UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ALGER

ECOLE NATIONALE POLYTECHNIQUE

departement d'electronique et d'electrotechnique

PROJET DE FIN D'ETUDES

CONCEPTION D'UNE U.A.L. EN VIRGULE FIXE POUVANT RESOUDRE LES OPERATIONS ARITHMETIQUES SIMPLES

Propose par :
Mr B. SHEKHOUTSOV
Professeur à l'ENPA

Etudie' par :
N. SLIMANE

Promotion Juin 1979

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ALGER

ECOLE NATIONALE POLYTECHNIQUE

departement d'electronique et d'electrotechnique

PROJET DE FIN D'ETUDES

CONCEPTION D'UNE U.A.L. EN VIRGULE FIXE POUVANT RESOUDRE LES OPERATIONS ARITHMETIQUES SIMPLES

Propose par :

Mr B. SHEKHOUTSOV

Professeur à l'ENPA

Etudié par :

N. SLIMANE

Promotion Juin 1979

A mon père, à ma mère

A mes frères et sœur

A toute ma famille et amis

Je dédie ce modeste ouvrage

S. NOUREDDINE

- REMERCIEMENTS -

Je tiens à remercier en premier lieu Mr SHEKHOUTSOV mon promoteur dont les conseils et les encouragements m'ont été très utiles à l'élaboration de ce travail.
Que Mr le chef de département d'électronique et d'électrotechnique ainsi que tous les professeurs de l'ENP et de l'université de Constantine qui ont contribué à ma formation veillent bien trouver dans ce modeste ouvrage l'expression de ma gratitude.

Que tous mes amis et camarades de classes trouvent ici l'expression de ma sincère amitié.

Enfin que tous ceux qui ont contribué au tirage et à la reproduction du présent ouvrage trouvent ici l'expression de ma profonde reconnaissance.

TABLE DE MATIERES

Introduction

chapitre 1 . Généralités

11 - Organisation générale d'une calculatrice

12 - Principe de fonctionnement

13 - Présentation des nombres en machine

chapitre 2 . Opérations arithmétiques

21 - Addition et soustraction binaire

22 - Multiplication binaire

23 - Division binaire

chapitre 3. Unité Arithmétique et logique

31 - Schéma Global et microcommandes

32 - Microprogramme général

chapitre 4. Réalisation d'un opérateur de division

Conclusion

Annexe

Bibliographie

— INTRODUCTION —

L'évolution de la technologie à permis de passer de la machine de Pascal qui était basée sