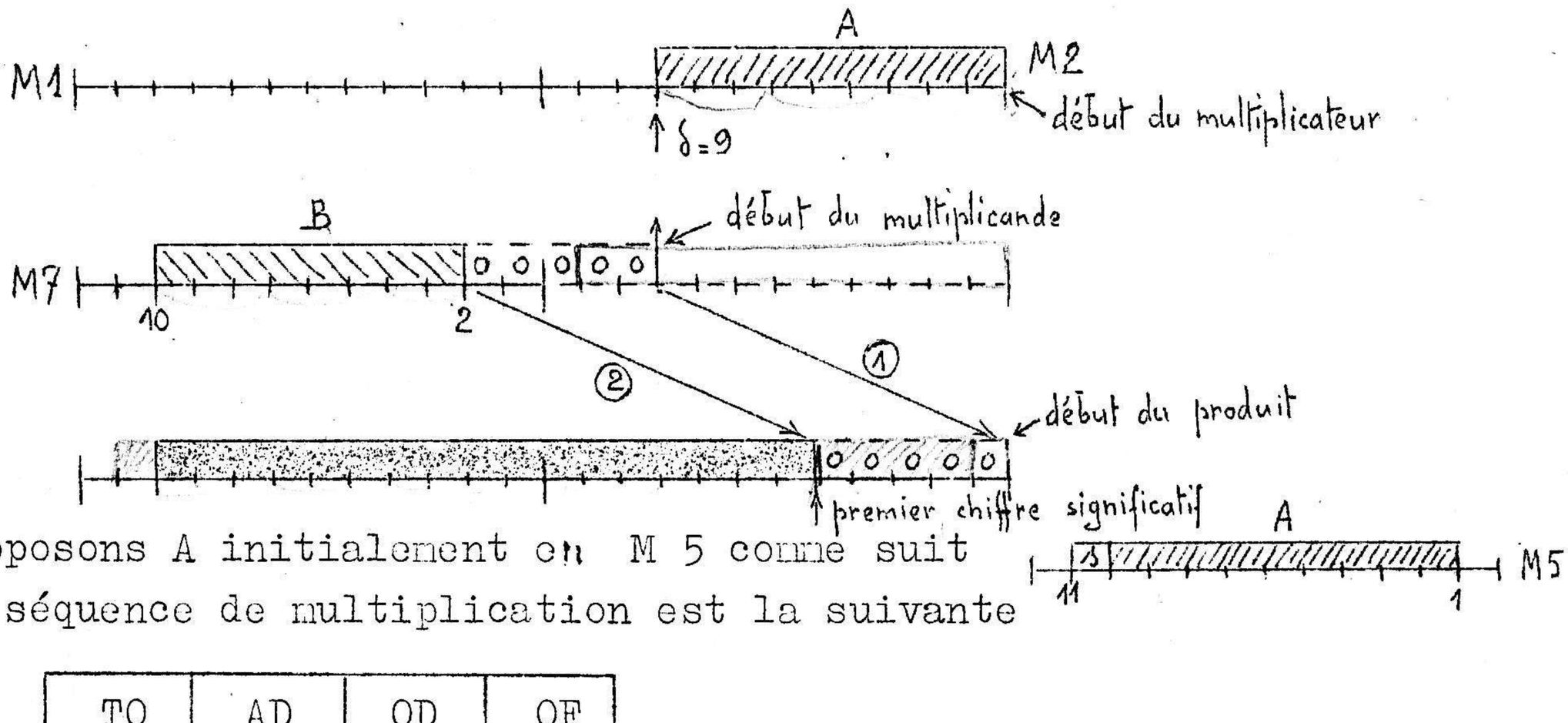
$$13 - f' + d' \text{ zéros}$$

c'est-à-dire au plus 13 - f + d et au moins 1.

Remarque : Il existe d'autres manières de préparer la MC. En particulier, si l'on évite d'introduire le signe du multiplicateur en M2, on peut positionner MD sur 12 (il suffit d'y placer 0 avec une AMD).

L'AMD positionne la MD sans remettre à 0 la M 1, ce qui permet des opérations de la forme ab + c.

Schéma de positionnement en multiplication complète



Supposons A initialement en M 5 comme suit

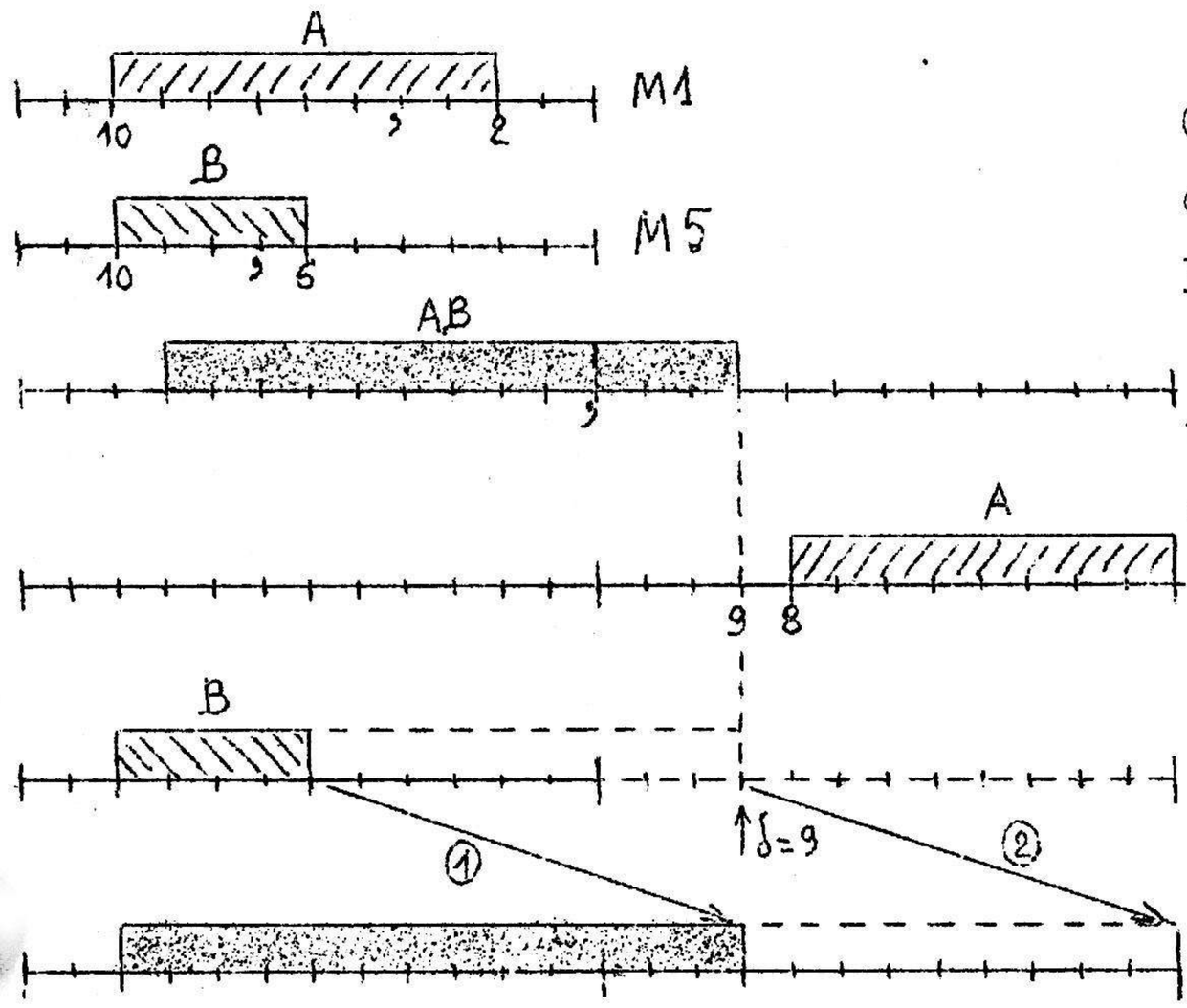
La séquence de multiplication est la suivante

TO	AD	OD	OF	
6	5	10	11	Appel du multiplicateur en M 1
8	2	-	10	Transfert du multiplicateur en M 2
6	2	9	10	Rappel du signe en MS 1 (qui y est déjà) et positionnement de MD à 9
14	7	2	11	Multiplication A.B.

Les zéros à droite de B (ou à gauche de δ) se retrouvent dans les poids faibles du produit par le tracé du parallélogramme.

Pour éviter que la multiplication soit trop longue, A ne doit pas avoir de zéros à droite ou à gauche.

Exemple : A et B sont écrits en M 1 et M 5, on fait AB avec partie décimale en M 2 (pour abandonner cette partie décimale)



On met d'abord A en place en le cadrant le plus à droite dans M2
La règle du parallèlogramme donne
∂ = 9 (Ici on connait la position du 1er chiffre significatif, on en déduit ∂).

TO	AD	OD	OF
8	2	-	9
3	2	8	9
7	-	-	9
14	5	6	11

Multiplications complètes particulières .

MC avec AD = 0. 14 - c d

Multiplication de la constante d placée en c par le contenu de M 2. Il n'y a aucune restriction sur c.

MC avec AD = 0 et OF = 0; 14 - c -

c n'importe pas. Il s'agit seulement d'un décalage global de M 1 - M 2 vers la droite de δ positions (δ = contenu de MD)
Avec δ = 0, on a 12 décalages.

Exemple : d'utilisation.

Préparation d'une MC avec MD = 12.

TO	AD	OD	OF	
6	b	d	f	
14	-	-	-	décalage de d positions
14	-	-	-	décalage de 12 positions

Exemple de Multiplication Complète

TO	AD	OD	OF
7	-	-	-
14	B	-	-

5 1 6 1 9 2

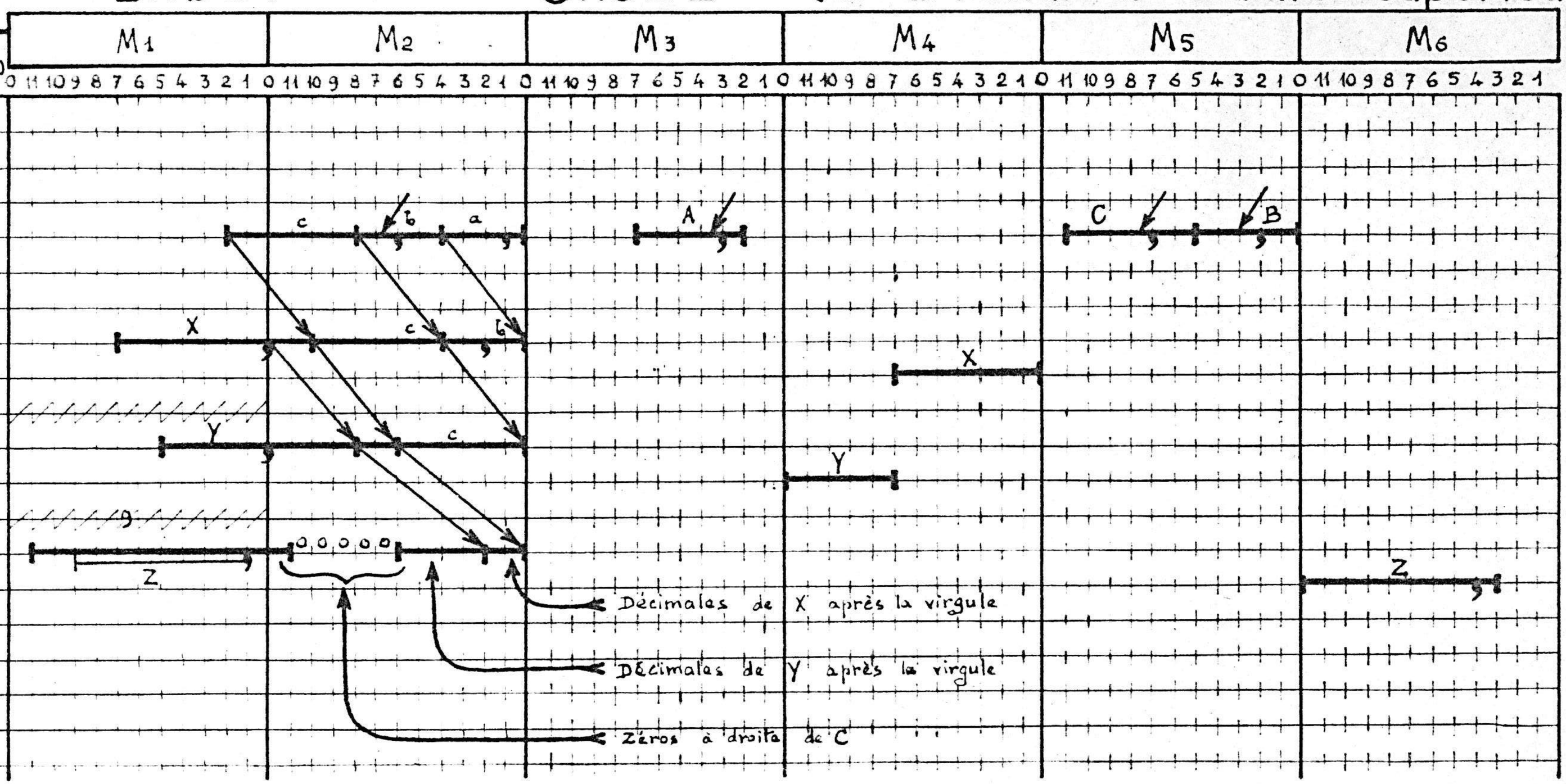
											MB													MD
												M1										M2		
												2	1	3	4	0	5	2	4	6	3	1	2	12
	3	2	4	5	1	1	3	4	6	2	1	2	1	3	4	0	5	2	4	6	3	1	1	11
	6	4	9	0	2	2	6	9	2	4	2	2	1	3	4	0	5	2	4	6	3	1	0	11
		6	4	9	0	2	2	6	9	2	4	2	2	1	3	4	0	5	2	4	6	3	1	11
	3	8	9	4	1	3	6	1	5	4	5	2	2	1	3	4	0	5	2	4	6	3	0	11
		3	8	9	4	1	3	6	1	5	4	5	2	2	1	3	4	0	5	2	4	6	3	10
	3	6	3	4	5	2	7	0	7	7	5	5	2	2	1	3	4	0	5	2	4	6	2	11
	6	8	7	9	6	4	0	5	3	9	6	5	2	2	1	3	4	0	5	2	4	6	1	11
1	0	1	2	4	7	6	4	0	0	1	7	5	2	2	1	3	4	0	5	2	4	6	0	11
	1	0	1	2	4	7	6	4	0	0	1	7	5	2	2	1	3	4	0	5	2	4	6	9
2	0	4	8	3	1	5	7	1	7	2	7	7	5	2	2	1	3	4	0	5	2	4	0	11
	2	0	4	8	3	1	5	7	1	7	2	7	7	5	2	2	1	3	4	0	5	2	4	8
1	5	0	2	8	7	6	9	5	6	5	6	7	7	5	2	2	1	3	4	0	5	2	0	11
	1	5	0	2	8	7	6	9	5	6	5	6	7	7	5	2	2	1	3	4	0	5	2	7
	7	9	9	3	1	0	3	8	7	0	7	6	7	7	5	2	2	1	3	4	0	5	0	11
		7	9	9	3	1	0	3	8	7	0	7	6	7	7	5	2	2	1	3	4	0	5	6
1	7	0	2	4	8	7	7	6	9	7	5	7	6	7	7	5	2	2	1	3	4	0	0	11
	1	7	0	2	4	8	7	7	6	9	7	5	7	6	7	7	5	2	2	1	3	4	0	5
		1	7	0	2	4	8	7	7	6	9	7	5	7	6	7	7	5	2	2	1	3	4	4
1	3	1	5	0	7	0	2	6	2	5	3	7	5	7	6	7	7	5	2	2	1	3	0	11
	1	3	1	5	0	7	0	2	6	2	5	3	7	5	7	6	7	7	5	2	2	1	3	3
1	1	0	5	0	4	1	0	6	4	8	8	3	7	5	7	6	7	7	5	2	2	1	0	11
	1	1	0	5	0	4	1	0	6	4	8	8	3	7	5	7	6	7	7	5	2	2	1	2
	4	3	5	0	1	5	4	5	2	6	9	8	3	7	5	7	6	7	7	5	2	2	0	11
		4	3	5	0	1	5	4	5	2	6	9	8	3	7	5	7	6	7	7	5	2	2	1
	6	9	2	5	2	4	2	3	7	6	8	9	8	3	7	5	7	6	7	7	5	2	0	11
		6	9	2	5	2	4	2	3	7	6	8	9	8	3	7	5	7	6	7	7	5	2	0

EXEMPLE DE MULTIPLICATIONS SUCCESSIVES

-:-:-

A x a = X B x b = Y C x c = Z (Z arrondi à l'unité supérieure)

NL	Codes TO	AD	OD	OF	Opération	MD
					Intr.	
0	7			4	AMD	4
1	14	3	2	7	MC	0
2	8	4		7	OB	0
3	6		4		BO	4
4	14	5		5	MC	0
5	8	4	7		OB	7
6	6		6	9	BO	6
7	14	5	5	11	MC	0
8	8	6	3		OB	3



On enregistre plusieurs multiplicateurs en M 2. En positionnant avant chaque MC la MD au nombre de chiffres du multiplicateur à traiter, l'opération décale les autres facteurs en amenant le suivant à droite de M 2.

Le 1er positionnement de la MD se fait à l'aide d'une AMD et non à l'aide d'une BO comme dans les séquences suivantes pour ne pas effacer la mémoire opérateur qui contient les

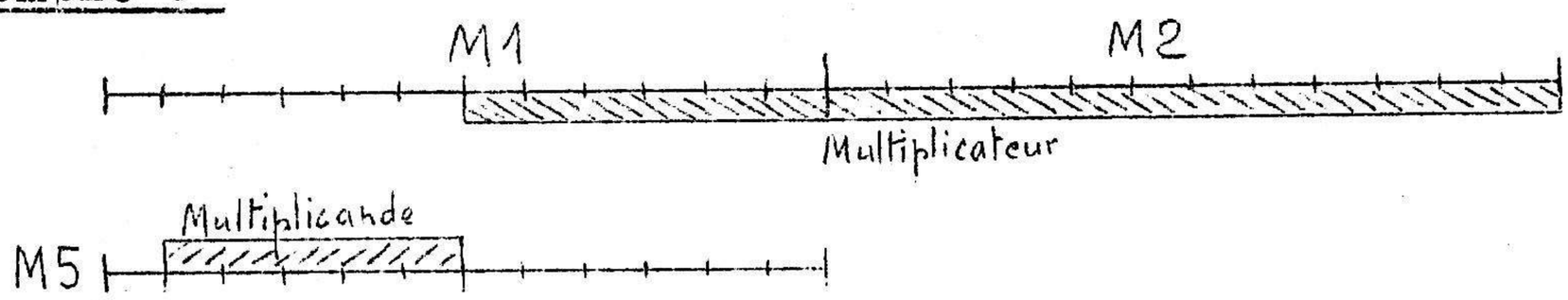
Multiplication par un multiplicateur de plus de 12 chiffres.

Il peut arriver que l'on ait à effectuer des multiplications par un terme de plus de 12 chiffres, le produit ne dépassant pas 23 chiffres.

La MC ne peut traiter, en une opération, de multiplicateur de plus de 12 chiffres.

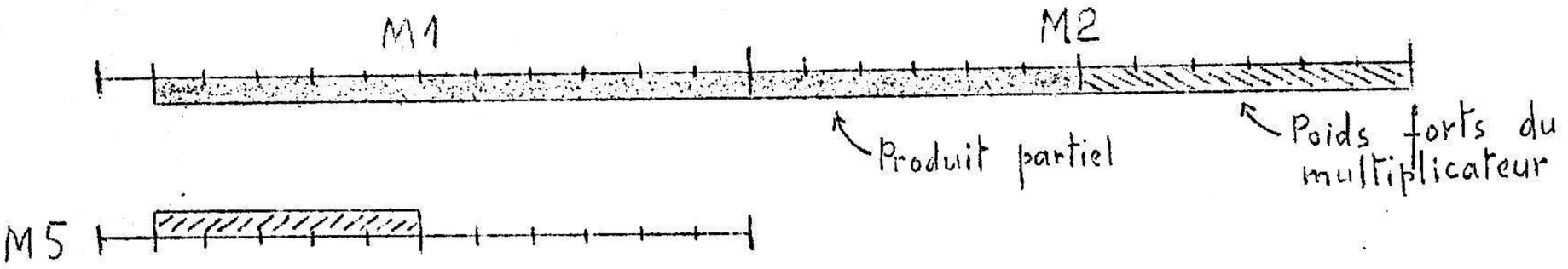
On peut traiter le problème en deux MC par manière de multiplications successives sans effacement de la M 1.

Exemple :



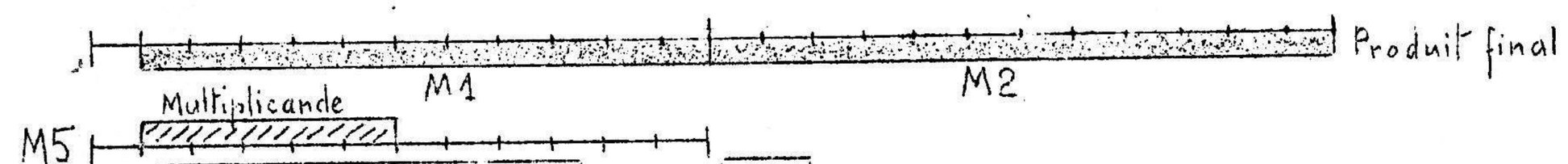
1ère multiplication : multiplicateur de 12 chiffres (les 12 poids faibles du multiplicateur)

MD = 0 par AMD



2ème multiplication : multiplicateur de 6 chiffres (les 6 derniers poids forts du multiplicateur).

MD = 6 par AMD sans effacement du produit



TO	AD	OD	OF	MD
7	–	–	–	0
14	5	6	11	0
7	–	–	6	6
14	5	6	11	0

DIVISION . -

La division est effectuée par une succession de soustractions. Le signe est traité à part.

Il existe en réalité deux divisions différentes par les capacités mises en jeu.

DIVISION REDUITE DR. TO = 13.

Un exemple fera comprendre le mécanisme de cette opération.

Dividende 6 9 3 4 6 5 4 M1

Diviseur 2 3 2 MB

									MD
			9	3	4	6	5	4	4
		9	3	4	6	5	4	0	3
		7	0	2	6	5	4	1	3
		4	7	0	6	5	4	2	3
		2	3	8	6	5	4	3	3
		0	0	6	6	5	4	4	3
		0	6	6	5	4	4	0	2
		6	6	5	4	4	0	0	1
		4	3	3	4	4	0	1	1
		2	0	1	4	4	0	2	1
	2	0	1	4	4	0	2	0	0
	1	7	8	2	4	0	2	1	0
	1	5	5	0	4	0	2	2	0
	1	3	1	8	4	0	2	3	0
	1	0	8	6	4	0	2	4	0
		8	5	4	4	0	2	5	0
		6	2	2	4	0	2	6	0
		3	9	0	4	0	2	7	0
		1	5	8	4	0	2	8	0

Reste Quotient

On fait donc une succession de soustractions (arrêtées lorsqu'elles deviennent impossibles) et de décalages.

MD initiale = OD du diviseur

On remarque qu'au cours des décalages, le reste partiel peut dépasser de 1 position la gauche du diviseur.

Cette condition est certainement vérifiée lorsque OF du diviseur (abstraction faite du signe) est $\leqslant 11$.

On s'imposera d'une manière permanente cette condition.

Difficulté de la division.

La division présente une difficulté que nous allons maintenant étudier.

L'opération commence par un décalage car il faut avoir à droite une place pour le 1er chiffre du quotient. Il en résulta qu'au départ les nombres doivent être positionnés pour que la soustraction soit impossible.

Si cette condition n'est pas satisfaite :

- la division durera longtemps,
- on perdra un ou, plusieurs chiffres en tête du quotient.

Or cette condition ne peut être exprimée en OD et OF. Elle nécessite une connaissance effective des nombres mis en jeu.

On devra donc :

- ou bien savoir (pour des raisons physiques par exemple) que le diviseur est au moins égal à une certaine quantité,
- ou bien faire avant la division une comparaison pour s'assurer que la première soustraction est impossible.

Division par zéro.

Elle se présente comme cas limite de la difficulté précédente. Dans ce cas, la soustraction se poursuit indéfiniment sans jamais rencontrer d'impossibilité. La machine se bloque sur la ligne de programme relative à la division. Il ne reste qu'à arrêter la machine et recommencer le calcul.

Interprétation du résultat.

Le résultat s'interprête en nombres entiers comme le quotient du dividende écrit avec ses unités en (O-1) par le diviseur écrit avec ses unités en (OD, OD+1).

On peut donner au quotient OD = O, OF = OD du diviseur. On trouve finalement la position de la virgule du quotient quand on connait celle des facteurs.

Remarques :

Le reste a toujours le signe du quotient. Son OD est celui du diviseur. A la fin de l'opération MD = O.

DR avec AD = O.

Cette division obéit aux règles habituelles.

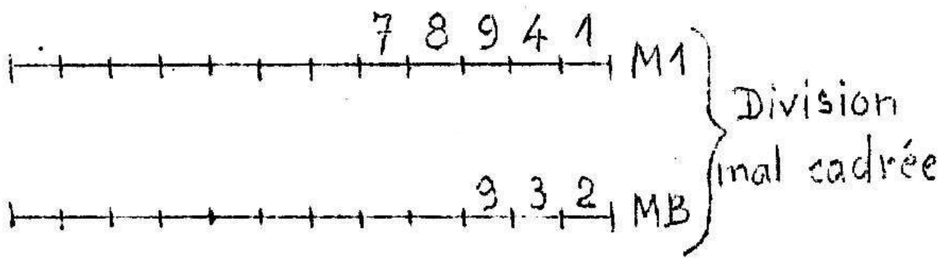
Le diviseur doit être non nul (OF $\neq$ O), positionné au plus en 10 (OD $\leqslant$ 10) et la soustraction doit être impossible au départ.

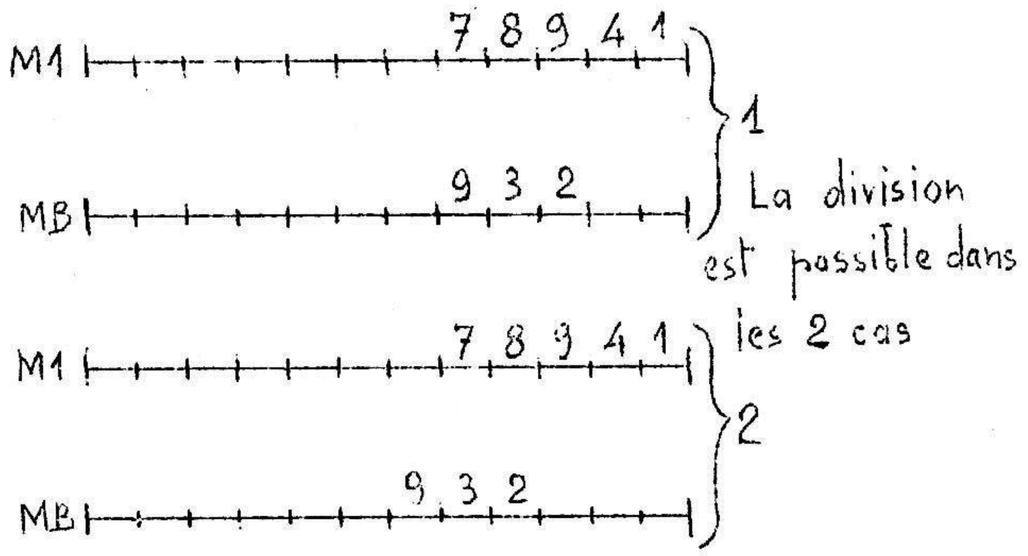
Règle d'opération.

1) Placer le dividende A en M 1, le rang des unités est en O de M 1.

2) Placer en MB le diviseur B de sorte que la soustraction des nombres A et B sans décalage donne un résultat négatif (ou que la soustraction arithmétique soit impossible du contenu de M 1 moins le contenu de MB éventuellement complété par des zéros).

Exemple.

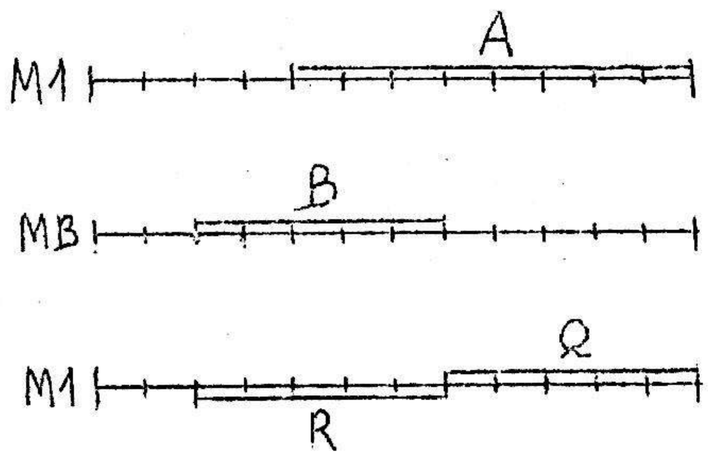




3) Commander la division

4) a) Le prenier chiffre du reste R est en OD du diviseur

b) Le quotient est écrit de 0 à OD du diviseur.



Remarque : La condition du 2) assure que le quotient a assez de place pour être contenu entre 0 et OD.

DIVISION COMPLETE DC. TO = 15.

Cette division utilise les mémoires 1 et 2 couplées en une seule mémoire à 24 positions.

Un exemple fera comprendre son fonctionnement. (page 42)

Exemple de division complète.

TO	AD	OD	OF
7	-	-	8
15	7	3	7

MD
8
0

On positionne MD = 8

Sur cet exemple on constate :

a) que les nombres doivent être positionnés de manière que la 1ère sous traction soit impossible. Cela provoque les mêmes difficultés qu'en division réduite.

b) que l'OF du diviseur compte non tenu du signe, ne peut dépasser 11.

c) que le nombre de décalages est déterminé par le contenu initial de MD qu'il fait positionner. Ce positionnement peut être 1,2,....,12 (pour positionner à 12, on affiche 0).

Le résultat s'interprête donc comme la division en nombres entiers du dividende supposé écrit avec unités en (0-1) de M 2 par le diviseur écrit avec unités en δ de M 2.

Remarques :

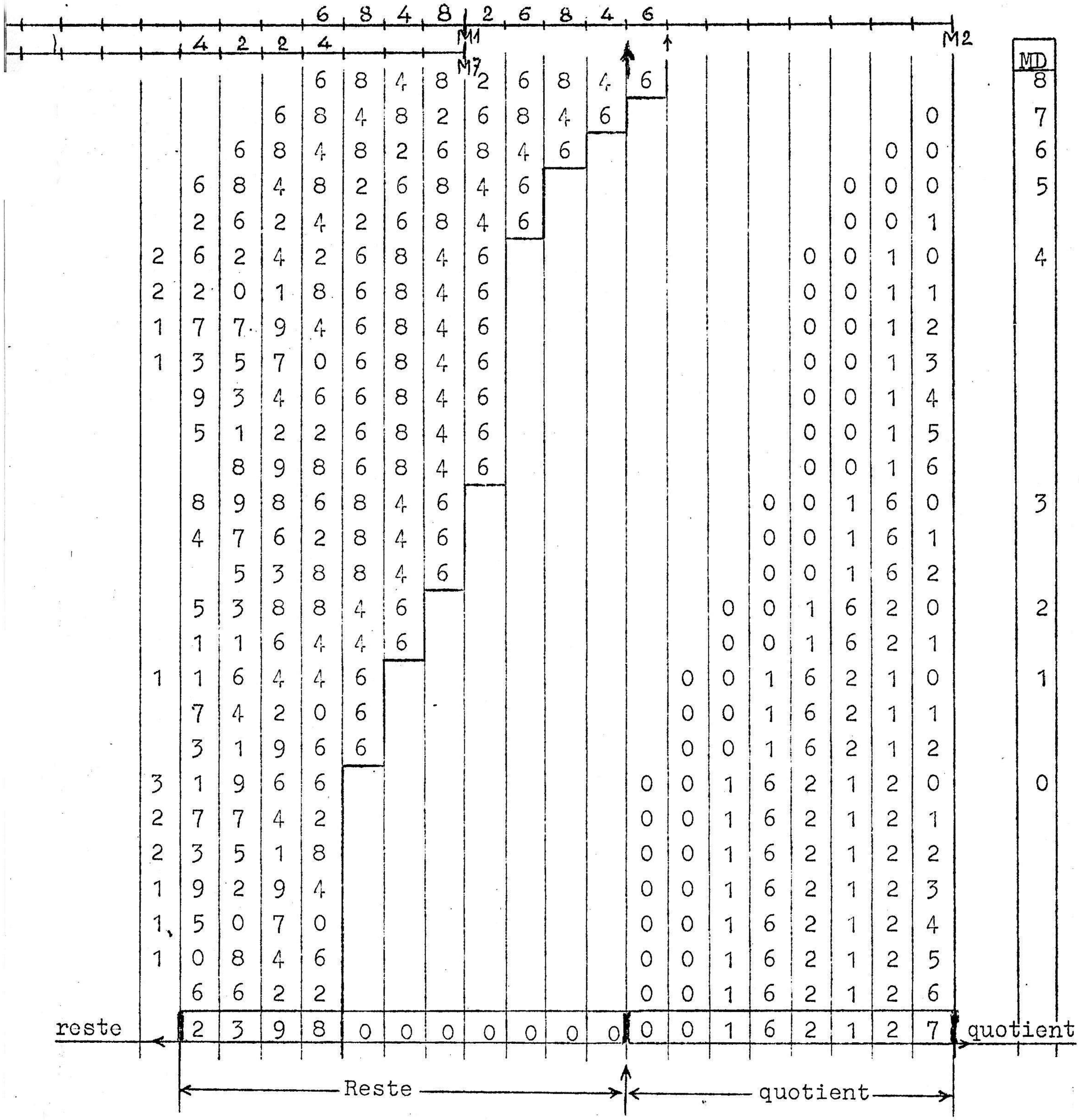
A la fin de l'opération , MD = 0.

A la fin de l'opération, le quotient est dans M 2, mais son signe est en MS. Le regroupement peut se faire en M 2 par la séquence

TO	AD	OD	OF
7	-	-	11
8	2	11	-

ou en M 1 par une division complète de cadrage que nous étudierons plus loin.

EXEMPLE DE DIVISION COMPLETE

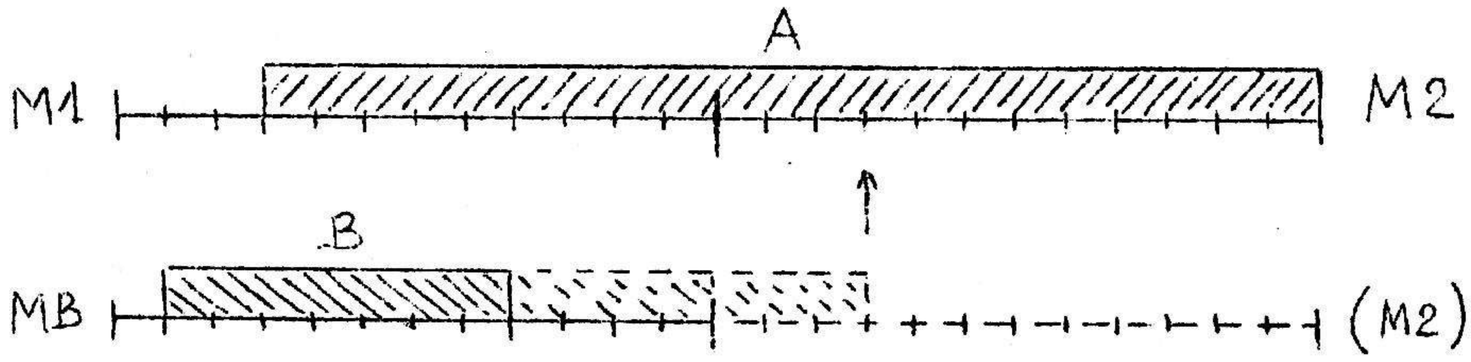


- le quotient occupe les 8 premières positions
- le reste, les 11 positions suivantes.

Règles de la division complète :

1) Placer le dividende A en M 1- M 2 réunis, la position des unités est en position 0 de M 2 (A peut avoir 23 chiffres).

2) Supposons la mémoire décalage à ∂



Le diviseur s'interprète comme un nombre ayant ses unités en ∂ de M 2 et par conséquent pour que la division soit possible il faut que les positions de B et A soient telles que la soustraction entre la partie de A en M 1 en correspondance avec B ne soit pas possible arithmétiquement.

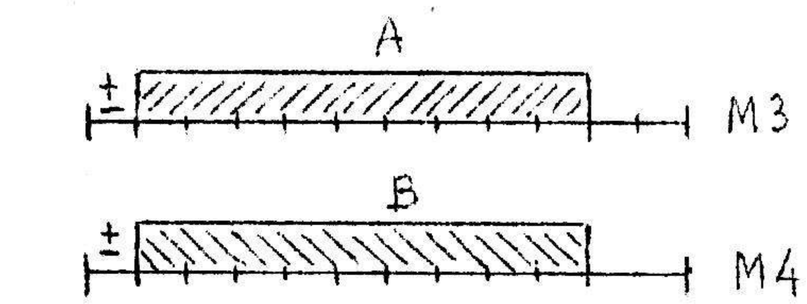
3) Commander la division

4) a) Le quotient a ses unités en 0 de M 2 et est écrit entre 0 et ∂

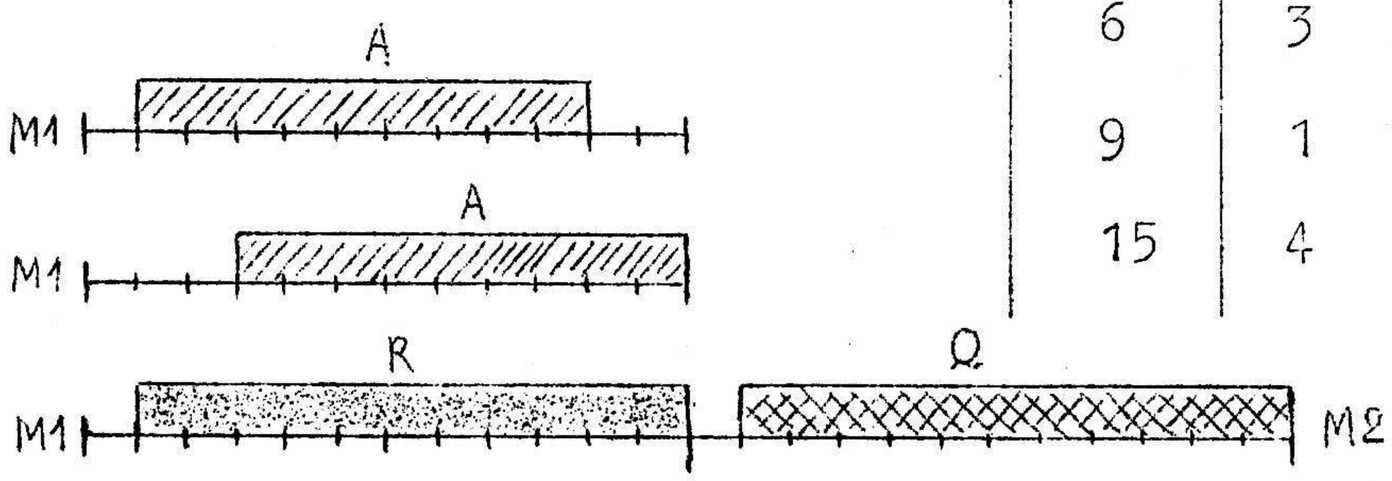
b) Le reste a ses unités en ∂

Problème :

Soit deux nombres A et B en M 3 et M 4, on veut effectuer A / B de telle sorte que le quotient Q soit en M 2 et le reste en M 1



TO	AD	OD	OF
3	2	-	-
6	3	2	-
9	1	-	-
15	4	2	-



DC avec AD = 0 15 - c d

C'est une division par le code OF qui obéit à toutes les règles classiques (en particulier, la 1ère soustraction doit être impossible et OD $\leqslant 10$).

La constante ne doit être ni 0, ni 1.

Pour 0, la raison est évidente.

Pour 1, il s'agit d'une opération sans intérêt que l'on a remplacé par la division complète de cadrage.

DIVISION COMPLETE DE CADRAGE DCC. 15 - c 1

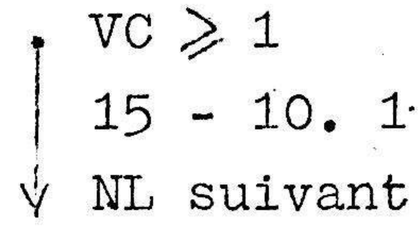
Cette opération suppose la MD positionnée à une valeur δ.

On effectue des décalages jusqu'à ce que la tête du nombre vienne en OD ou jusqu'à ce que MD = 0.

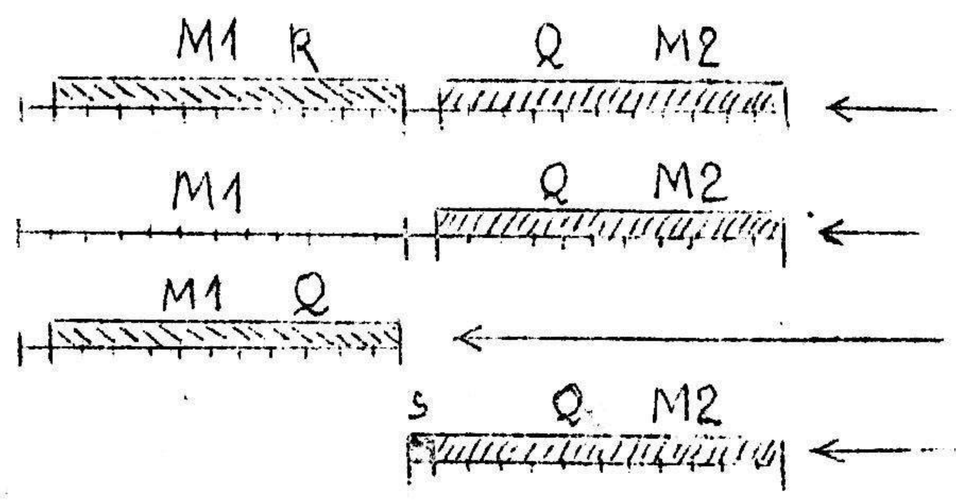
A chaque décalage vers la gauche, la MD est décomptée de 1 unité.

Avant chaque décomptage, le contenu de MD est transféré vers la première position de M 2. Si δ est le contenu initial de MD, après d décalages, le contenu de MD est δ-d et celui de la première position de M 2 = $\left[\delta - (d-1)\right]$ (opération utilisée en P.D.F.)

Remarque : Il est nécessaire qu'il y ait au moins un décalage car il s'agit d'une division, l'opération commence donc par un décalage vers la gauche. On devra donc ou positionner le nombre tel qu'il y ait au moins un décalage (tableau P.D.F. page 78) ; ou tester la présence d'un chiffre en position 10-11 de M 1 avant l'opération, par exemple : 9 - 10 . 1



Exemple : Regroupement du quotient et de son signe en M 2 en utilisant une DCC.

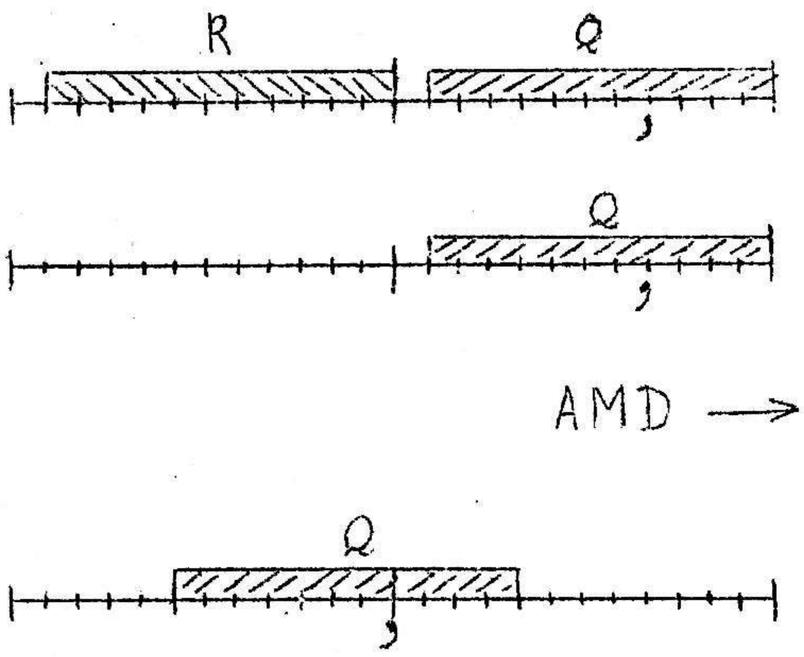


TO	AD	OD	OF
15	B	-	-
3	1	-	-
15	-	10	1
8	2	-	-

MD
0
0
0
0

Si le quotient possède une virgule et que l'on désire n'amener en M 1 que les chiffres à gauche de la virgule, on fera en sorte de positionner la MD pour qu'elle n'amène que cette partie en M 1.

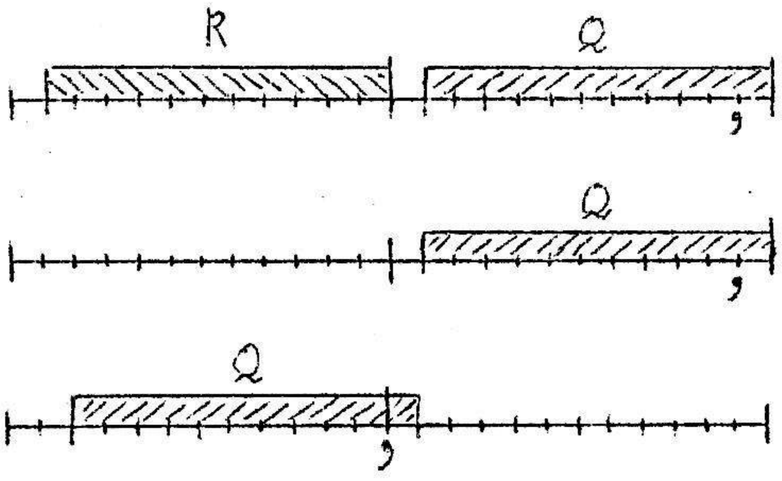
Exemple :



TO	AD	OD	OF	MD
15	B	-	-	0
3	1	-	-	0
7	-	-	8	8
15	-	10	1	0

Dans certains cas, on peut employer une BO qui remplace la ZB suivie de l'AMD.

Exemple :



TO	AD	OD	OF	MD
15	B	-	-	0
6	1	11	-	11
15	-	10	1	0

C H A P I T R E IV

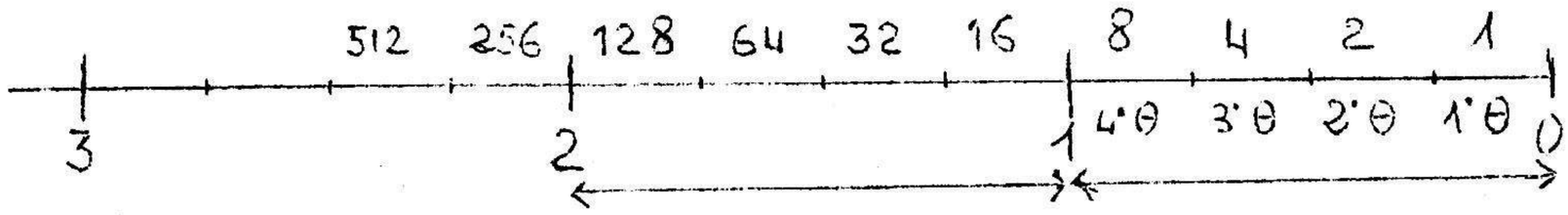
CALCUL BINAIRE - APPLICATIONS

En calcul binaire les mémoires Gamma ne sont plus des mémoires à 12 positions décimales mais des mémoires à 48 positions binaires.

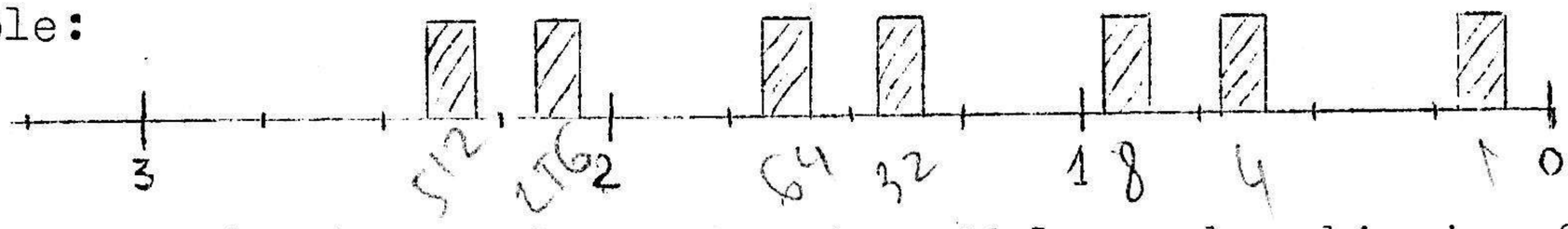
Une position décimale est désignée par τ et une position binaire par θ . $1\tau = 4\theta$

Le code de numération utilisée dans le Gamma (décimal codé binaire) permet d'utiliser complètement les 4 positions binaires de chaque τ (de poids 1-2-4-8) puisque chaque τ peut enregistrer des codes de 0 à 15.

Lorsque la M 1 est utilisée en calcul binaire les poids binaires qu'elle contient sont les puissances successives de 2.



Par exemple:



représente sur les 3 premiers τ le nombre binaire 877

- Les mémoires Gamma sont adressables position décimale par position décimale au moyen des codes OD et OF des instructions.

L'adressage d'une position binaire n'est pas possible directement et dans le cas où cet adressage est nécessaire il faudra recourir à des procédés indirects que nous préciserons.

- Lorsque la mémoire opérateur fonctionne en calcul binaire le report dans l'addition, d'une position décimale à l'autre, n'est plus émis à 10 mais à 16, ce qui permet d'effectuer l'addition de deux nombres binaires avec résultat écrit en binaire.

Dans les opérations de transfert entre la M 1 et les MB, les codes binaires sont transférés sans altération (En CD, si on appelle en M 1 une position de mémoire contenant un code binaire $\geqslant 9$, ce code est transformé et se trouve écrit en décimal sur 2 positions consécutives (cf. pages 8 et 11).

Emploi du calcul binaire -

Le calcul d'instructions necessite des opérations sur des codes binaires représentant les diverses parties d'une instruction élémentaire.

Ces codes binaires sont des nombres binaires essentiellement positifs, la notion du signe pour une instruction n'ayant pas de sens sur Gamma.

L'emploi du calcul binaire est différent suivant que l'on traite des nombres binaires toujours positifs ou de nombres binaires quelconques.

Nous allons étudier séparément le cas des opérations sur instruction et le cas des opérations sur des nombres binaires de signe quelconque.

1.- OPERATIONS SUR INSTRUCTIONS ET CALCUL BINAIRE -

Nous allons passer en revue les opérations élémentaires en indiquant les changements qui se produisent lors de leur emploi en cabul binaire portant sur des nombres toujours positifs.

ZB - sans changement

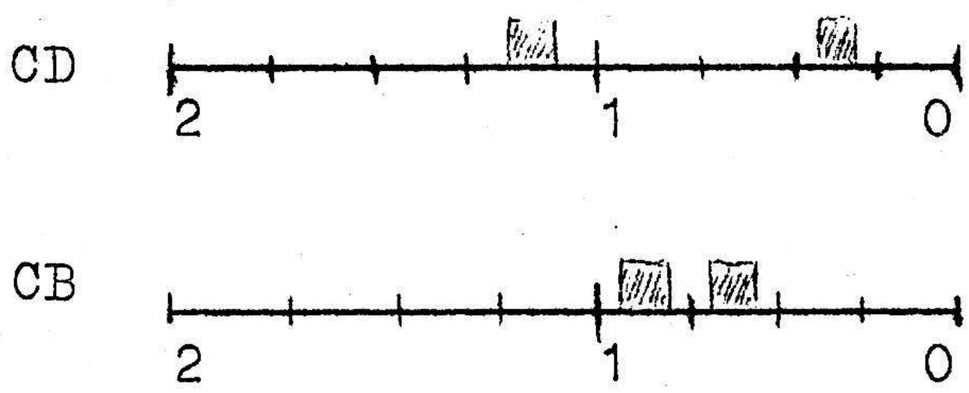
KB Permet d'introduire des constantes positives en M1 et MB sans restrictions.

BO
Transfert de codes binaires sans altération
OB

CN Permet de comparer deux ensembles de codes binaires

AN Effectue la somme binaire de deux codes ou deux nombres binaires positifs.

Exemple: 8 + 4 = 12 avec les écritures suivantes



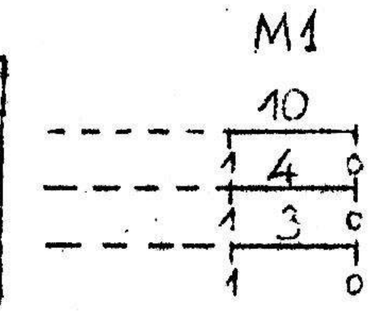
SN Permet de soustraire deux nombres binaires positifs. Le résultat doit être un nombre positif, sinon il y a redressement binaire.

Redressement binaire. -
Soit deux nombres binaires A et B avec B > A, le résultat de la soustraction binaire de A et B est R = |A-B|

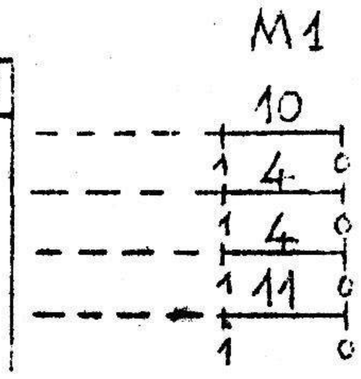
Ce redressement binaire subsiste à la suite d'une nouvelle opération arithmétique (addition ou soustraction) à condition d'effacer la MS1, au moyen de l'opération 12.1.-.-. ou par transfert du nombre, vers une mémoire banale et rappel en mémoire opérateur.
(Par exemple 8.2.-.-. suivi de 6.2.-.-.
En effet, l'opérateur de signes n'est pas invalide en CB, on supprime seulement le transfert signe (dans l'opération OB), mais le signe reste en MS1

Ex.

TO	AD	OD	OF	MS1
6.	-.	-.	10	+
11.	-.	-.	14	-
10.	-.	-.	7	+



	MS1
6.-.-.10	+
11.-.-.14	-
12.1.-.-	+
10.-.- 7	+

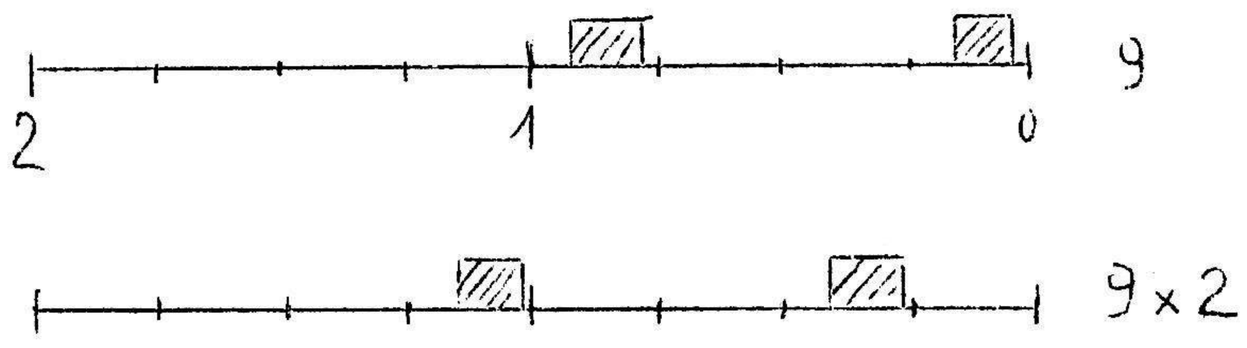


MR et MC Permettant d'effectuer le produit de 2 nombres binaires positifs

Multiplication par 2^n -

La multiplication par 2 d'un nombre écrit en M 1 décale ce nombre de 1 position binaire vers la gauche (en numération décimale le produit par 10 décale de 1 position décimale vers la gauche)

Exemple :



La multiplication par 2^n décale le nombre contenu en M 1 de n θ vers la gauche.

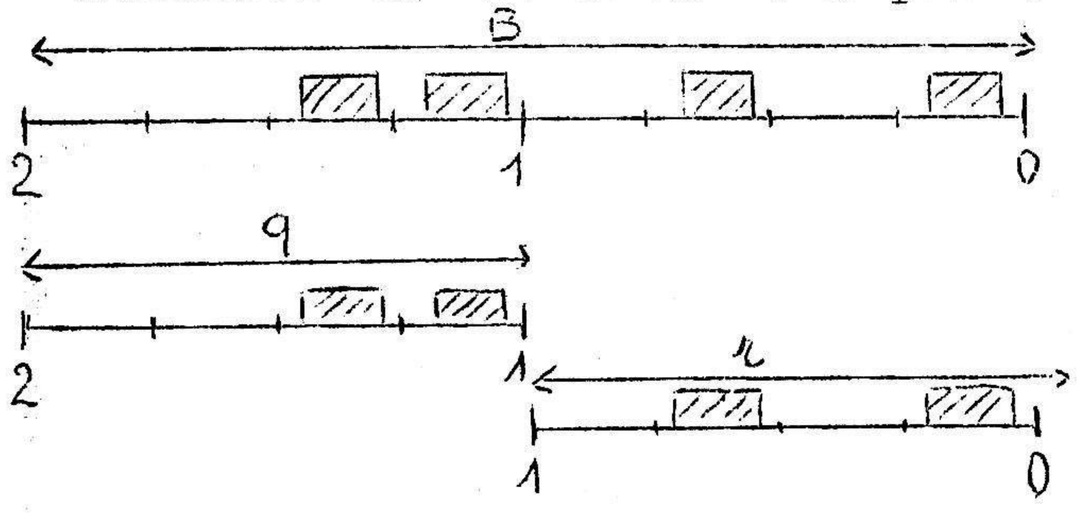
DR et DC

Propriétés symétriques de celle de la division

Division par 2^n -

La division par 2^n décale le nombre contenu en M 1 de n θ vers la droite.

Remarque Etant donné un nombre binaire B, nous aurons souvent à utiliser la division de B par 16 ; B = 16 q + r

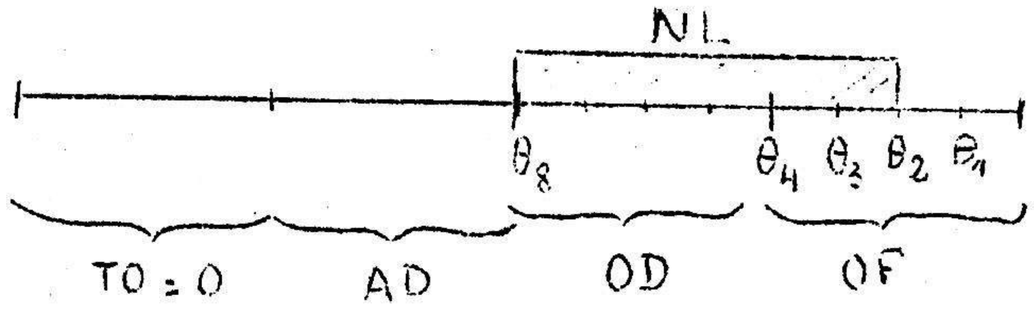


Si le nombre B est écrit sur 2 positions décimales, le reste est sur les 4 premiers θ et le quotient sur les 6 suivants

Ce résultat est valable dans toute division d'un nombre binaire par un multiple de 2.

Applications - Structure des instructions utilisant le tableau de code

1. Variantes



TO = 0

AD = 0 à 15 (0= variantes systématiques, 1-2-3 = comparaisons , 4 = signe).

Les 2 derniers θ de l'OF (θ_3 - θ_4) et les 4 θ de l'OD permettent d'écrire en binaire le NL variant de 0 à 63 ($63 = 2^6 - 1$)

$\theta_1 = 0$ si variante ordinaire (jamais, $>$, $=$, $\geqslant$, MS^-

$\theta_1 = 1$ si variante barre (toujours, $\leqslant$, $\neq$, $<$, MS^+

$\theta_2 = 0$ pour variantes systématiques, comparaison, signe et VPM 8.3 à 15.3

$\theta_2 = 1$ pour variantes à relais et $\overline{VAC}$ - $\overline{VAC}$ 0 à 6

Pour les variantes classiques (θ_2=0), on aura les règles suivantes:

- Désignons par mot le contenu d'une mémoire contenant des instructions et syllabe l'ensemble des 4 codes constituant une instruction il y a 3 syllabes par mot numérotées 0-1-2, il y a 16 mots par série de programme numérotés de 0 à 15
(sur le tableau, on aurait 15 mots de 4 syllabes 0-1-2-3)

OD = n° du mot	OD = M	$0 \leqslant M \leqslant 15$

OF = 4 x n° de syllabe	si variante ordinaire	OF = 4s $0 \leqslant s \leqslant 2$

OF = 4 x n° de syllabe + 1 si variante barre

AD = 0 jamais
= 1 $>$
= 2 $=$
= 3 $\geqslant$
= 4 signe

Ceci permet de coder directement les variantes sur la feuille de programmation - (Pour les mémoires de l'octade paire, on retranche 8 au n° de mémoire pour avoir le n° de mot).

Exemple 1 - Variante toujours en NL 57 (- . - . 14.5)

AD = 0 (variantes systématiques)

OD = 14 (N° de mémoire M=14)

OF = 5 (N° de syllabe s=1, +1 pour VS toujours)

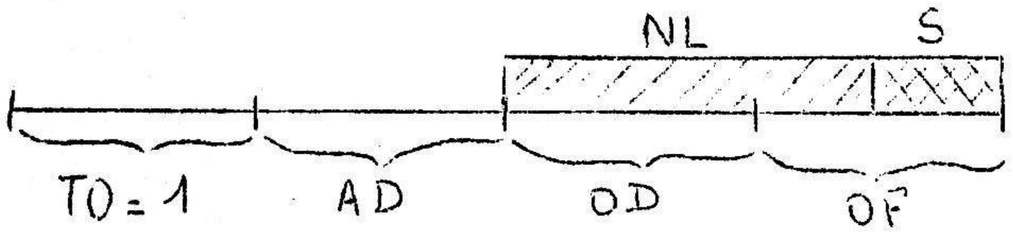
2- Variante égale en NL 22 (- . 2.5. 8)

AD = 2

OD = 5

OF = 8

2- Variantes changement de série -



S = n° de série 0-1-2-3

OD = n° de mot

OF = 4 x n° de syllabe + S

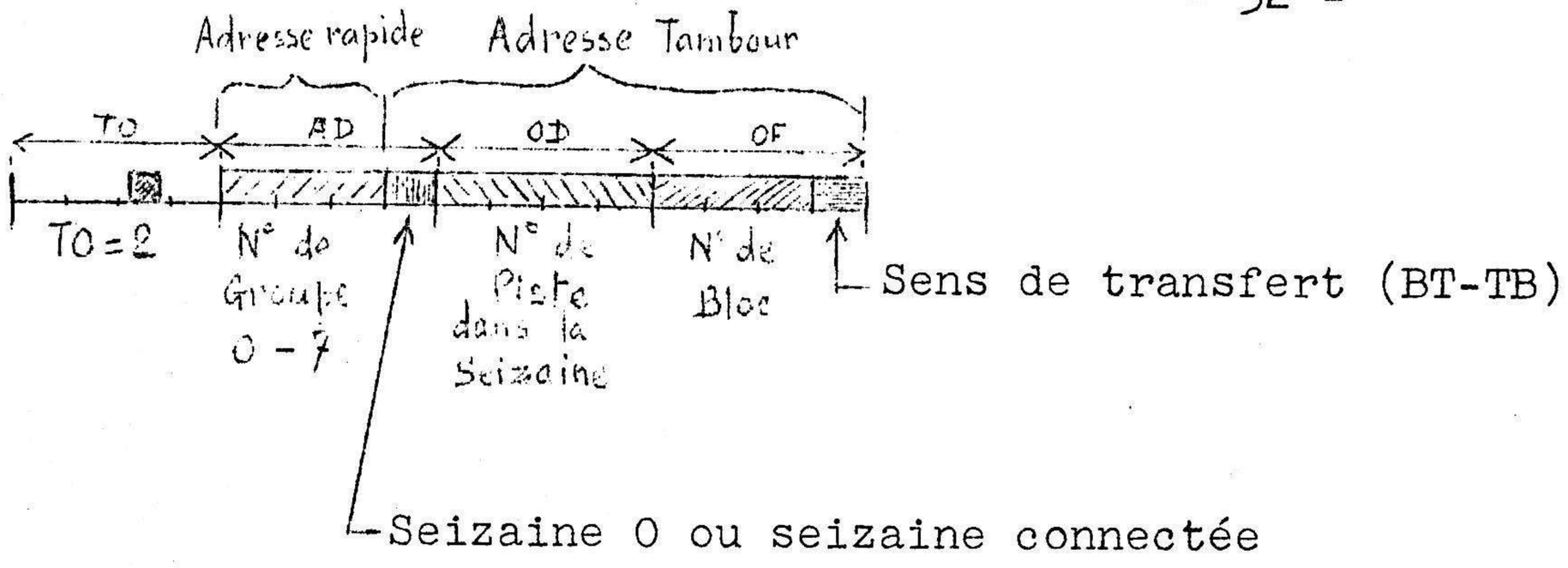
Exemple

NL 45.S2

OD = 11

OF = 6

3- Transferts tambour -



Dans le premier θ de AD : rien si seizaine 0
1 si seizaine commutée

Dans le premier θ de OF : rien si BT
1 si TB

Soit G le N° de groupe de MR (0 - 1 - 2 - 3)
P le N° de piste dans la seizaine (0 - 15)
B le N° de bloc dans la piste (0 - 7)

AD = 2G pour seizaine 0
AD = 2G+1 pour seizaine commutée

OD = P

OF = 2B si BT
OF = 2B+1 si TB

<u>Exemple</u> : Sz 3 P 12 B 7 ⇒ G 2

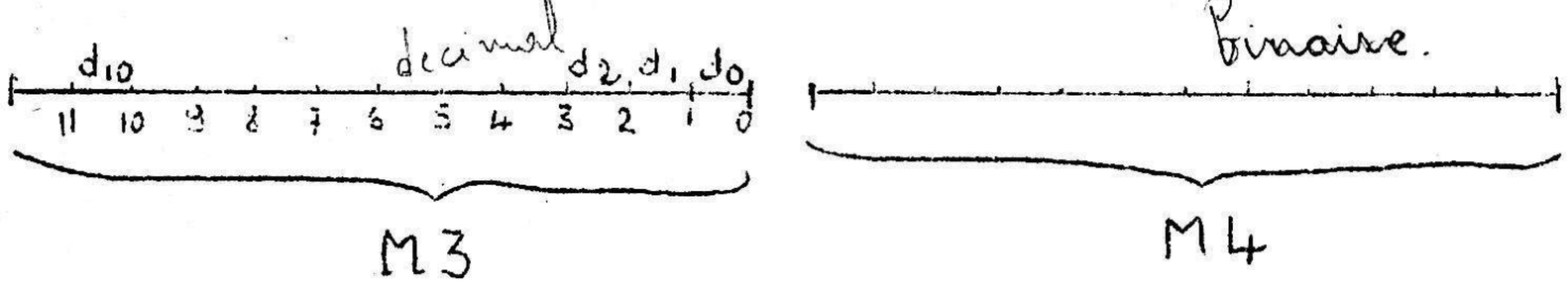
AD = 5
OD = 12
OF = 15

→ {1 . 13 . - . 3
{2 . 5 . 12 . 15

Transformation d'un nombre écrit en décimal en binaire et réciproquement.

Le nombre écrit en décimal est en M 3 }
Le nombre écrit en binaire est en M 4 } dans les deux cas

1. Transformation Décimal -Binaire-



Le nombre décimal occupe 11 chiffres au plus en M 3.
On utilise un schéma de Hörner.

$$\left[(10\ d_{10} + d_9)\ 10 + d_8\right]\ 10 + d_7 \ldots\ldots\ldots$$

Le nombre décimal en M 3 est d'abord transféré en M 4 qui par par la suite contient les résultats intermédiaires du calcul du polynôme.

Le nombre décimal est conservé en M 3.

2. Transformation Binaire - Décimal-

L'écriture binaire pure étant plus condensée que l'écriture décimale codée binaire on ne considerera que les nombres binaires inférieurs au nombre binaire correspondant à un nombre décimal de 11 chiffres.

$2^{36} = 68.\ 719.\ 476.\ 736$ et $2^{37} = 137.\ 438.\ 953.\ 472$

On prendra donc 36 positions binaires, soit 9 positions décimales au maximum pour le nombre binaire.

Les digits binaires sont traités par paquets de 4, le nombre binaire correspondant à 1 position décimale est transformé en décimal par l'instruction CD (1. 10. - . -) . On utilise le schéma de Hörner.

$$(16\ d_8 + d_7)\ 16 + d_6 \ldots\ldots$$

d_0 est traité à part

DÉVELOPPEMENT E.T. Problème : Conversion D ⇒ B et B ⇒ D

Seizaine: 0 Piste: 15 Bloc: 7 Série: 1

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération CD/CB	MD	M1	M2	MB
											0 11 10 9 8 7 6 5 4 3 2 1	0 11 10 9 8 7 6 5 4 3 2 1	0 10 8 6 4 2 0
2 2 15 15			Octade: 2										
1 2 - 1	D ⇒ B	0	1	15			8					M3 6871947673,6	
		1	6	3									
		2	8	4							M4		
		4	6	=	=	10	9						
		5	12	4	10	11						1	
		6	10	3	9	10							
		8	8	4	9		10						
		9	1	12		2							
		10	6	10	1	2			D(M3) ⇒ B(M4)				
		12	8	9	8	9	11						
		13	8	9	5	6							
		14	9	-	5	-							
		16	11		5	1	12						
		17	8	10	1	2							
		18	8	9	9	10							
		20		2	1	1	13						
		21	1	6					V ≠				
		22											
		24					14						
		25											
		26											
		28					15						
		29											
		30											
2 2 15 15			Octade 3										
1 2 8 1	B ⇒ D	32	3	3			8						
		33	1	15									
		34	6	4	8	9							
		36	1	10			9						
		37	10	3	-	-							
		38	9	1	1	-							
		40	10	-	-	4	10						
		41	12	1	1								
		42	8	3									
		44	1	12		3	11						
		45	6	8	9	10							
		46	9		9	1			B(M4) ⇒ D(M3)				
		48	8	8	8	9	12						
		49	11		8	1							
		50	8	8	9	10							
		52		2	8	5	13						
		53	1	15					V ≠				
		54	6	4		1							
		56	1	10			14						
		57	10	3									
		58	8	3									
		60	1	6			15						
		61											
		62	X										

0 11 10 9 8 7 6 5 4 3 2 1 0 11 10 9 8 7 6 5 4 3 2 1 0 10 8 6 4 2 0

ORDRES INITIAUX. -

Exploitation d'une carte chercheuse.

La lecture de la carte chercheuse permet de dérouler en série 3 deux lignes de programme qui alimentent le premier bloc de la piste 0 en série 0 et s'y renvoient (TB - VCS).

Simultanément, ont été introduites en M 615 les indications correspondantes portées sur la carte (S - NL - Sz - P - B) : adresse de la première instruction à exécuter écrite en décimal.

A partir de ces indications, il s'agit de coder une CZ, un TB et une VCS qui renvoient sur le programme à exécuter.

Pour coder la VCS, il suffit d'écrire en binaire le N° de ligne, décaler ensuite de 20 vers la gauche et d'y ajouter le N° de série (voir page 51).

Pour la CZ, il suffit de renvoyer le N° de seizaine en position OF de la CZ.

Les 3 codes AD, OD et OF du TB sont calculés globalement en tenant compte de l'écriture de l'instruction TT (voir page 52).

Les 3 lignes d'instructions codées (40 - 41 - 42) permettent d'alimenter le premier bloc du programme à exécuter en mémoires rapides et de s'y renvoyer.

DÉVELOPPEMENT E.T. Problème : Exploitation d'une carte chercheuse

Seizaine: 0 Piste: 0 Bloc: 0 Série: 0 (Ordres initiaux)

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD/CB	MD	M1	M2	MB
	Série 3	1	2	-	-	1	} Lignes série 3 - Cas d'une carte chercheuse							
		2	1	-	-	-								
		Octade: 0												
		0	1	15			D0		CB	B				
		1	6	3			8		BO			1 8 S_2 P BD		
		2	3	3					ZB					
		4	8	4					OB					
		5	1	12		6	9		CO					
		6	6	12					BO					
		8	8	5					OB					
		9			3	2	10		VR°			appelée par contrôle		
		10	6	15					BO					
		12	8	7					OB					5 NL5 P B
		13	5		8	3	11		GG			effacement de l'introduction		
		14	6			10			BO					
		16	12	7	6	7	D1		MR					
		17	10	7	5	6	12		AN					
		18	12		7	4			MR			Calcul de la VCS		
		20	7			5			AMD					
		21	10	7	7	8	13		AN					
		22	1	12		1			CO					
		24	8	10	8	10			OB					
		25	6	7	4	5	14		BO					
		26	8	10		1			OB					
		28	6		5	2			BO					
		29	12	7	7	8	15		MR					
		30	10		1	10			AN					
		Octade 1												
		32	12	7	3	4	D2		MR			Calcul du TB		
		33	10	7	1	3	8		AN					
		34	10	1	1	2			AN					
		36	10		1	1			AN					
		37	10		3	1	9		AN					
		38	8	10	4	7			OB					
		40	1	13					CS_2					
		41	2				10		TB			appel du programme à exécuter		
		42	1						VCS					
		44												
		45					11							
		46												
		48												
		49					12							
		50												
		52												
		53					13							
		54												
		56												
		57					14							
		58												
		60												
		61					15							
		62	X											

APPEL ET RENVOI D'UN TERME a_{ij} DEFINI PAR 2 INDICES. -

Supposons que l'on ait rangé sur le tambour une matrice à raison de 1 ligne par piste (on peut adopter d'autres modes de rangement en particulier le rangement par blocs sans tenir compte des pistes).
Si le rangement commence en piste 0 (seizaine 1) on a :

$i - 1 = P$
$j - 1 = 16\ B + M$

P N° de piste

B N° de bloc

M N° de mémoire dans le bloc (0 à 15)

le terme a_{11} est rangé en piste 0, bloc 0, M0 (M 8 de l'octade paire)

Pour appeler un terme en mémoire opérateur, on se donnera en M 7 (par exemple) les indices i et j en décimal, en positions 0-2, de même pour ranger un terme, on donne ses indices i et j en positions 2-4 ; la M 1 est utilisée comme pivot pour les transferts avec les mémoires rapides, on prendra le groupe 3 comme pivot pour les transferts avec le tambour.

Dans le cas d'un appel, la donnée des indices i et j doit permettre:le calcul d'une CZ, le calcul d'un TB alimentant le bloc contenant a_{ij} dans le groupe 3, le calcul de la CO et de l'AD en mémoire rapide permettant le transfert de a_{ij} en M 1.

Dans le cas d'un renvoi, la donnée des indices i et j doit permettre : le calcul d'une CZ, d'un BT qui range sur le tambour le contenu du groupe 3 (contenant le précédent terme à ranger), le calcul de la CZ et du TB alimentant le bloc devant recevoir a_{ij} en groupe 3 et le calcul de la CO et de l'AD permettant de transférer le terme contenu en M 1 dans le groupe 3.

On peut alors écrire un sous-programme qui permettra une programmation plus facile des petits problèmes matriciels (Au lieu de faire des comparaisons sur pistes, blocs, octades et mémoires, on effectuera des comparaisons et des progressions sur les indices i et j).

Dans le programme qui suit : i et j $\leqslant$ 16

Pour être utilisable dans les bonnes conditions la séquence doit être optimisée. On procède de la façon suivante :

Les indices i et j permettent de calculer les codes d'un BT (ceux du TB s'obtiennent en ajoutant 1) ainsi que la CO et l'AD.

Les codes du TB correspondant au dernier terme a_{ij} (appelé ou renvoyé) sont mis en réserve en ligne 62 et ceux du BT (obtenus par soustraction de 1) en ligne 42 ; pour le terme a_{ij} en cours on compare les codes calculés du nouveau TB aux codes mis en réserve en ligne 62 ; s'il y a identité il n'y a pas lieu de faire un TB, le bloc interessant le terme a_{ij} en cours étant le même que celui du terme précédent, on calcule alors seulement la CO et l'AD ; s'il y a différence on exécute un BT du G 3 sur le bloc concernant le terme précédent suivi du TB du bloc interessant le terme en cours (lignes 36, 37 et 38).

Rangement d'un a_{ij}

Ce sous-programme permet de lire des cartes contenant un terme a_{ij} dans la zone de M 613 et les indices i et j en colonnes 37-38 et 43-44 (Positions 6-8 et 0-2 de M 612). (i et j $\leqslant$ 16)

Le rangement s'effectue à raison de 1 ligne par piste, l'origine étant la Piste 0 Bloc 0 Mémoire 0 (M 8 Octade paire) de la seizaine 1.

Le rangement du terme contenu sur une carte s'effectue pendant la lecture de la carte suivante.

Le groupe 3 étant utilisé pour l'introduction, on utilise le groupe 2 comme pivot entre MR et Tambour.

Remarque : Ces séquences sont données à titre d'exemple ; elles ne permettent pas de résoudre tous les problèmes de manipulation de nombres ou de rangement, et sont destinées à fournir des procédés utilisables.

DÉVELOPPEMENT E.T. — Problème: Appel et renvoi d'un a_{ij}

Seizaine: 0 Piste: 9 Bloc: 6 Série: 1

2.2.9.13

Appel 1.2.–.1
Renvoi 1.2.1.1

Sélection lignes	NL	TO	AD	OD	OF		Opération
	Octade: 2						
Appel →	0	1	12		3	8	
	1	4	13	7	6		
	2			2	1		
Renvoi →	4	1	12		3	9	
	5	4	13	7	8		
	6	8	5				
	8	1	13		1	10	Fig 1
	9	1	15				
	10	6	9	4	7		
	12	8	15	4	7	11	TB
	13	11		4	1		BT
	14	8	9		3		
	16	6			10	12	Passage en binaire
	17	12	7	1	2		
	18	10	7		1		
	20	11			1	13	L – 1+7
	21	10		1	7		7
	22	8	9	5	7		
	24	6			10	14	
	25	12	7	3	4		
	26	10	7	2	3		
	28	11		2	1	15	
	29	8	15		1		
	30						
	Octade 3						
	32	6	15	4	7	8	
	33	9	9	4	7		
	34		2	10			=
	36	2	BT			9	BT
	37	2	7	–	1		
	38	1		10	1		
	40	6	15		1	10	
	41	9			8		
	42	4	13		7		
	44		3	12		11	≥
	45	4	13		6		
	46	10			8		
	48	8	13	6	7	12	
	49	1	10				
	50	6	5				
	52	1	12		6	13	
6 ou 8 ←	53	6	8				
	54	1	6				
	56					14	
	57						
	58						
	60					15	
	61						j en binaire
	62	X					

DÉVELOPPEMENT E.T. Problème : Rangement d'un a_{ij}

Seizaine: 0 Piste: 9 Bloc: 7 Série: 0

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD / CB	MD	M 1	M 3	MB
												0 11 10 9 8 7 6 5 4 3 2 1	0 11 10 9 8 7 6 5 4 3 2 1	0 10 8 6 4 2 0
		Octade: 0												
		0	1	2		3								
		1		11		8	8							
		2	1	12		6								
		4	6	12										
		5	3	12			9						i d	
		6	8	3										
		8	6	13										
		9	3	13			10							
		10	8	4										
		12	1	2		3								
		13	1	15			11							
		14	6			10								
		16	12	3	7	8								
		17	10	3	6	7	12							
		18	11		6	1								
		20	1	12		1								
		21	8	10	5	6	13							
		22	8	12	1	2								
		24	6			10								
		25	12	3	1	2	14							
		26	10	3		1								
		28	11			1								
		29	9			8	15							
		30												
		Octade 1												
		32	4	11		5								
		33		3	9	4	8							
		34	4	11		4								
		36	10			8								
		37	8	11	10	11	9							
		38												
		40	1	13		1								
		41	2	5	□	1	10							
		42	1		11									
		44	1	12		□								
		45	6	4			11							
		46	8	□										
		48	2	5	□									
		49	1			4	12							
		50												
		52												
		53					13							
		54												
		56												
		57					14							
		58												
		60												
		61					15							
		62	X											

0 11 10 9 8 7 6 5 4 3 2 1 0 11 10 9 8 7 6 5 4 3 2 1 0 10 8 6 4 2 0

M 1 M 2 MB

IL - Intersection logique

L'intersection logique de 2 nombres binaires est un troisième nombre binaire composé des poids communs aux deux premiers.

Exemple : L'intersection logique de 1 1 0 1 0 0 1 1
et de 0 1 1 0 1 0 1 0
est 0 1 0 0 0 0 1 0

Remarque importante : Les nombres décimaux étant écrits à l'intérieur de la machine en décimal codé binaire, on peut définir l'intersection logique de 2 nombres décimaux comme un nombre décimal composé des poids binaires communs aux 2 premiers dans chaque position décimale

Cette définition n'a de sens que pour les nombres positifs (codage du signe -)

- Il existe sur Gamma deux types d'intersection logique
  - intersection logique d'un nombre avec une constante
  - intersection logique de 2 nombres.

1- IL 1 - Intersection logique d'un nombre avec une constante

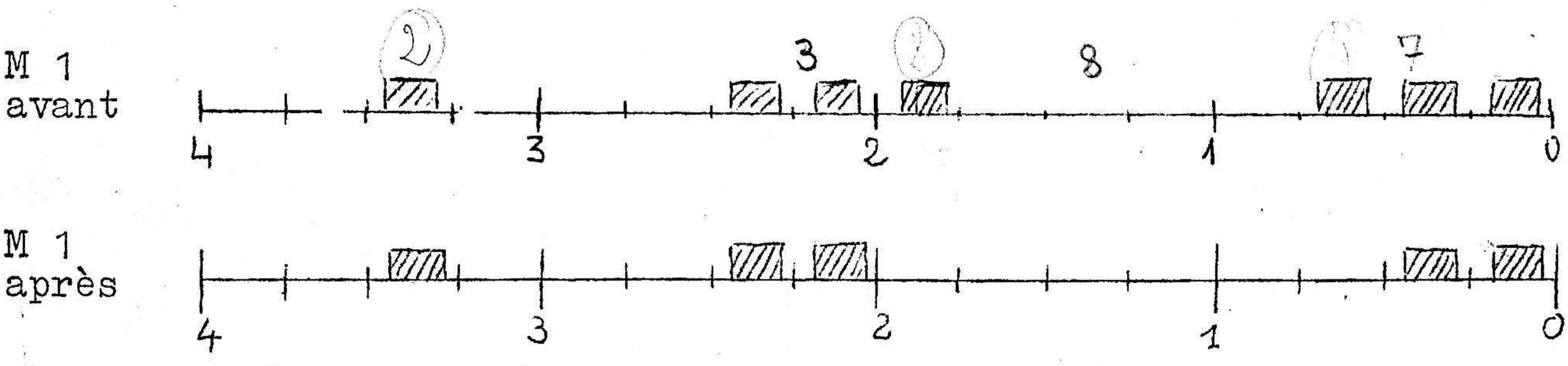
TO = 7 AD = 10 OD = 0 OF = valeur de la constante

Le nombre doit se trouver en M 1

La constante est définie par le code OF

Le résultat se trouve en M 1 à la place du nombre initial

Exemple : 7 . 10 . - . 3 avec en M 1 le nombre 2387



Le résultat est 2303

Le tableau suivant donne l'intersection logique des codes de 1 à 15.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Base
1		1		1		1		1		1		1		1	1
	2	2			2	2			2	2			2	2	2
		3		1	2	3		1	2	3		1	2	3	3
			4	4	4	4					4	4	4	4	4
				5	4	5		1		1	4	5	4	5	5
					6	6			2	2	4	4	6	6	6
						7		1	2	3	4	5	6	7	7
							8	8	8	8	8	8	8	8	8
								9	8	9	8	9	8	9	9
									10	10	8	8	10	10	10
										11	8	9	10	11	11
											12	12	12	12	12
												13	12	13	13
													14	14	14
														15	15

Cas particuliers

IL 1 avec OF = 0 : Remise à zéro de M 1

IL 1 avec OF = 15 : Maintien de M 1

2- IL 2 - Intersection logique de deux nombres

TO = 7 AD = 12 OD = OF = 0

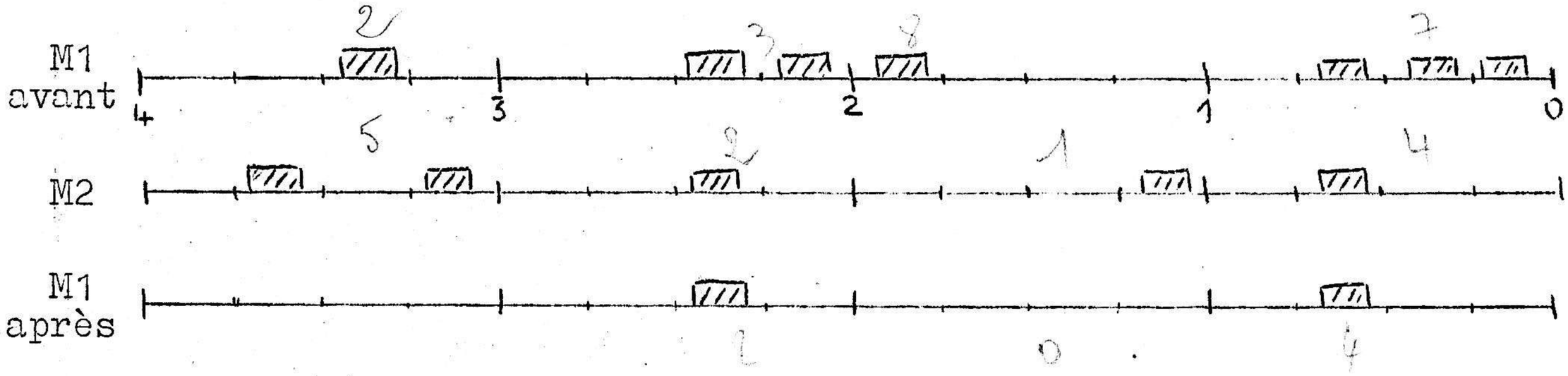
Les deux nombres sont placés en M 1 et M 2

On réalise l'intersection logique de M 1 avec le contenu de M 2 dans les positions homologues; l'intersection s'effectue

positon par position.

Le résultat se trouve en M 1 à la place du contenu précédent.

Exemple : 2387 en M 1 et 5214 en M 2

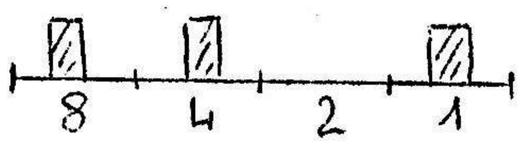


Le résultat est 0204

Remarque : Les opérations IL ne comportent pas de filtrage.

Applications de l'intersection logique -

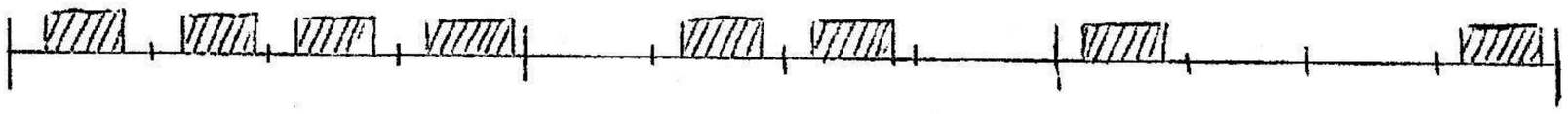
Considérons un code binaire écrit dans une position décimale : par exemple



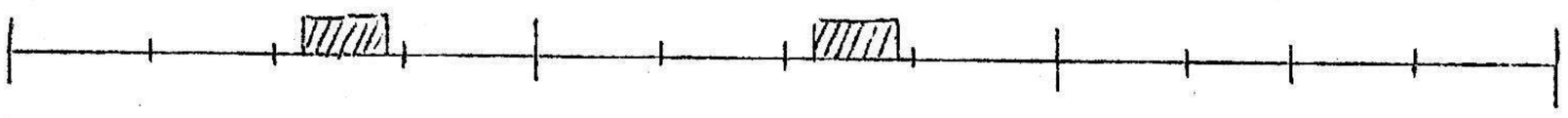
Si l'on veut isoler le poids binaire 4, on effectuera une intersection logique avec 4, on a ainsi la possibilité d'isoler un poids binaire parmi les 4.

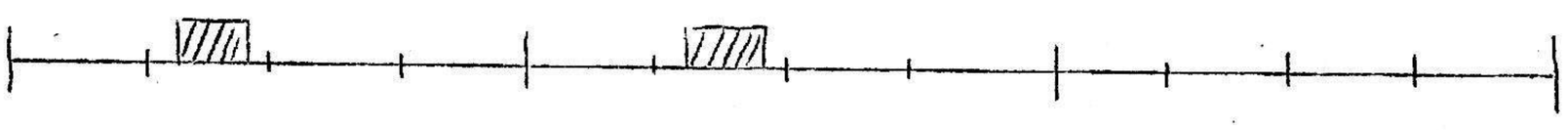
Décomposition en poids binaires

Etant donné un ensemble de codes binaires, par intersections logiques avec 1-2-4-8 on déduira 4 ensembles de codes binaires formés de codes binaires d'origine dans lesquels on ne conserve que le poids 1, le poids 2, le poids 4, le poids 8, dans chaque position de mémoire.



Exemple :

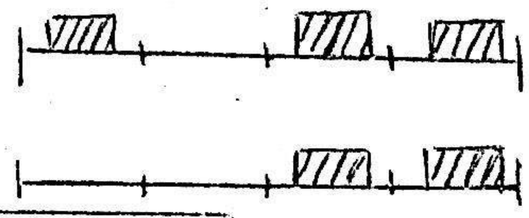




Suppression d'un poids binaire

Si, dans un code binaire, on veut supprimer le poids 8, on effectuera une intersection logique avec 7 (1-2-4).

C'est l'inverse de la décomposition en poids binaire.

Exemple 

Parité d'un nombre

L'intersection logique d'un nombre avec 1 donne 0 si le nombre est pair et 1 si le nombre est impair

Complément d'un code binaire

Pour prendre le complément d'un code binaire b écrit sur 1 position décimale, on effectuera une soustraction de 15, le redressement binaire après soustraction donnera comme résultat

$|b - 15|$ c'est-à-dire $15 - b$ car $b \leqslant 15$

UNION LOGIQUE DE 2 CODES BINAIRES -

On utilisera la propriété $A \cup B = C\left[C(A) \cap C(B)\right]$

ou $A + B = \overline{\overline{A} \cdot \overline{B}}$

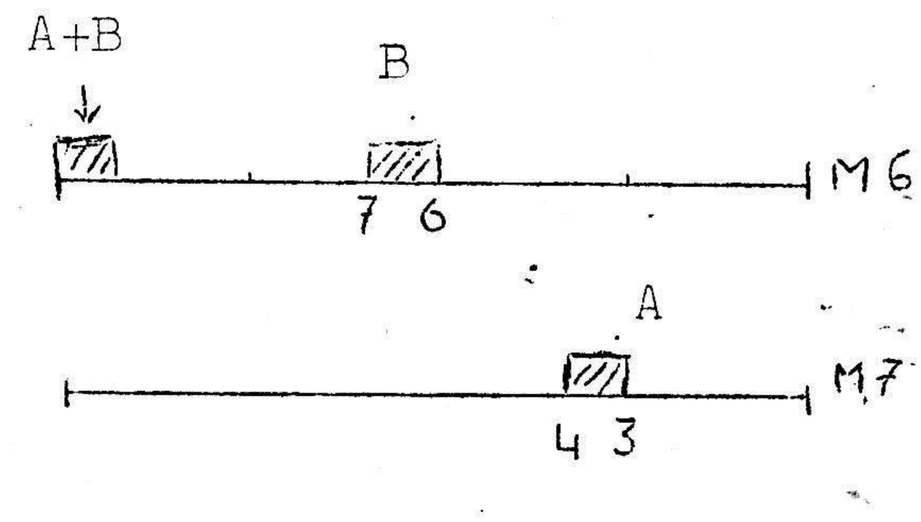
d'où la séquence

M 8
1. 15. - . -
6. 7. 3. 4
11. -. 3. 15 $\overline{A}$

M 9
8. 10. -. 1
6. 6. 6. 7
11. -. 6. 15 $\overline{B}$

M 10
7. 10. -[$\overline{A}$] $\overline{A}\cdot\overline{B}$
8. 2. -. -
6. 2. -. -
11. -. 6. 15 $\overline{\overline{A}.\overline{B}}$

8. 6. 11. -



Calcul de la CO et de l'AD pour un terme d'adresse N ($0 \leqslant N \leqslant 15$)

Le terme se trouve en Groupe 3 des MR (CO 6 et CO7) N est en position 0-1 de M 2.

1. 15. - -
6. 2. - 1
9. -. -. 7
4. 14. 8. 7

VC > (N > 7)

4. 14. 8. 6
10. -. -. 8

M 14
8. 15. 2. 3
1. 10. - -
1. 12 - □

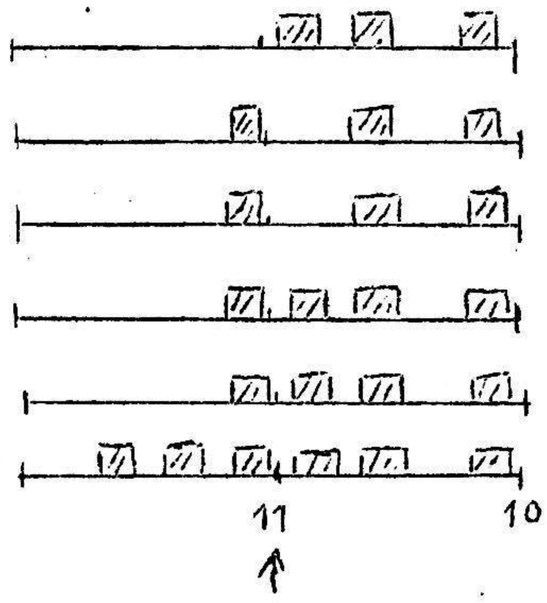
M 15
6. □. -. -

Séquence utilisée p.58 (Appel et renvoi a_{ij})

1. 15. - -
6. 2. - . 1
10. -. -. 8
7. 10. -. 7

10. -. -. 8
8. 14. 10. 11
10. -. 11. 6
7. -. -. 11
8. 14. 4. 5

M 14
1. 10. -. -
1. 12. -. □
6. □. -. -



Cette séquence par IL
- n'utilise pas de comparaison
- consiste à écrire en binaire l'adresse des mémoires rapides de 0 à 63

Transformation code binaire en code alphabétique

Les 4 codes binaires représentant une instruction sont enregistrés sur la machine à raison de 1 code par position de mémoire.

Lorsqu'on veut extraire un programme écrit sur le tambour, ou effectuer l'analyse d'un programme, il est nécessaire d'extraire les 4 codes de l'opération sur l'inprimante à raison de 1 code par position de mémoire.

Pour les TO et les AD compris entre 10 et 15 et les OD et OF compris entre 10 et 11, il est nécessaire de représenter ces codes par des lettre pouvant s'extraire sur 1 position.

A l'aide du code alphabétique BULL (1ère partie page 60) on établit une équivalence entre un code binaire compris entre 10 et 15 et un élément alphabétique.

Ainsi :

10 ⟹ M (8+2)
11 ⟹ V (9+2)
12 ⟹ P (8+4)
13 ⟹ X (9+4)
14 ⟹ R (8+6)
15 ⟹ Z (9+6)

Ainsi à la place d'un code 10, on imprimera la lettre M,
L'instruction 10. 15. 3. 11 en mémoire s'écrira à l'imprimante
M. Z . 3 . V

Au moment de l'extraction, les codes binaires de chaque instruction devront donc être transformés en lettres pour les codes supérieurs à 9.

Pour les codes >9 il sera nécessaire d'effectuer une séparation en poids forts et poids faibles de façon à obtenir dans tous les cas une même lettre comme équivalent d'un même code.

En effet : 10 peut être considéré comme la somme de

9 et 2 ce qui donne la lettre V
7 et 3 E
8 et 2 M

or on veut que 10 soit l'équivalent de M ce qui impose la décomposition 8+2 et exclue les 2 autres.

DÉVELOPPEMENT E.T. Problème Transformation :

Seizaine: 0 Piste: 0 Bloc: 4 Série: 2 Code binaire ⇒ Code alphabétique

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD / CB	MD	M 1 (0 11 10 9 8 7 6 5 4 3 2 1 0)	M 2 (0 11 10 9 8 7 6 5 4 3 2 1 0)	MB (0 10 8 6 4 2 0)
		Octade: 4												
		0	1	15					CB					
		1	1	12		1	8		CO					
		2	6	12					BO			1 3 5 7 8 9 10 11 12 13 14 15		
		4	8	4					OB					1 3 5 7 8 9 10 11 12 13 14 15
		5	7	10		8	9		IL			8 8 8 8 8 8 8 8		
		6	8	6					OB					8 8 8 8 8 8
		8	10	1					AN			1 1 1 1 1 1 1 1		
		9	7			1	10		AMD					
		10	8	2					OB				1 1 1 1 1 1 1 1	
		12	6	4					BO			1 3 5 7 8 9 10 11 12 13 14 15		
		13	7	12			11		IL			1 1 1 1		
		14	10	6					AN			8 9 8 9 8 9 8 9		
		16	11	4					SN			1 3 5 7 - - 2 2 4 4 6 6		
		17	8	8			12		OB					1 3 5 7 - - 2 2 4 4 6 6
		18	6	6					BO			8 8 8 8 8 8 8 8		
		20	11	2					SN			7 7 7 7 7 7 7 7		
		21	11	2			13		SN			6 6 6 6 6 6 6 6		
		22	8	2					OB				6 6 6 6 6 6 6 6	
		24	6	4					BO			1 3 5 7 8 9 10 11 12 13 14 15		
		25	7	12			14		IL			2 2 4 4 6 6		
		26	11	4					SN			1 3 5 7 8 9 8 9 8 9 8 9		
		28	8	12					OB					1 3 5 7 8 9 8 9 8 3 8 9
		29	1	12		4	15		CO					
		30	6	12	6	7			BO					
		Octade 5												
		32	9		6	11			CN					
		33	10		6	1	8		AN					
		34	8	12	6	7			OB					
		36	10		6	4			AN					
		37	8	8	10	11	9		OB					
		38	8	15	2	3			OB					
		40		2		5			VC ≠					
		41					10							
		42												
		44												
		45					11							
		46												
		48												
		49					12							
		50												
		52												
		53					13							
		54												
		56												
		57					14							
		58												
		60												
		61					15							
		62	X											

0 11 10 9 8 7 6 5 4 3 2 1 0 | 11 10 9 8 7 6 5 4 3 2 1 0 | 10 8 6 4 2 0

M 1 | M 2 | MB

Les codes alphabétiques choisis appellent les remarques suivantes :

- Ils contiennent tous les poids 8
- Seuls les codes impairs (V - X - Z) contiennent le poids1

Les codes à transformer sont dans les mémoires M 112 à 115 - on doit aiguiller les poids forts vers les M 112 à 115 et les poids faibles vers les M 108 à 111

Une intersection logique avec 8 permet de séparer les codes inférieurs à 8 et ceux supérieurs ou égaux à 8. Le résultat de l'IL est envoyé en M 6. les codes de poids 8 en M 1 sont transformés en poids 1 par décalage de 1 θ (la 10. 1 - - de la ligne 8 est équivalente à 1 multiplication par 2). Ces poids 1 sont envoyés en M 2 dans les mêmes positions que les poids 8 d'origine (AMD et OB).

On effectue alors une 2ème IL des codes d'origine avec les codes 1 en M 2 et on ajoute au résultat de l'intersection, les poids 8 transférés en M 4.

En effectuant alors la soustraction avec les codes d'origine on obtient les poids faibles des codes à transformer.

Exemple : Les codes 11 et 12 ont fourni un poids 8, l'IL de 11 et 12 avec 1 donne 1 et 0, en ajoutant 8, on a 9 et 8 et par soustraction des codes initiaux (redressement binaire) on obtient 2 et 4 c'est-à-dire les poids faibles des codes à transformer.

Les codes 8 contenus en M 6 sont ensuite transformés en codes 6 par 2 SN des codes 1 en M 2 et envoyés en M 2.

Une IL des codes d'origine avec ces codes 6 donne les poids forts des codes à transformer.

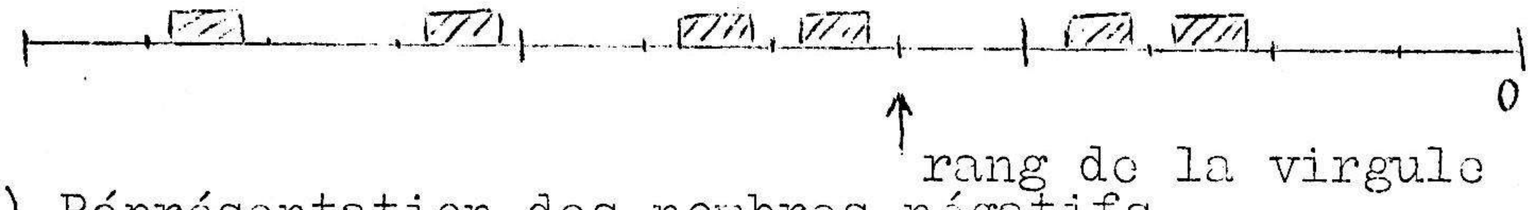
Exemple :

Les codes 12 et 15 donnent 4 et 6 par intersection logique avec 6, et par soustraction des codes initiaux, on obtient 8 et 9, c'est-à-dire les poids forts.

II. OPERATIONS SUR NOMBRES BINAIRES QUELCONQUES

A) Représentation des nombres fractionnaires

Il suffit de définir le rang de la virgule



représente 43,375

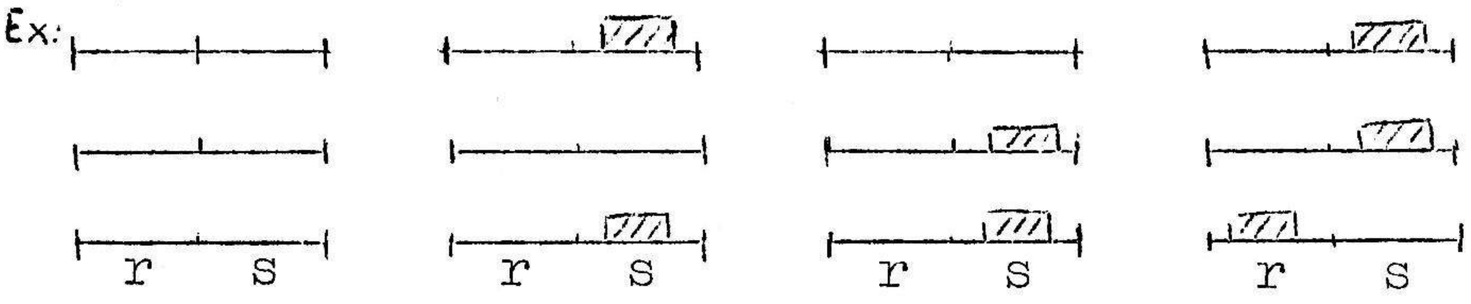
B) Réprésentation des nombres négatifs

L'espace de garde des blocs Tambour interdit l'utilisation des 2 dernières positons binaires de chaque mémoire. Il reste 46 positions binaires utilisables.

On pourra utiliser 45 positions binaires pour la valeur absolue, la dernière position binaire étant affectée au digit de signe. (le signe moins est représenté par un digit de poids 2 dans le dernier τ).

C) Opérateur de signes

En multiplication et division le résultat est donné par la règle des signes. Avec la représentation ci-dessus, il suffit de faire la somme binaire des 2 digits représentant les signes des 2 facteurs, le digit de somme représente le digit de signe du résultat, le digit de report est négligé.



Dans les 4 cas possibles le digit de somme s représente le digit de signe du résultat

En addition on effectue d'abord une comparaison sur les signes

- si les signes sont identiques, on calcule A+B et le résultat est affecté du signe commun.

- si les signes sont différents, on effectue une comparaison sur les valeurs absolues

on calcule alors $|\alpha| - |\beta|$ (avec $|\alpha| > |\beta|$ pour éviter le redressement binaire) le résultat est affecté du signe de α.

D) Adressage d'une position binaire

On réalise cet adressage par intersection logique, ainsi le signe étant représenté par l'impulsion de poids 2 du dernier τ, on effectuera une IL avec 2 du contenu du dernier τ pour isoler le digit de signe.

C H A P I T R E V

EXEMPLES DE SOUS-PROGRAMMES ELEMENTAIRES

-:-:-:-

- Opérations arithmétiques en virgule flottante,
- $\sqrt{X}$
- SIN X et COS X

1.- OPERATIONS EN VIRGULE FLOTTANTE -

Remarque préliminaire :

Les opérations en virgule flottante sont dans la réalité constituées par des séquences d'opérations élémentaires en virgule fixe.

Les codes "opérations virgule flottante" sont des V C S d'adresse 2 qui renvoient sur le tableau de connexions (série 3) à la ligne où commence la séquence d'instructions de l'opération correspondante.

Une V R S commune aux séquences des 4 opérations renvoie dans le programme principal à la ligne qui suit la V C S .

Les facteurs A et B de l'opération à exécuter doivent se trouver en place dans les mémoires 3 et 4 (A en 3, B en 4) avant de commander l'opération.

a) Addition en virgule flottante (Tableau page 78)

La V C S d'addition (1-2-2-3) renvoie à la ligne 8 du tableau de connexions. La séquence comprend 36 lignes d'instructions elle se termine à la ligne 43 par une V R S d'adresse 6 (qui renvoie

dans le programme principal) à la ligne suivant l'opération)

Après le cadrage correct des 2 facteurs, l'addition en virgule flottante se ramène à une addition normale. Les deux facteurs A et B gardent leur individualité et il y a lieu de distinguer 2 cas suivants :

Exposant a $\geqslant$ Exposant b ou non

La comparaison est fait une fois pour toutes au début du programme et est ensuite exploitée pour commander divers aiguillages (schéma page 75, addition et soustraction gamma).

On commence par décaler vers la droite le terme qui a le plus faible exposant, on fait ensuite l'addition normale.

S'il n'y a pas eu de report dans l'addition normale l'exposant de la somme est égal au plus grand des deux exposants ; s'il y a eu report le plus grand des exposants est augmenté de 1.

Le cadrage de la somme est effectué à l'aide d'une D C C le positionnement convenable de la MD avant cette opération permet de faire la correction d'exposant.

Remarque importante :

Il résulte de la définition des nombres en virgule flottante que le résultat n'est pas utilisable pour des calculs ultérieurs par la machine si l'exposant du résultat dépasse 99 ou devient négatif.

Analyse du programme d'addition en virgule flottante (page 73)

Soustraction : Elle se ramène à l'addition

$$A - B = A + (-B)$$

Il faut 3 lignes d'instructions supplémentaires

NL
5 RAZ de M 1, MD, MS (BO avec AD=0 OD=0 OF=0)
6 SN de (B)
7 OB de -(B) dans M 4

Multiplication : On calcule l'exposant maximum du produit a+b-50 et on effectue la multiplication normale des deux mantisses, une VS toujours renvoie alors au début du SP de cadrage où à lieu le calcul correct de l'exposant du résultat.

Division :

On calcule l'exposant minimum du quotient a-b+50 et on effectue la division normale des 2 mantisses. Une VS toujours renvoie ensuite au début du SP de cadrage.

SECURITES ET PRECISION DES OPERATIONS EN VIRGULE FLOTTANTE -

I.- Sécurité de Mantisse nulle.
Lorsque le résultat d'une opération est nul, le tableau PDF fournit un zéro flottant normalisé (mantisse et exposant nuls) lignes 38 à 41 .

II.- Ecriture non normalisée des zéros.
Le tableau PDF peut donner des résultats érronés dans le cas d'une écriture non normalisée des zéros.

Ex.1 - A = 000 000 000.70
B = 100 000 000.50
A+B = 000 000 000.00 Faux

Ex.2 - A = 000 000 000.70
B = 000 000 000.65
A+B = 000 000 000.00

L'écriture des zéros non normalisés pour l'introduction des données constitue donc une source d'erreurs grossières.

III.- SECURITES D'EXPOSANT.
Sur le tableau PDF, il n'est pas prévu de sécurités dans le cas d'un exposant négatif ou supérieur à 99

Ex. 1 - A = 999 000 000.99
B = 100 000 000.97
A+B = 100 000 000.00 Faux

Ex.2 - A = 100 000 000.02
B = 998 000 000.01
A-B = 200 000 000.-2 code 10 en 1-2
Faux

Ex.3 - Codifications des zéros à droite (voir page 101)

VI.-PRECISION DE l'ADDITION ALGEBRIQUE

L'addition de deux nombres PDF égaux, et de signe contraire, donne un résultat égal à zéro.
Mais le tableau PDF peut donner un résultat égal à zéro dans l'addition de deux nombres PDF non égaux (et de signe contraire).

Ex. A = + 125 347 872.52
B = - 125 347 871.52
} A+B = 000 000 000.00 Faux

Le résultat correct est 100 000 000.44

En effet, dans ce cas $\delta = o$, l'addition (sans décalage) des mantisses donne 000 000 001, mais lors du cadrage du résultat, la DCC s'arrête avant le cadrage correct (9 décalages au plus).

V.- ARRONDIS

Les décalages sont faits sans arrondi

-décalage préalable à l'addition des mantisses (Addition, soustraction).
- décalage en vue du cadrage du résultat (par les 4 opérations).

1er cas- On néglige l'un des facteurs $\delta > 8$

Ex. A = 237 624 346.57
B = 999 999 999.48
} A+B = A = 237 624 346.57

Résultat arrondi = 237 624 347.57

2ème cas - Report dans l'addition

Ex A = 999 999 973.55
B = 469 999 999.48
} A+B = 1 00 000 001.56

Résultat arrondi 100 000 002.56

VI.- Cas de la division sans remise à zéro préalable de M2

Dans la division des mantisses, on utilise la division complète, qui définit le dividende en M1 et M2 réunies.

Si M2 n'est initialement nulle, on obtient un quotient supérieur au quotient cherché.

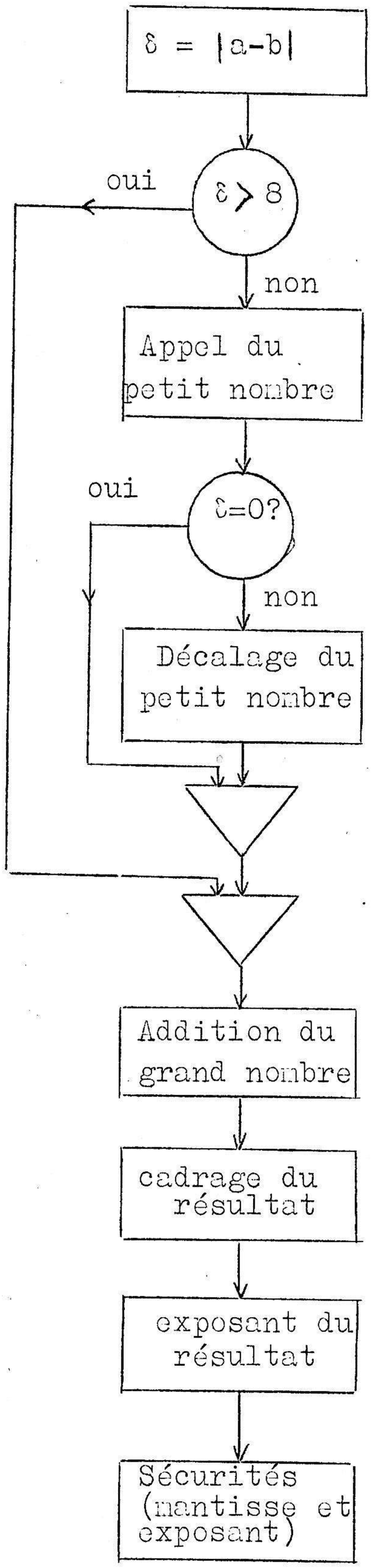
Le tableau PDF remet à zéro la M2, après chaque opération PDF.

L'éventualité d'un contenu non nul de M2 avant une division peut se produire :

- si la première opération PDF dans un programme est une division
- si l'on introduit en M2 en cours de calcul
- si l'on utilise la M2 pour des transferts entre deux opérations PDF.

```
NL
1.2.2.3→ 8    BO de a
         9    CN (a comparé à b)
        10    a-b  SN
        11    |a-b| = δ (MR avec AD=1  OD=OF=0)
        12    OB de δ position 0-2 de M 2
        13    δ-8 (SN avec AD=0  OD=0   OF=8)
        14    Variante MS+ (Si δ-8 > 0 on saute à la ligne 21)
        15    BO de A
        16    Variante VC < (a< b)  si a<b   saut à NL 19
        17    BO de B
        18    Variante VC = (a=b) si a=b saut à NL 25
        19    BD Transfert de la position (0-1) de M 2 dans MD
        20    MC (AD=0  OD=0  OF=0) décalage de A ou B dans M1-M2
                                    de δ positions
        21    BO avec AD=1 maintien filtré de M1 de 2 à 11
        22    Variante VC ⩾ (a ⩾b)  si a ⩾b saut à NL 25
        23    AN (A+B)
        24    VS toujours
        25    AN (B+A)
        26    MR avec AD=0 et OD=2 décalage de C de 2 positions à droite
        27    AMD pour positionner MD à 9
        28    DCC Début de sous programme de cadrage
        29    BO avec AD=1 maintien filtré de C entre 2 et 11
        30    OB de C dans M 4
        31    BO de b                          } Calcul de l'exposant du
        32    VC ≤ (a ⩽ b)                     } résultat.
        33    BO de a                          } (Exploitation d'un re-
        34    AN de 9-(d-1)                    } port éventuel dans l'ad-
        35    SN avec AD=0 (SN de 8 en 0-1)    } dition)
        36    OB de ç corrigé                  }
        37    BO de (C)                        }
        38    CN avec AD=0 Pour savoir s'il y a 1 en position 10 de M 1
        39    VC ⩾(Le chiffre en position 10 de M 1 est-il ⩾1 )
        40    OB avec AD=0  OD=0  OF=0  RAZ de M 1, MD, MS
        41    ZB de M 4
        42    ZB de M 2
P.P.←   43    VRS
```

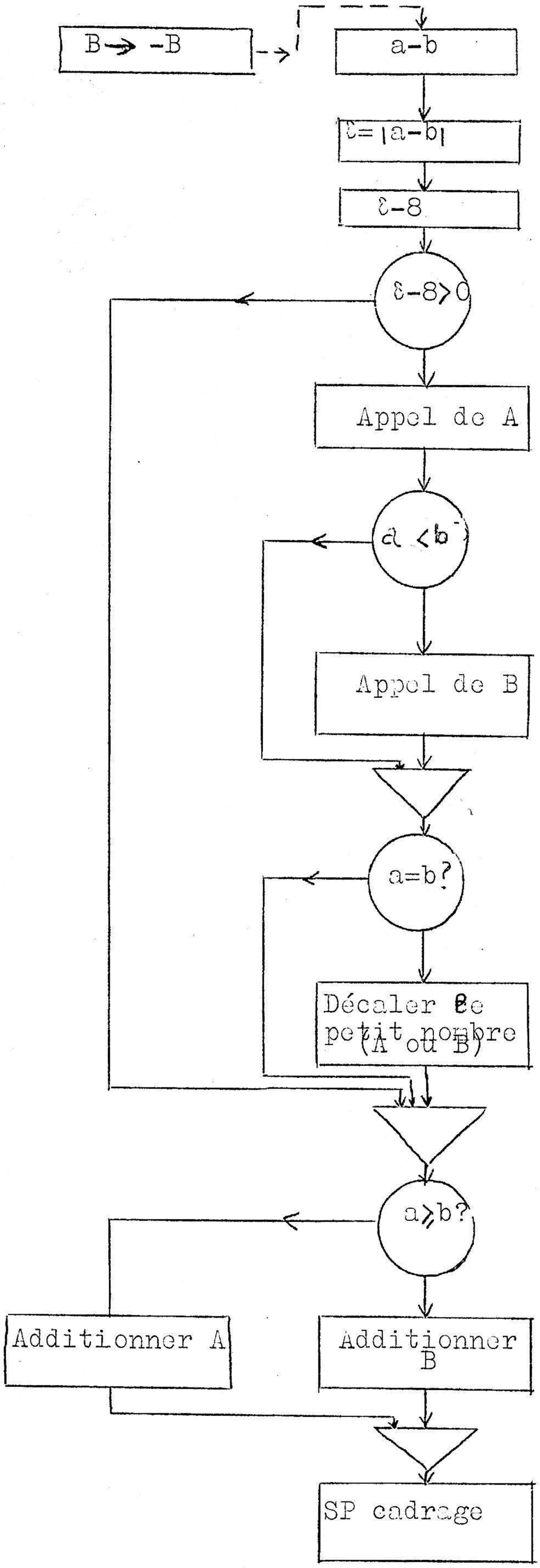
ADDITION EN VIRGULE FLOTTANTE

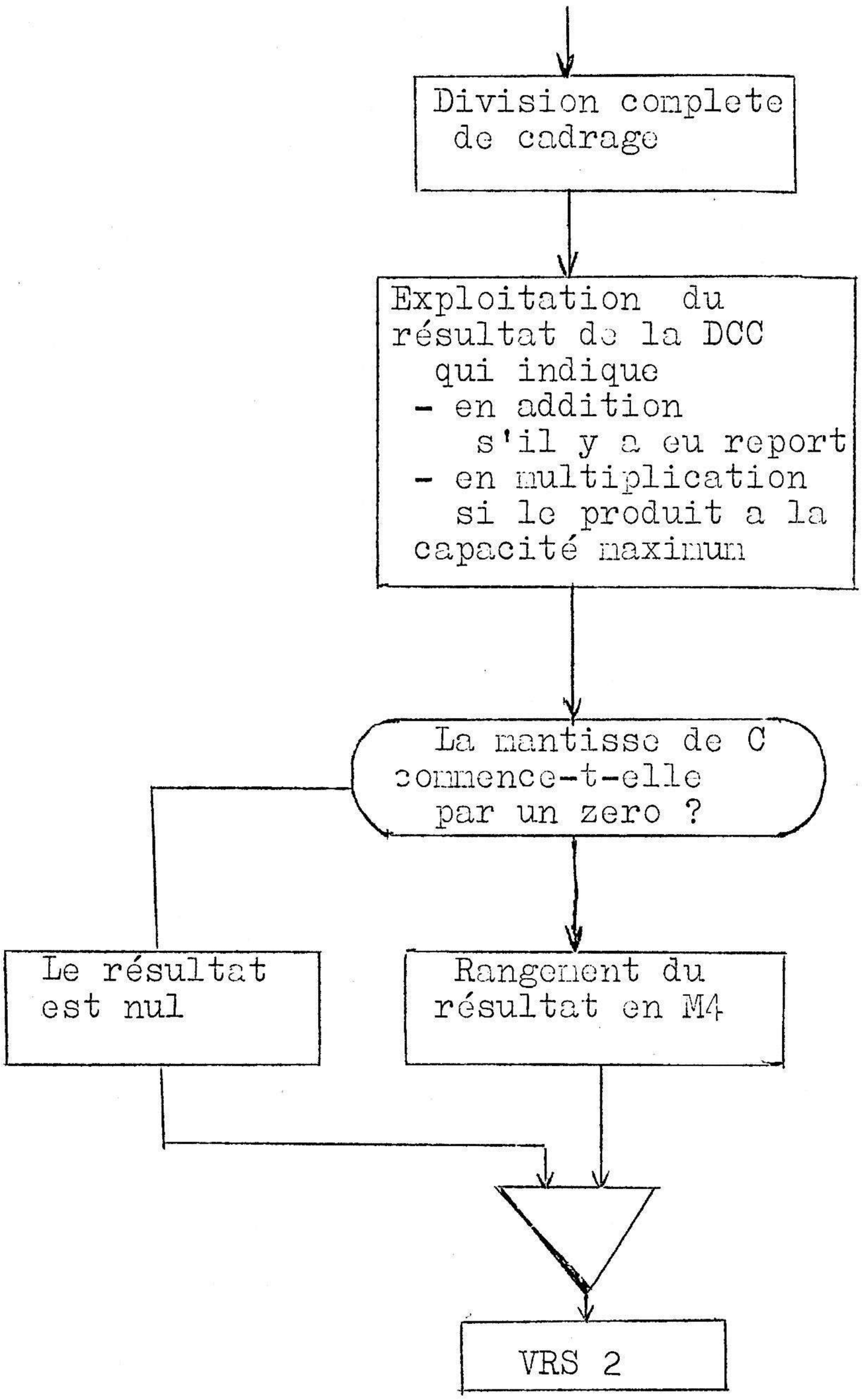


Remarque : le petit nombre est le nombre de plus petit exposant

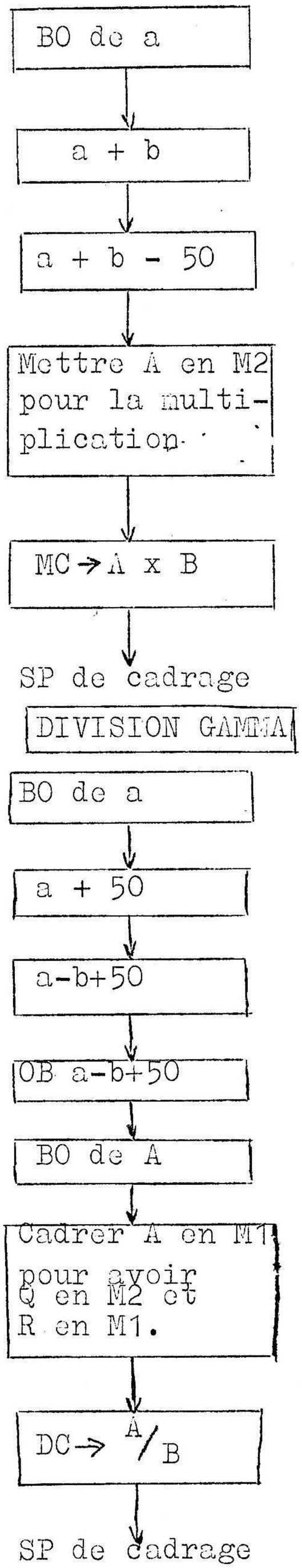
Si les exposants sont égaux, c'est l'un ou l'autre.

ORGANIGRAMME D'ADDITION ET DE SOUSTRACTION GAMMA




Division complete de cadrage
Exploitation du résultat de la DCC qui indique
- en addition s'il y a eu report
- en multiplication si le produit a la capacité maxinum
La mantisse de C commence-t-elle par un zero ?
Le résultat est nul
Rangement du résultat en M4
VRS 2

MULTIPLICATION GAMMA



$\sqrt{x}$

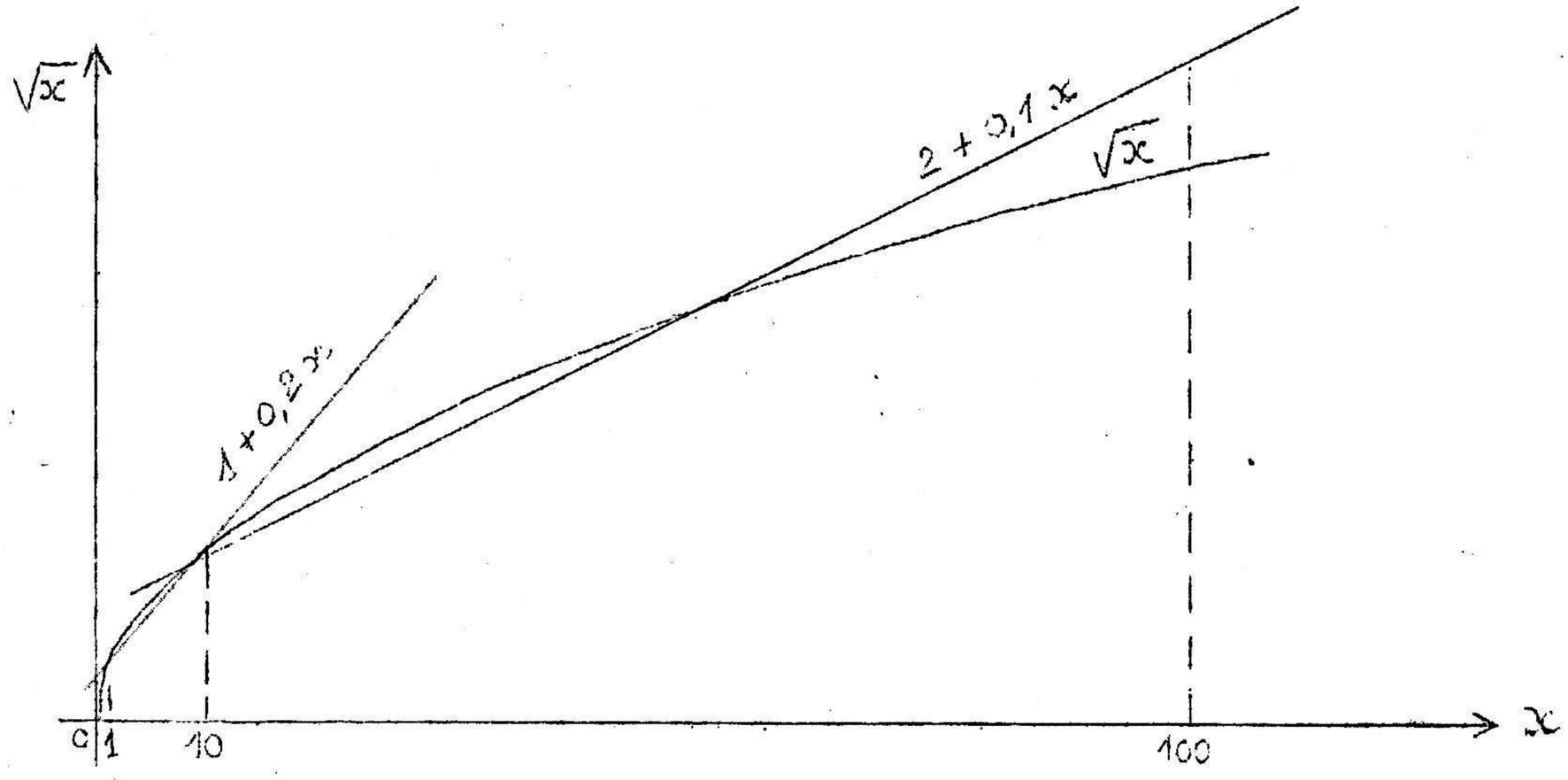
1.- METHODE MATHEMATIQUE -

On utilise la formule de Newton donnant la racine d'un nombre A par valeurs approchées successives.

$$\beta_{n+1} = \frac{1}{2}\left[\beta_n + \frac{A}{\beta_n}\right]$$

Les itérations s'arrêtent lorsque $|\beta_{n+1} - \beta_n| < 10^{-9}$

Choix de la valeur initiale β_o



Pour des raisons de facilités de programmation, on cherche à interpoler linéairement la courbe $\sqrt{x}$ dans chacun des 2 intervalles $[1,10]$ et $]10,100[$

Les droites 1 + 0,2 x pour l'intervalle $[1,10]$ et 2 + 0,1x dans l'intervalle $]10,100[$ fournissant une valeur initiale satisfaisante.

La valeur initiale β_o sera donc égale à

1 + 0,2 A si a est impair

2 + 0,1 A si a est pair

Ces calculs se font très simplement en langage machine au moyen d'additions et de décalages.

2.- DOMAINE DE VALIDITE DU SOUS-PROGRAMME -

Le domaine de validité est constitué par l'ensemble de toutes les valeurs permises par l'écriture en virgule flottante (10^{-50} , 10^{49}) pour lesquelles le programme fonctionne correctement, moins certaines valeurs explicitement indiquées

Par la suite on indiquera donc seulement les valeurs particulières ou les domaines interdits.

Pour $\sqrt{A}$ si A = 0 A = 0

si A < 0 Blocage NL 17 S 1

3.- PRECISION-

La précision est définie de la façon suivante :

- soit y la vraie valeur de la fonction, son écriture flottante est

$$\underline{s.A.a}$$

- soit y la valeur approchée calculée à l'aide du sous-programme son écriture flottante est $\underline{s'.A'.a'}$

Pour tous les sous-programmes

s = s'

a = a' sauf cas très exceptionnels où l'arrondi ne peut se produire et où par exemple 1 ---- 51 apparait sous la forme 999999999.50

La précision des S-P est indiquée en donnant le nombre de chiffres communs pour A et A' en faisant la convention suivante :

Les 2 écritures ($x \neq 0$)

p	x	0 0 0 0
p	x-1	9 9 9 9

} sont "équivalentes"

Pour $\sqrt{x}$, l'erreur est légèrement inférieure à une demi-unité du 9° ordre.

4.- PROGRAMMATION -

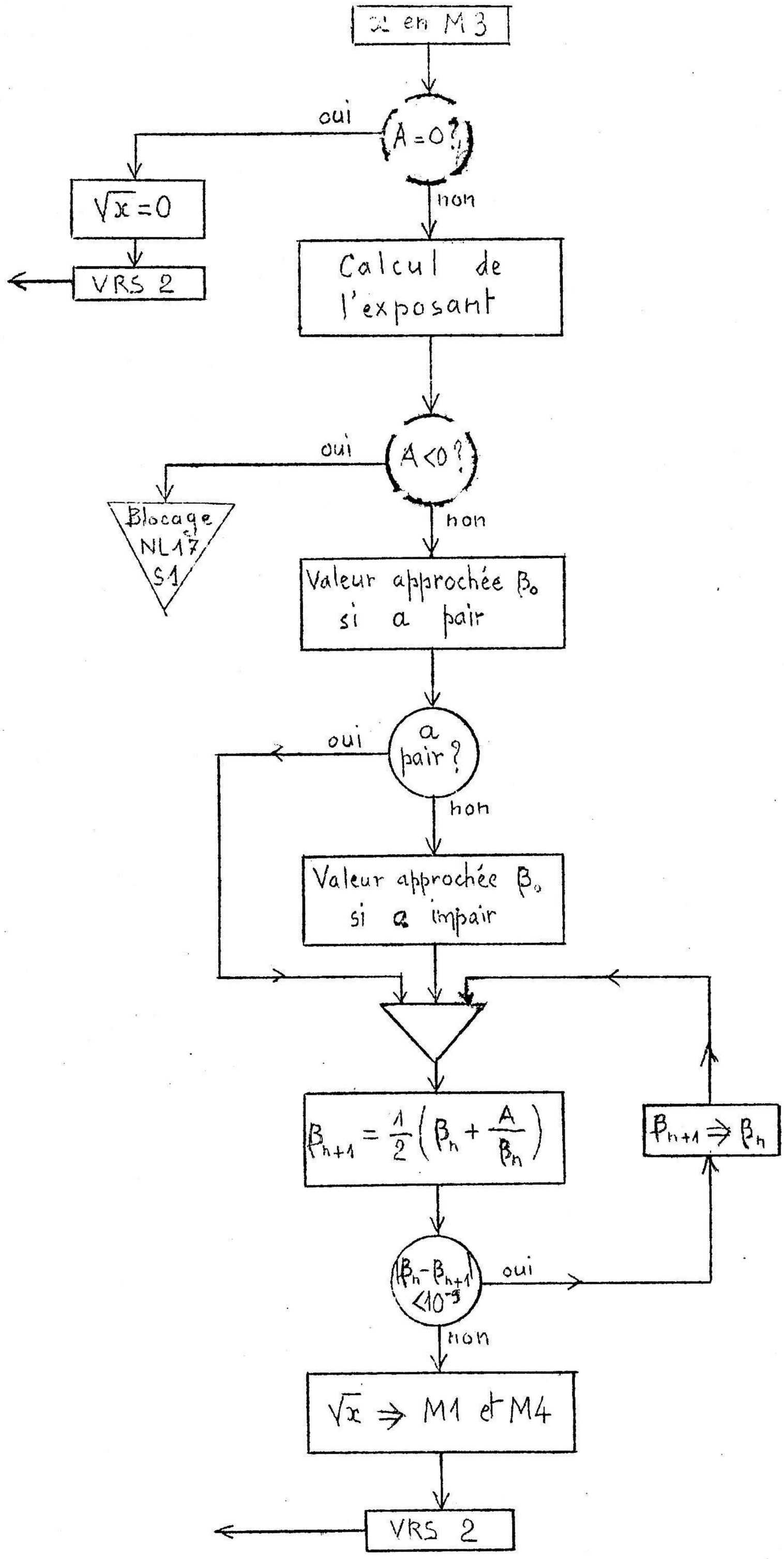
Calcul de l'exposant -

Après s'être assuré que le nombre n'est pas nul, on détermine l'exposant de la racine carrée par a' = partie entière de $\left[\frac{a + 51}{2}\right]$ Une comparaison permet de connaitre la parité de a pour le choix de la valeur initiale β_o.

Calcul de la mantisse -

Après s'être assuré que la mantisse n'est pas négative on détermine la valeur initiale β_o suivant la parité de a.

Un arrondi légèrement biaisé (9 sur un nombre divisé ensuite par 2) permet d'éviter que 99,9999 = 10.


A en M3
A=0 ?
oui
non
√x = 0
VRS 2
Calcul de l'exposant
A<0 ?
oui
non
Blocage NL17 S1
Valeur approchée β0 si a pair
a pair ?
oui
non
Valeur approchée β0 si a impair
$\beta_{n+1} = \frac{1}{2}\left(\beta_n + \frac{A}{\beta_n}\right)$
$\beta_{n+1} \Rightarrow \beta_n$
$|\beta_n - \beta_{n+1}| < 10^{-9}$
oui
non
$\sqrt{x} \Rightarrow$ M1 et M4
VRS 2

DÉVELOPPEMENT E.T.

Problème: $\sqrt{x}$

Seizaine: 0 Piste: 5 Bloc: 0 Série: 1

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD/CB	MD	M1	M2	MB
												0 11 10 9 8 7 6 5 4 3 2 1	0 11 10 9 8 7 6 5 4 3 2 1	0 10 8 6 4 2 0
		Octade: 2												
		0	1	10					CD					
		1	1	12		2	8							
		2	6	3	2	11			A			1 1		
		4	9		5	1						5 4 3 2		
		5	8	4			9							
		6		3	13	1			<					
		8	6	3		2						1 a		
		9	10	9		2	10					a + 51		
		10	12		3	5						51		
		12	7			1								
		13	8	4		2	11							
		14	9		11	5								
		16	6	3	2									
		17		4	4	4	12		MS =			Blocage:		
		18		3	5	4			≥					
		20	3	13	5	6								
		21	8	8	1		13		<					
		22		3	6	9								
		24	10		9	2			VS touj.					
		25			7	5	14					0,1A + 0,5		
		26	10		8	5						0,2A + 1		
		28	10	1										
		29	7				15							
		30	8	9										
		Octade 3												
		32	6	8										
		33	3	2			8							
		34	7			10								
		36	15	9										
		37	6	2			9							
		38	10			9								
		40	10	9										
		41	12		1		10							
		42	12		10	5								
		44	8	9										
		45	6	1	1		11							
		46	9	4	2									
		48	8	4	2									
		49		2	8	1	12							
		50	6	4										
		52	3	2										
		53	1	6			13							
		54												
		56												
		57					14							
		58												
		60												
		61					15							
		62	X											

arrondi en vue de troncature de 10

décalage de 10 à droite

placement de (0-1) en M1

$x_{n+1} \Rightarrow x_n$

≠

0 11 10 9 8 7 6 5 4 3 2 1 | 0 11 10 9 8 7 6 5 4 3 2 1 | 0 10 8 6 4 2 0

M1 M2 MB

$$\boxed{Y = \text{SIN } X}$$

1.- METHODE MATHEMATIQUE -

Après avoir réduit l'arc au premier quart de quadrant, on utilise un développement en série de Taylor.

A- Réduction de l'arc

a) On détermine K et φ_1 , k entier positif, négatif ou nul

et $0 \leqslant \varphi_1 < \frac{\pi}{2}$

tels que $\boxed{x = K.\frac{\pi}{2} + \varphi_1}$

$$\sin x = \sin(K.\frac{\pi}{2} + \varphi_1) = \sin K.\frac{\pi}{2} \cos \varphi_1 + \cos K.\frac{\pi}{2} \sin \varphi_1$$

α) si K est pair $K = 2p$

$$\boxed{\sin x = (-1)^p \sin \varphi_1}$$

β) si k est impair : $K = 2p + 1$

$$\boxed{\sin x = (-1)^p \cos \varphi_1 = (-1)^p \sin(\frac{\pi}{2} - \varphi_1)}$$

on est donc toujours ramené à calculer le sinus d'un arc φ_2

$0 \leqslant \varphi_2 < \frac{\pi}{2}$ avec $\varphi_2 = \begin{cases} \varphi_1 & \text{pour K pair} \\ \frac{\pi}{2} - \varphi_1 & \text{pour K impair} \end{cases}$

b) Si $0 \leqslant \varphi_2 < \frac{\pi}{4}$ posons $\varphi = \varphi_2$ d'où $\boxed{\sin \varphi_2 = \sin \varphi}$

Si $\frac{\pi}{4} \leqslant \varphi_2 < \frac{\pi}{2}$ posons $\varphi = \frac{\pi}{2} - \varphi_2$ d'où $\boxed{\sin \varphi_2 = \cos \varphi}$

B- Développement en série de Taylor

$$\sin \varphi = \varphi - \frac{\varphi^3}{3!} + \ldots\ldots\ldots + (-1)^p \frac{\varphi^{2p+1}}{(2p+1)!} + \ldots\ldots$$

$$\cos \varphi = 1 - \frac{\varphi^2}{2!} + \ldots\ldots\ldots + (-1)^p \frac{\varphi^{2p}}{(2p)!} + \ldots\ldots$$

C- Erreur de méthode

Les séries étant altérnées, l'erreur due en négligeant le reste des termes de la série est inférieur au dernier terme négligé.

On arrête la sommation lorsque le dernier terme calculé est inférieur à 10^{-9} en valeur absolue .

2.- DOMAINE DE VALIDITE DU SOUS-PROGRAMME -

Pour $x < 10^{-4}$ on écrira directement

sin x = x
cos x = 1

Pour $x > 10^{10}$, la réduction de l'arc n'a plus de sens (φ n'a aucun chiffre significatif) Blocage NL 10 S 1

3.- PRECISION -

On admet que l'arc est donné sans erreur

Pour $10^{-50} < x < 10^{-4}$ sin x est déterminé exactement

$10^{-4} < x < 10$ sin x a 9 chiffres significatifs exacts

$10 < x < 10^{10}$ le nombre de chiffres significatifs diminue lorsque x augmente

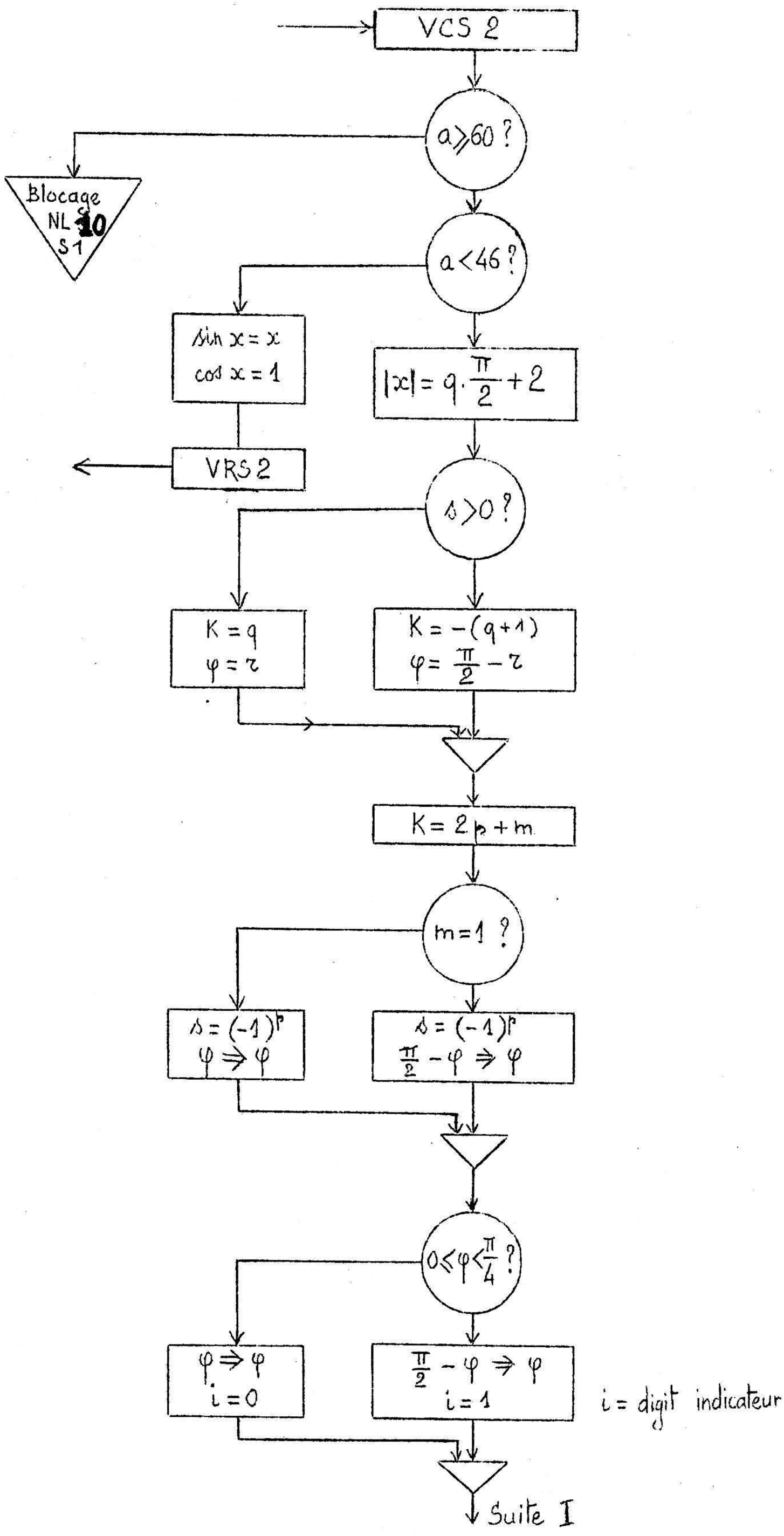
si $x = K.\ 10^{p}$, le nombre de chiffres significatifs exacts est de l'ordre de 10-p

4.- PROGRAMMATION -

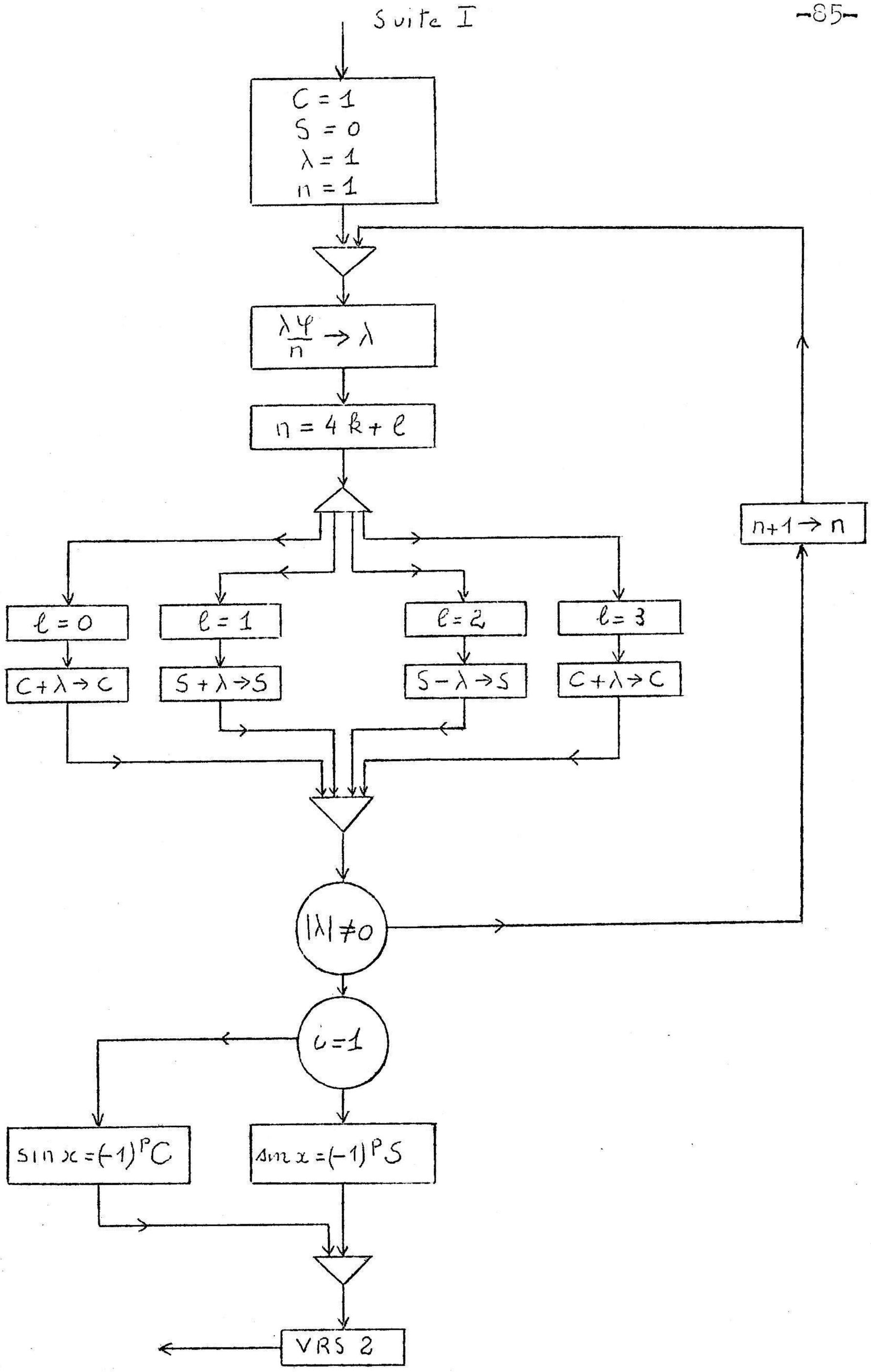
En plus de la mémoire qui contient le terme φ de la série on utilise 4 mémoires C, S, λ et n

(C, S, λ sont des mémoires complètes à 12 positions, n utilise 2 positions de mémoire).

Le reste de la division de n par 4 permet d'additionner ou de soustraire ce terme à la série sinus ou sinus.


VCS 2
a ⩾ 60 ?
Blocage
NL S10
S1
a < 46 ?
sin x = x
cos x = 1
|x| = q · π/2 + 2
VRS 2
δ > 0 ?
K = q
φ = τ
K = −(q+1)
φ = π/2 − τ
K = 2p + m
m = 1 ?
δ = (−1)^p
φ ⇒ φ
δ = (−1)^p
π/2 − φ ⇒ φ
0 ⩽ φ < π/4 ?
φ ⇒ φ
i = 0
π/2 − φ ⇒ φ
i = 1
i = digit indicateur
Suite I

Suite I


C = 1
S = 0
λ = 1
n = 1
λφ/n → λ
n = 4k + ℓ
ℓ = 0
ℓ = 1
ℓ = 2
ℓ = 3
C + λ → C
S + λ → S
S − λ → S
C + λ → C
n+1 → n
|λ| ≠ 0
i = 1
sin x = (−1)^p C
sin x = (−1)^p S
VRS 2

DÉVELOPPEMENT E.T. Problème: Sin - Cos (1)

Seizaine: 0 Piste: 5 Bloc: 1 Série: 1

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD/CB	MD	M1	M2	MB
												0 11 10 9 8 7 6 5 4 3 2 1	0 11 10 9 8 7 6 5 4 3 2 1	0 10 8 6 4 2 0
		Octade: 2												
		0	1	10			8							
		1												
		2	2	4	7	12								
		4	1	12		2	9							
		5	6	3		11								
		6	8	4										
		8	6	3		2	10							
		9	9		1	6								
↱		10		3	2	8			≫					
		12	9		1	5	11							
		13	3	9										
		14		1	14	1			≪					
		16	11		1	5	12							
		17	8	13	4	5								
		18	6	3	2	11								
		20	9	1	1		13							
		21	7											
		22			8	1			VS rouj.					
		24	1	6	3	3	14		π/4					
		25	5	3	9	8								
		26	4	1	7	8								
		28	3	2	6	7	15		π/2					
		29		7	9	6								
		30		1	5	7								
		Octade 3												
		32	3	2			8							
		33	15	15										
		34	9	14		10								
		36		1	10	1	9		≪					
		37	11	15										
		38	4	4	11	1								
		40	8	8			10							
		41	6	2		10								
		42	8	9										
		44	3	2			11							
		45	6	8										
		46	7			9								
		48	15		10	1	12							
		49	6	1	2									
		50	8	4	2	11								
		52	6	2		1	13							
		53	10	14	10									
		54	8	4		2								
		56	6	9			14							
		57	2	2	5	5								
		58	1			5								
		60					15							
		61												
		62	X											

	M1	M2	MB
	C M 10 9 8 7 6 5 4 3 2 1 0	M 10 9 8 7 6 5 4 3 2 1 0	M 8 6 4 2 0

DÉVELOPPEMENT E.T. Problème : Sin - Cos (2)

Seizaine: 0 Piste: 5 Bloc: 2 Série: 1

	Sélection lignes	NL	TO	AD	OD	OF		Calcul instructions	Opération	CD/CB	MD	M1	M2	MB
		Octade: 2												
		0	4	5										
		1	13		10	4	8							
		2	2	4	5	7								
		4	6	1	10	11								
		5	8	2	11		9							
		6	6	3	11									
		8		4	3	1			MS+					
	↓	9	4	8	1	1	10							
		10	4	1	11	3								
		12	10	2	11									
		13	8	8		1	11							
		14	6			1								
		16	4	1	1	5								
		17	11	4		2	12							
		18	8	2										
		20	6	8		1								
		21	9			1	13							
		22												
		24		2	7				=					
	↓	25	9			2	14							
		26		2	7	9			≠					
	↓	28	4	2	11	10								
		29	9			2	15							
		30	10	4	11									
		Octade 3												
		32	10	8	1	2								
		33	7	10		1	8							
		34	3	4	11									
		36		3	9	9			<					
	↓	37	4	4	11	10	9							
		38	8	2	4	11								
		40	6	4		2								
		41	9	8	2	4	10							
		42	1	12		5								
		44	6	2										
		45	1			2	11							
		46												
		48												
		49					12							
		50												
		52												
		53					13							
		54												
		56												
		57					14							
		58												
		60												
		61					15							
		62	X											

0 11 10 9 8 7 6 5 4 3 2 1 0 11 10 9 8 7 6 5 4 3 2 1 0 10 8 6 4 2 0

M1	M2	MB

DÉVELOPPEMENT E.T. Problème: Sin - Cos (3)

Seizaine: 0 Piste: 5 Bloc: 3 Série: 2

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD / CB	MD	M 1	M 2	MB
		Octade: 4												
		0	2	2	5	9								
		1	8	15		1	8							
		2		3	1	8								
		4	3	2										
		5	1		5	1	9							
		6	1	12		4								
		8	8	8										
		9	6	4	2	11	10							
		10	7											
		12	8	9										
		13	3	10			11							
		14												
		16	4	8	1	1								
		17	3	11			12							
		18	1	15										
		20	6	8	1	3								
		21	7	10		3	13							
		22	9		1	2								
		24	1	12		5								
		25	4	12	3	10	14							
		26		3	7	5								
		28	4	12	3	11								
		29	7	10		1	15							
		30	9											
		Octade 5												
		32	4	11	10	11								
		33	4	12	6	11	8							
		34		2	9	8								
		36	4	11	10	10								
		37	4	12	6	10	9							
		38	1	12		4								
		40	6	8	1	3								
		41	10		1	1	10							
		42	8	8	1	3								
		44	1	10										
		45	8	8	9	11	11							
		46	6											
		48		9										
		49	8				12							
		50	6	4	2	11								
		52	7			1								
		53	8	2			13							
		54	6											
		56	14	9										
		57	15	8	9	11	14							
		58	6	2										
		60	7											
		61	14				15							
		62	1			1								

DÉVELOPPEMENT E.T. Problème: Sin - Cos (4)

Seizaine: 0 Piste: 5 Bloc: 4 Série: 1

	Sélection lignes	NL	Codes TO	AD	OD	OF		Calcul instructions	Opération	CD/CB	MD	M1	M2	MB
		Octade: 2												
		0	8	9			8							
		1	9											
		2		2	1	4			=					
		4	1		4	10	9							
		5	6	10	2									
		6	9		10	1								
		8		3	3	8	10		≥					
		9	6	4		2								
		10	11			1								
		12	8	4		2	11							
		13	6	10	1									
		14	8	4	2	11								
		16	6	11			12							
		17	8	2										
		18	6	8										
		20	2	4	7	13	13							
		21	1	12		2								
		22	8	8										
		24	8	14	9	10	14							
		25	6	2										
		26	12		□									
		28	10		10	1	15							
		29	7			11								
		30	9		10	1								
		Octade 3												
		32		3	9	1	8		<					
		33	6		11	1								
		34	4	1	9	1								
		36	4	1		5	9							
		37	8	2		11								
		38	6	8	4	5								
		40	9				10							
		41	6	2		11								
		42	8	8		11								
		44	6	4		11	11							
		45		2	13	8			=					
		46	12	8										
		48	9		10	1	12							
		49		3	13				≥					
		50	6											
		52	8	8			13							
		53	6	2		11								
		54	12	4										
		56	9		10	1	14							
		57		3	15				≥					
		58	6											
		60	8	4			15							
		61	1		15	9								
		62	X	6										

M1 M2 MB

C H A P I T R E VI

PROGRAMME INTERNE ET DUREE DES OPERATIONS ELEMENTAIRES

I. - CIRCUITS DE POSITION ET CIRCUITS DE COMMANDE. -

Dans les machines électromécaniques, l'exécution d'un travail se trouve découpée en intervalles de temps égaux, chaque intervalle de temps correspondant à un cycle de la machine électromécanique. Les circuits de position (véhiculant les nombres) et les circuits de commande (véhiculant les ordres) sont accessibles à l'utilisateur qui, au moyen de connexions externes, détermine pour chaque cycle des circuits de position bien définis ; les cycles se succédant dans un ordre défini lui aussi par des connexions externes. (cf. 3ème partie, Tabulatrice).

Dans les machines électroniques, les circuits de position ne sont pas accessibles à l'utilisateur, les connexions entre circuits de position sont remplacées par des commutations établies directement par les circuits de commande.

II. - ORGANISATION DES CIRCUITS DE COMMANDE. -

A partir d'un programme externe (suite d'opérations élémentaires) affiché sur tableau (série 3) ou enregistré en mémoires rapides (séries 0 - 1 - 2), le calculateur prend connaissance de ce programme externe au moyen des organes de programme qui se composent :

- de 2 mémoires contenant l'adresse de la prochaine instruction à exécuter - mémoires numéro de ligne (NL) et numéro de série (NS) ;
- de 4 mémoires contenant l'instruction en cours d'exécution : mémoires-programme : TO, AD, OD, OF.

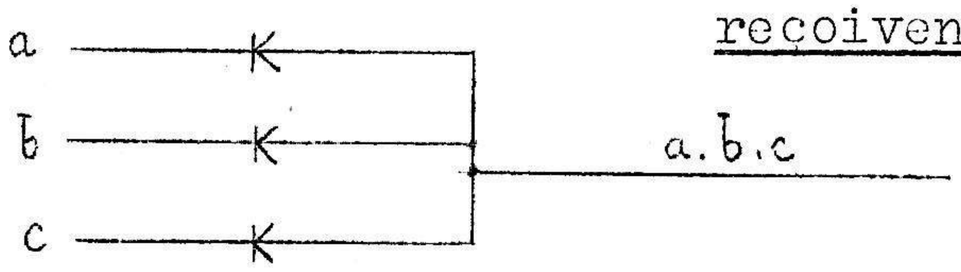
Les codes contenus dans les mémoires-programme permettent d'établir les commutations intervenant dans l'opération à exécuter. Par exemple, le transfert de la mémoire 5 vers la mémoire opérateur doit valider la sortie de M 5 sur le canal données (commutation établie à partir de AD = 5) et valider le canal-données sur l'entrée de la mémoire opérateur (commutation établie à partir de TO = 6).

CONDITIONNEURS ET MELANGEURS.

Les commutations sont réalisées à partir des deux circuits logiques de base correspondant à la somme et au produit logiques de deux informations binaires.

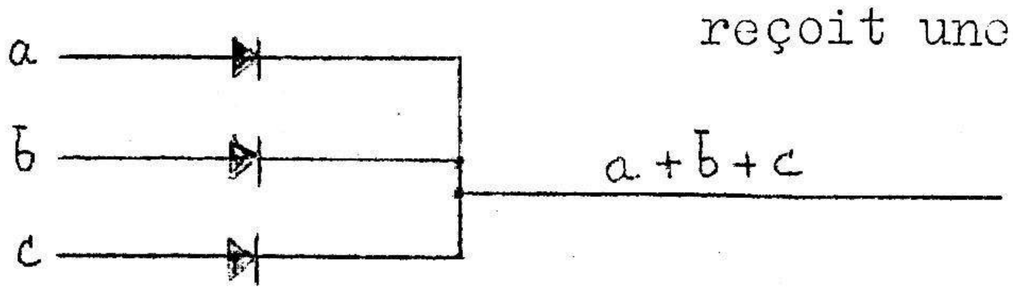
La somme logique (condition et) est réalisée par un conditionneur.

La sortie n'émet une impulsion que si toutes les entrées reçoivent simultanément une impulsion.



Le produit logique (condition ou) est réalisé par un mélangeur.

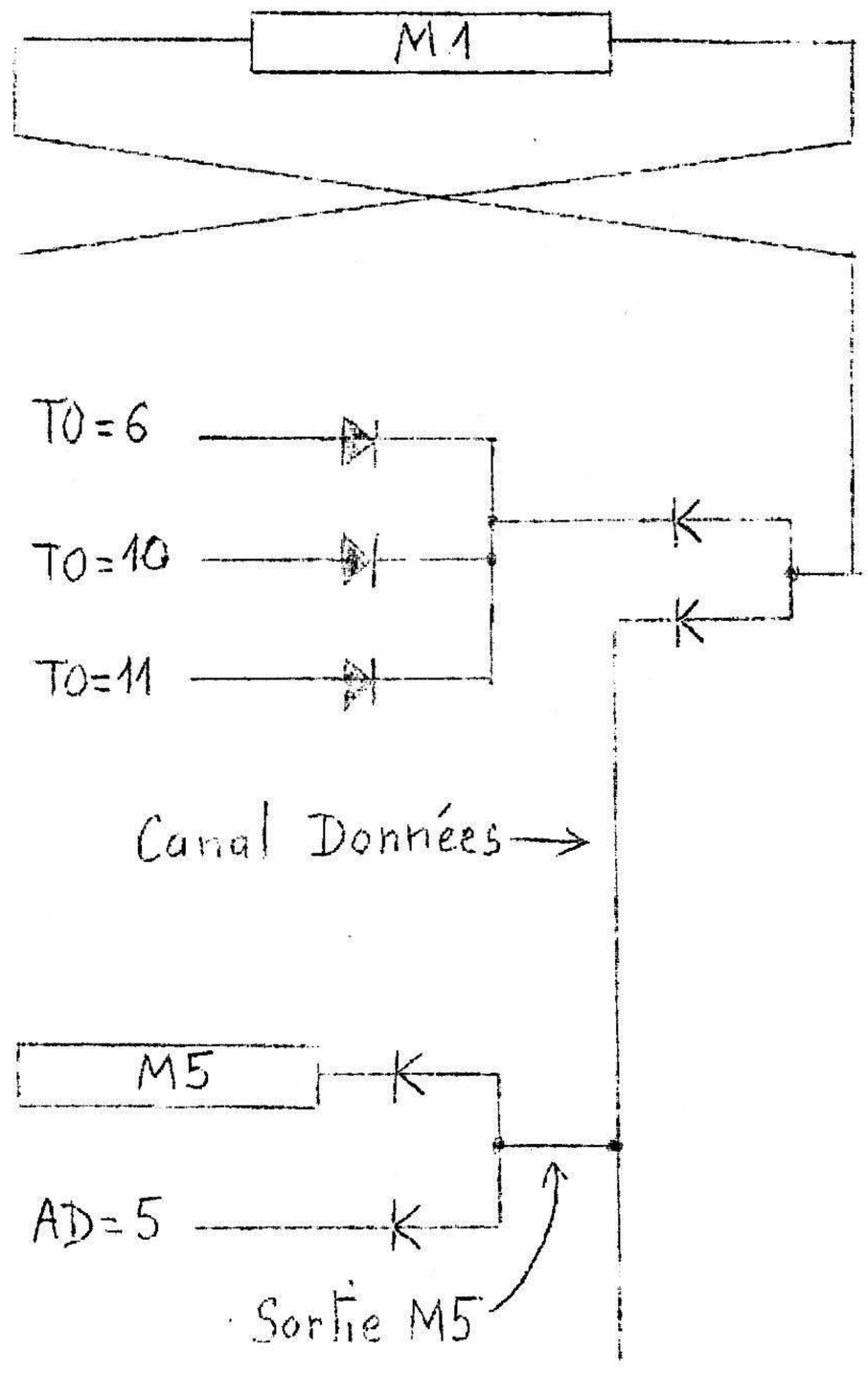
La sortie émet une impulsion si l'une quelconque des entrées reçoit une impulsion.



Exemple :

La sortie de M 5 sur le canal données est validée par un conditionneur dont une des entrées reçoit les impulsions circulant dans M 5 et l'autre entrée des impulsions émises en synchronisme lorsque la mémoire AD contient 5.

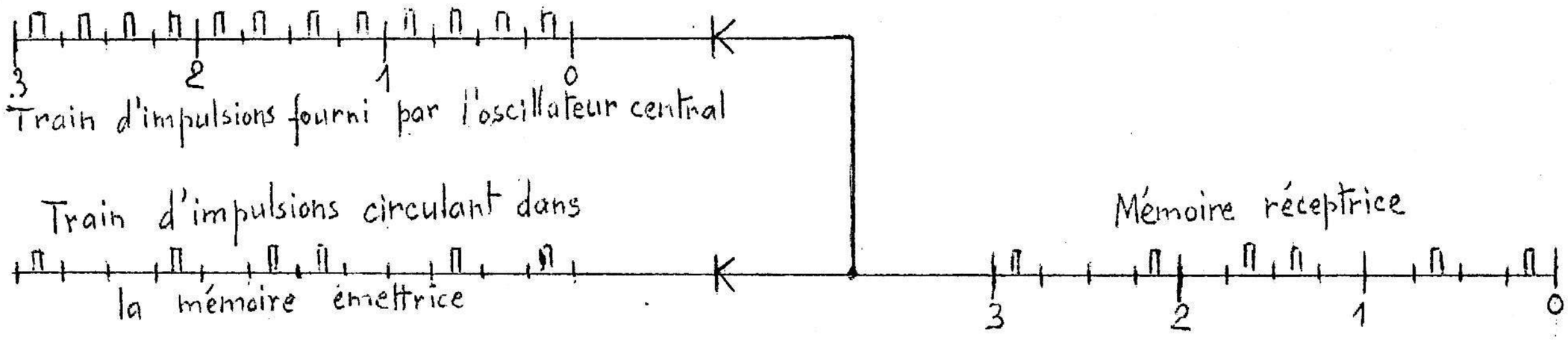
Le canal-données est validé sur l'entrée de M 1 par un conditionneur dont l'une des entrées provient du canal-données reliant les sorties de toutes les mémoires et de la sortie d'un mélangeur dont les entrées émettent des impulsions lorsque la mémoire TO contient 6 ou 10 ou 11.



PROGRAMME INTERNE.

Les impulsions permettant d'établir les commutations sont fournies en permanence par un oscillateur central. Elles sont émises par trains (Pour valider le transfert du contenu d'une mémoire banale de 12 positions décimales, soit 48 positions binaires, vers la mémoire opérateur, il est nécessaire d'émettre à l'entrée des conditionneurs, 48 impulsions en synchronisme avec les 48 impulsions présentes ou absentes, correspondant au contenu de la mémoire banale).

Exemple :



Les graduations correspondent aux positions décimales des mémoires. Ces positions décimales repèrent en fait des instants de passage pour les impulsions représentant l'information.

La séquence des trains permettant d'établir les commutations nécessaires pour l'exécution de chaque opération élémentaire constitue le programme interne.

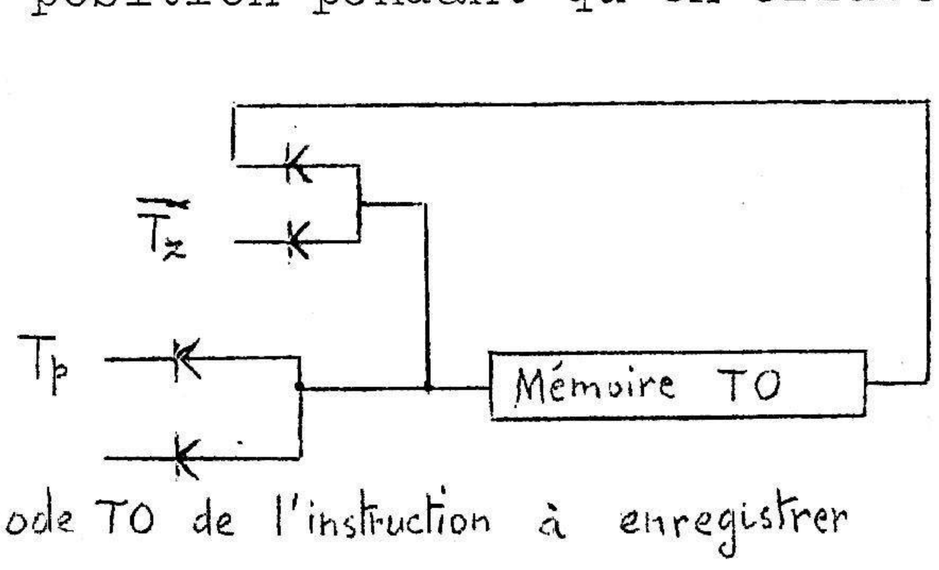
A chaque type d'opération élémentaire correspond un programme interne fixé une fois pour toutes par le constructeur.

L'introduction du programme : Trains préparatoires.

L'introduction des 4 codes d'une opération élémentaire dans les mémoires-programmes constitue l'introduction du programme.

Les commutations qui valident cette introduction sont contrôlées par deux trains d'impulsions ; l'un permet l'effacement des codes de l'opération précédente, l'autre permet le transfert des codes à enregistrer.

Le train d'effacement se nomme Tz et le train d'introduction Tp. Les deux trains Tp et Tz sont démarrés simultanément ; en effet, l'effacement d'une mémoire consiste à ouvrir sa boucle de telle façon que le code qui sort, ne se présente plus à l'entrée. Il est donc possible d'entrer sur une position pendant qu'on efface la position qui va suivre.



La boucle est fermée tant qu'il y a absence de Tz ($\overline{Tz}$)
La présence simultanée de Tz et Tp permet l'introduction et l'effacement simultanés.

Exécution du programme interne : Trains opératoires.

Les commutations des nombres sont contrôlées par des trains qui jouent un double rôle :

- décomposer l'opération en plusieurs phases élémentaires,
- valider les commutations pendant des durées bien déterminées correspondant au temps de passage des nombres.

Les trains opératoires sont au nombre de 3 :

- le train T 1 valide les transferts par le canal-donnée ou le canal-résultat suivant le code du type d'opération ; il assure également l'effacement intégral de M 1 dans l'opération BO.
- le train T 2 assure l'effacement des mémoires banales ;
- le train T 3 commande les décalages de la mémoire opérateur intervenant dans le cadrage préalable, ainsi qu'en multiplication et division.

Ces trains ont une durée fixe égale à 1 tour de mémoire soit 172 μ s. Le tour de mémoire correspond au temps néeessaire pour que les 48 impulsions binaires d'une mémoire défilent successivement en un même point de la mémoire.

Ces trains sont démarrés en même temps que la première impulsion d'une mémoire (poids binaire 1 de la première position décimale) se présente à l'entrée des canaux et cessent avec la dernière impulsion (poids binaire de la dernière position décimale) après avoir émis systématiquement 48 impulsions.

Séquence des trains.

L'évolution des différentes phases d'une opération est surveillée en permanence par deux comparateurs : le comparateur de cadrage, principalement chargé des décalages et le comparateur général, principalement chargé des transferts.

Ces comparateurs sont dotés de mémoires qui conservent le souvenir des comparaisons faites. Ils ont pour rôle de déterminer, à la fin de chaque tour de mémoire, en fonction du type d'opération et de l'état d'avancement de l'opération, le départ d'un des trains T 1, T 2, T 3, ou l'émission du signal de fin d'opération démarrant Tp.

Les trains agissent eux-mêmes sur les comparateurs, soit

indirectement, en faisant évoluer le contenu des mémoires reliées aux comparateurs, soit directement, afin de faire conserver par les comparateurs, le souvenir de leur exécution.

Sélection des lignes du programme.

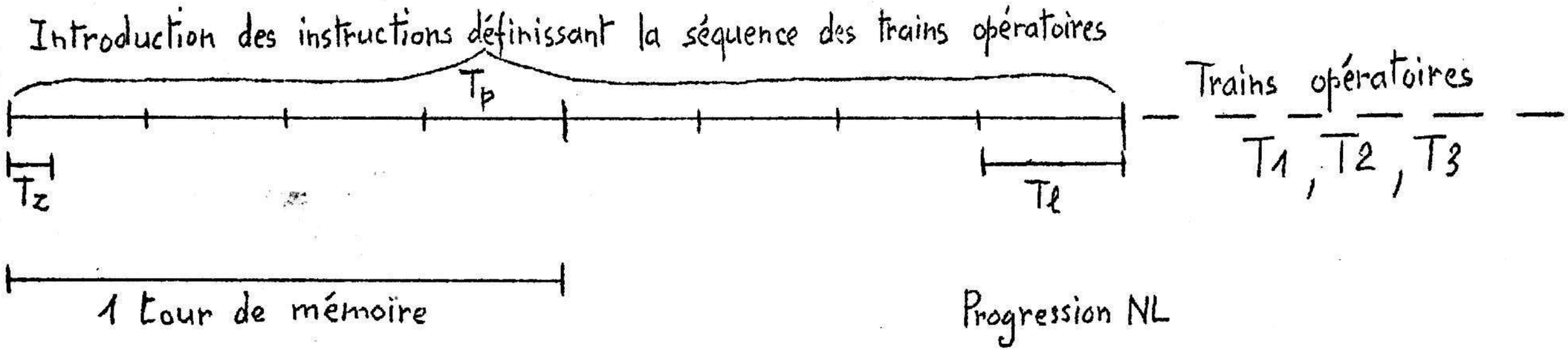
Cette sélection est obtenue à partir du contenu des mémoires NL et NS.

La mémoire NL progresse normalement d'une unité à chaque opération élémentaire sauf dans les cas de ruptures de séquence. (Les ruptures de séquences peuvent avoir lieu soit à l'intérieur d'une même série, soit d'une série sur l'autre par la commutation de série).

La mémoire NS ne peut changer son contenu que lors d'une opération variante changement de série.

La progression de la mémoire NL est assurée par un train T1 Le train T1 est démarré à la fin du train Tp de la ligne en cours, de telle sorte qu'au moment de l'exécution d'une instruction, la mémoire NL contient l'adresse de la prochaine instruction à exécuter.

Une ligne de programme se présente donc de la façon suivante



Cas du transfert tambour.

T1 est suivi de 2 trains démarrés simultanément.

En premier lieu, Tp de la ligne suivante, en second lieu, un train d'attente To de durée variable, suivi d'un train de transfert tambour égal à 16 T 1.

Ainsi peut se poursuivre le programme pendant l'attente du transfert.

Le train d'attente a pour rôle d'attendre que le tambour présente devant la tête intéressée le bloc sélectionné. Il dure au minimum 1 tour et au maximum 128 tours.

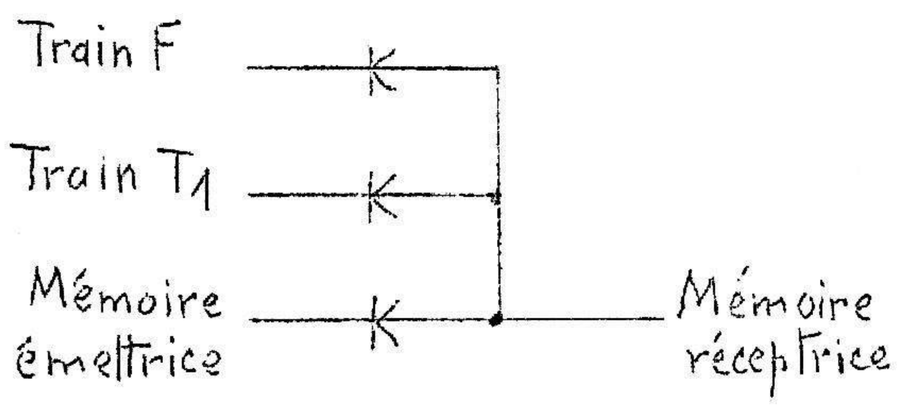
Le train de transfert conditionne le transfert des 16 mots du bloc dans le groupe sélectionné.

Il s'ensuit qu'un transfert tambour a une durée comprise entre 20 tours et 147 tours.

Remarque : Le train d'attente ayant une durée ninimun de 1 tour, dans le cas de nombreux transferts tambour successifs, il sera avantageux de transférer les blocs de 2 en 2 (par exemple tous les blocs pairs, ou tous les blocs impairs).

Filtrage.

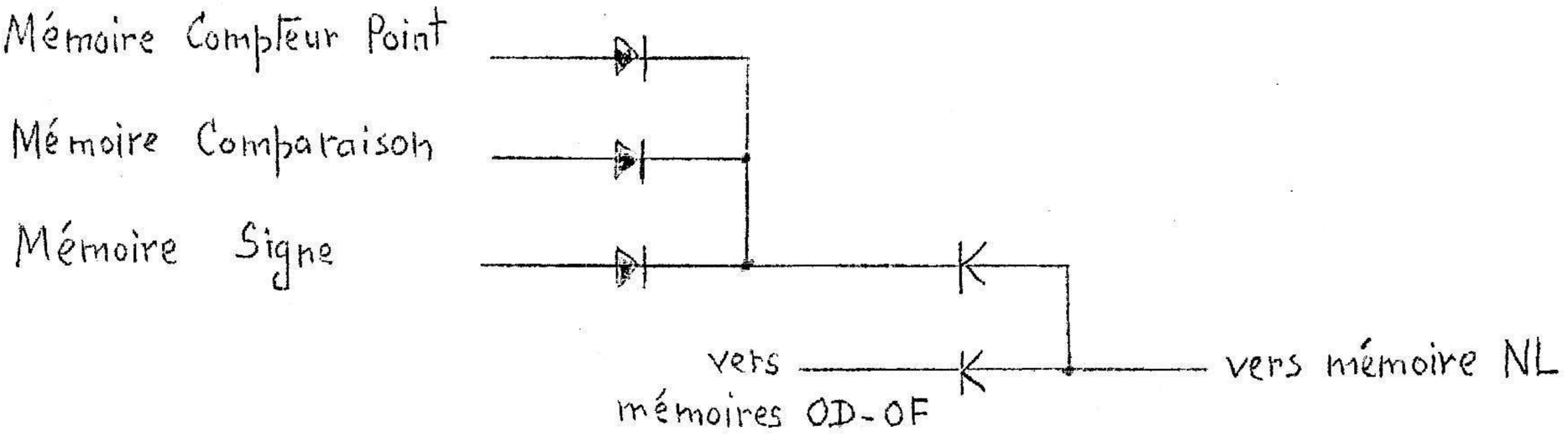
Le filtrage est obtenu à l'aide d'un train F de filtrage qui n'énet des impulsions qu'entre les instants correspondant à l'intervalle OD-OF.



Ce train F est amené sur l'une des entrées d'un conditionneur auquel arrive également le train T 1 de transfert, et le train d'impulsions de la mémoire émettrice.

La commutation vers la mémoire réceptrice n'est établie que lors de la présence simultanée des impulsions de chaque train.

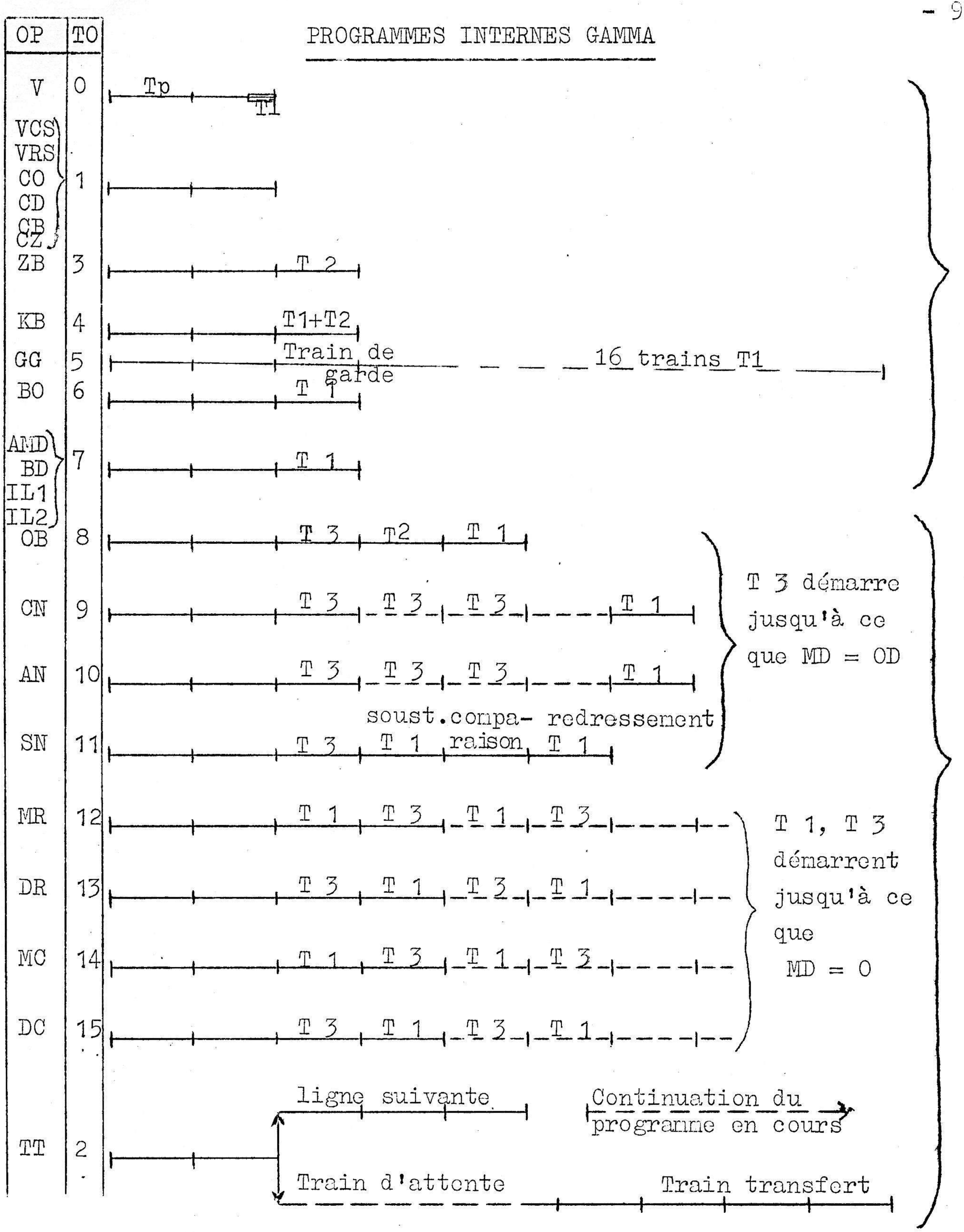
Ruptures de séquences conditionnelles.



Dans les ruptures de séquences conditionnelles, le NL de la prochaine instruction à exécuter est explicitement écrit en binaire dans les codes OD et OF de l'instruction **variante** (Plus exactement, six positions binaires sont utilisées pour le NL, les 2 autres ayant une autre signification).

La variante a pour effet de valider le transfert du contenu des mémoires OD-OF vers la mémoire NL suivant le contenu de l'une des mémoires attachées aux variantes (CP, MC, MS, ...)

PROGRAMMES INTERNES GAMMA



1 tour de mémoire = 172 μ s

Remarque : Tp + T1 = 344 μ s = 2 tours de mémoire en série 0, 1, 2
Tp + T1 = 516 μ s = 3 tours de mémoire en série 3

TEMPS (en μs) DES DIVERSES OPERATIONS

SUR "LE GAMMA E.T." (en SO - S1 - S2)

-:-:-

TO	Tp T1	TO	T1	T2	T3	Total (en tours)	Tps min.	Tps Max.
0 VAR	2					2	344	
1 C. mut.	2					2	344	
2 TB	2	$1 \leqslant z \leqslant 128$	16			18 + z	3 268	25 112
3 ZB	2			1		3	516	
4 KB	2		1			3	516	
5 GG	2	1	16			19	3 268	
6 BO	2		1			3	516	
7 AMD BD IL 1 IL 2	2					2	344	
8 OB	2		1	1	$0 \leqslant x \leqslant 11$	4+x	688	2 580
9 CN	2		1		$0 \leqslant x \leqslant 11$	3+x	516	2 408
10 AN	2		1		$0 \leqslant x \leqslant 10$	3+x	516	2 236
11 SN	2		3		$0 \leqslant x \leqslant 11$	5+x	860	2 752
12 MR	2		$1 \leqslant y \leqslant 90$		$1 \leqslant x \leqslant 10$	2+y+x	688	17 544
13 DR	2		$0 \leqslant y \leqslant 99$		$1 \leqslant x \leqslant 11$	2+y+x	516	19 264
14 MC	2		$1 \leqslant y \leqslant 108$		$1 \leqslant x \leqslant 12$	2+y+x	688	20 984
15 DC	2		$0 \leqslant y \leqslant 108$		$1 \leqslant x \leqslant 12$	2+y+x	516	20 984
15bis DCC	2				$0 \leqslant x \leqslant 12$	2+x	344	2 408

- En série 3 (tableau de connexions) Tp + T1 = 3 tours de mémoire

- Durée des opérations en P.D.F. (en μs)

Opération	Temps min.	Temps max.
Addition	17 888	23 392
Soustraction	20 468	25 972
Multiplication	19 608	33 196
Division	20 984	34 744

DISPOSITIFS COMPLEMENTAIRES

1.- CODIFICATION DES ZEROS -

Afin de faciliter la présentation des résultats, le calculateur est pourvu d'un dispositif de codification des zéros. En effet sur un état, les nombres 120 et 12000 par exemple; ne peuvent se distinguer que par le nombre de zéros à droite, si ces zéros ne figurent pas explicitement dans l'écriture de ces nombres, une erreur sur les ordres de grandeur est facile.

Jusqu'à présent on a considéré le chiffre 0 comme équivalent à l'absence de code dans une position de mémoire : un zéro codé sera représenté par un code 10.

Remarque importante : Le signe moins se représante également par un code 10, on le distingue toutefois d'un zéro codé parce qu'il se trouve toujours à gauche des chiffres significatifs du nombre tandis que les zéros codés se trouvent en général à droite.

Il existe deux façons de coder les zéros :

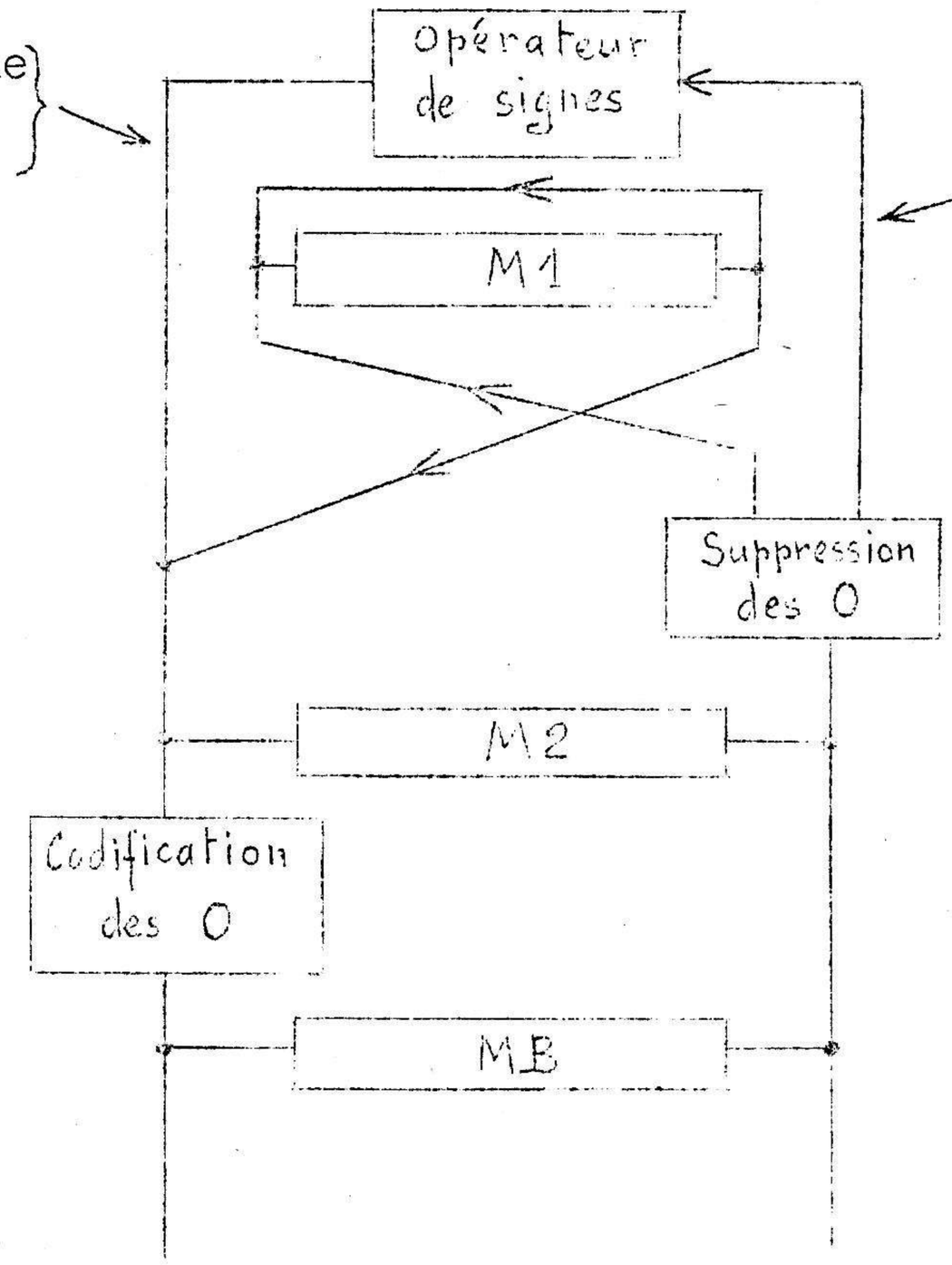
- Zéros à droite : ne sont codés que les zéros à droite du premier chiffre significatif du nombre.
- Tous les zéros : peu utilisée, toutes les positions ne comportant pas d'autre chiffre sont garnies de codes 10.

Ce cas exclue l'utilisation de l'opérateur de signes (impossibilité de reconnaitre le signe moins)

Un commutateur à 3 positions peut selectionner les 3 possibilités :

- Aucun zéro
- Zéros à droite
- Tous les zéros

Mécanisme de la codification des zéros :



1°) Codification des zéros.

Cette codification s'effectue dans l'opération CB à la sortie de la mémoire opérateur. Toutefois la M 2 étant utilisée pour des opérations complètes, les zéros ne sont pas codés lors d'une opération 8. 2 . - . - .

Ce dispositif est naturellement supprimé en calcul binaire.

2°) Suppression des zéros à l'entrée de la mémoire opérateur.

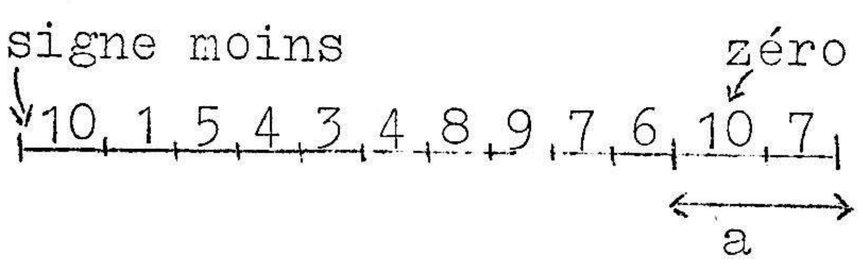
Les codes supérieurs à 9 donnent lieu en CD à un report lors de leur introduction en mémoire opérateur, les zéros codés doivent être éliminés.

Incidences de la codification des zéros sur la programmation en langage machine -

1°) Les zéros ne peuvent être codés en M 2

2°) A l'intérieur d'une mémoire, un nombre est parfaitement défini par son filtrage (OD-OF), il convient donc de ne pas modifier le filtrage sans précaution lors de la codification des zéros à droite.

Exemple 1 :Cas d'un nombre écrit en point décimal flottant avec un exposant inférieur à 10 et zéros codés à droite



l'exposant a est considéré comme négatif (a = -7)

Exemple 2 : es 2 séquences ci-dessous donnent les mêmes résultats sans zéros codés.

1
1	10	- -
8	X	- -
9	X	- -
V	=	

2
1	10	- -
8	X	- -
1	15	- -
9	X	- -
V	=	

Avec les zéros à droite le contenu de M X est différent de celui de M 1 et dans le cas de la séquence 2, la comparaison s'effectuant en CB, les zéros codés arrivent au comparateur et donnent différent .

2. - MISE HORS SERVICE DE L'OPERATEUR DE SIGNES -

Un interrupteur permet de travailler avec ou sans signes

3. - DISPOSITIF 9 ou 12 EN TETE -

Un interrupteur permet la lecture des cartes ligne des 12 en tête. Pas d'utilisation en calcul scientifique.

- Ces 3 interrupteurs sont placés sur le Gamma à gauche du tableau de connexions sous les voyants NL
- Avant l'exécution d'un programme, on doit vérifier leur position.

TABLE DES MATIERES

CHAPITRE 1· -

- Ecriture des nombres en virgule fixe, ...; p. 1
- Ordre début (OD) et Ordre fin (OF), ... p. 1
- Signe. des nombres, ... p. 2
- Mémoire opérateur. Mémoire banale; ..... p. 3
- Opérateur de signes, ... p. 4
- Mémoire décalage, ... p. 4
- Schéma de la mémoire opérateur et des mémoires banales, ... p. 5
- Filtrage, ... p. 6

CHAPITRE 2 . -

LES OPERATIONS ELEMENTAIRES -

1. OPERATIONS DE TRANSFERT -

- Remise à zéro d'une mémoire banale TO = 3 , ... p. 7
- ZB avec AD = 1, ... p. 7
- ZB avec AD = 0, ... p. 7
- Introduction de constantes en mémoire banale KB. TO = 4, ... p. 8
- KB avec AD = 1, ... p 8
- Transfert d'une mémoire banale en mémoire opérateur BO. TO = 6. p. 8
- BO avec AD = 1 , ... p. 9
- BO avec AD = 0 , ... p. 10
- BO avec AD = 0 et OF = 0, ... p. 11
- BO avec AD=0, OD =0, OF= 0, ... p. 11
- Transfert du signe de mémoire banale vers MS 1, ... p. 12

OPERATIONS DE TRANSFERT PARTICULIERES. ...

- Transfert de mémoire banale en mémoire décalage BD. TO = 7, ... p. 12
- Altération de la mémoire décalage. AMD, ... p. 13

II.- LES OPERATIONS AVEC CADRAGE PREALABLE -

A) OPERATIONS DE TRANSFERT

- Transfert de la mémoire opérateur en mémoire banale OB. TO =8, p. 14
- Cas d'un nombre négatif, ... p. 15
- Cas où la mémoire banale contient un nombre, ... p. 16
- OB avec AD = 1, ... p. 16
- Transfert du signe de MS 1 vers une mémoire banale, ... p.17
- Application aux transferts, ... p. 17

B) OPERATIONS LOGIQUES

- Comparaison CN. TO=9, ... p. 18
- CN avec AD = 1, ... p. 20
- CN avec AD = 0, ... p. 20

C) OPERATIONS ARITHMETIQUES

- Addition AN. TO =10, ... p. 21
- AN avec AD = 1, ... p. 22
- AN avec AD = 0, ... p. 22
- Soustraction SN. TO = 11, ... p. 23
- SN avec AD = 1, ... p. 23
- Signe de 0, ... p. 23
- SN avec AD =0, ... p. 24
- Inversion du signe d'un nombre, ... p. 24

CHAPITRE 3 . -

- Multiplication et division, ... p. 25
- Multiplication réduite,MR TO=12, ... p. 25
- Multiplications particulières, ... p. 27
- Mr avec AD = 1 OD = 0, OF = 0 , ... p. 27
- Mr avec AD = 0, ... p. 27
- Mr avec AD =0 et OF = 0, ... p. 27
- Multiplication complète MC. TO = 14, ... p. 29
- Multiplications complètes particulières, ... p. 32
- MC avec AD = 0, ... p. 32
- MC avec AD = 0 et OF = 0, ... p. 32
- Exemple de Multiplication Complète, ... p. 33

- Exemple de multiplications successives, ... p. 35
- Multiplication par un multiplicateur de plus de 12 chiffres.. p. 36
- Division, ... p. 37
- Division réduite DR. TO = 13, ... p. 37
- Division par zéro, ... p. 38
- DR avec AD = 0, ... p. 39
- Division complète DC. TO = 15, ...p. 40
- Exemple de division complète, ... p. 40
- DC avec AD = 0 , ... p. 44
- Division complète avec cadrage DCC, ... p. 44

CHAPITRE 4 .-

CALCUL BINAIRE - APPLICATIONS, ... p. 46

1.- Opérations sur instructions et calcul binaire, ...p. 47
- Multiplication par 2^n, ...p. 49
- Division par 2^n, ... p. 49
- Applications -Structure des instructions utilisant le tableau de code :
  1- Variantes, ... p. 50
  2- Variantes changement de série, ... p. 51
  3- Transfert tambour, ... p. 51
- Transformation d'un nombre écrit en décimal en binaire et réciproquement, ... p. 53
- Programme d'ordres initiaux, ... p. 55
- Appel et renvoi d'un terme a_{ij} défini par 2 indices, ... p. 57
- Rangement d'un a_{ij} ; p.60
- IL Intersection logique, ... p. 61
- IL 1 Intersection logique d'un nombre avec une constante, ... p. 61
- IL 2 Intersection logique de deux nombres, ... p. 62
- Union logique de 2 codes binaires, ... p. 65
- Calcul de la CO et de l'AD pour un terme d'adresse N, ... p. 65
- Transformation code binaire en code alphabétique, ...p. 66

2. - Opérations sur nombres binaires quelconques, ...p. 69

CHAPITRE 5 .-

EXEMPLES DE SOUS-PROGRAMMES ELEMENTAIRES -

1. - Opérations en virgule flottante
- Addition en virgule flottante, ... p. 70
- Soustraction , ... p. 71
- Multiplication , ... p. 71
- Division , ;..... p. 72
- Organigramme d'addition et de soustraction Gamma, ... p. 75
- Sous-programme de cadrage Gamma, ... p. 76
- Multiplication Gamma, ... p. 77
- Tableau P.D.F., ... p. 78

2. - $\sqrt{x}$
- Principe, ... p. 79
- Organigramme, ... p. 80
- Programme, ... p. 81

3. - SIN x
- Principe, ... p. 82
- Organigramme, ...?..... p. 84
- Programme, ... p. 86

CHAPITRE 6. -

PROGRAMME INTERNE ET DUREE DES OPERATIONS ELEMENTAIRES -

- Cricuits de position et de commande, ... p. 90
- Organisation des circuits de commande, ... p. 90
- Conditionneurs et mélangeurs, ... p. 91
- Programme interne, ... p. 92
- Séquence des trains, ... p. 94
- Sélection des lignes de programme, ... p. 9
- Filtrage, ... p. 9
- Ruptures de séquences conditionnelles, ... p.
- Durée des opérations élémentaires, ...

SERIE 3 (TABLEAU)

TO AD OD OF

0 1 63

0 1 2 3 4 5 6 7

8 9 10 11 12 13 14 15

CP

MC

MSB

Opérateur de signes

MS1

MD

AS M1

Opérateur

Comparateur Général

Séquence des trains

Sélection NL T_p . T_l

Décalage T_3

Effacement T_2

Transfert T_1

N° LIGNE

Comparateur de Cadrage

Filtr. Signe

Filtrage

Conditionneur des variantes

Canal Données

M 2

Introducteur

M 3

Canal Résultats

M 7

TO AD OD OF

SERIE 2

SERIE 1

SERIE 0

OCT. 7 OCT. 6 OCT. 5 OCT. 4 OCT. 3 OCT. 2 OCT. 1 OCT. 0

M 8 M 9 M 11 M 12 M 15

GROUPE 3

GROUPE 2

GROUPE 1

GROUPE 0

PISTE

7 6 5 4 0 1 2 3

0 1 63

BLOC

TRANS. TAMB^{R}.

Extracteurs

Introducteurs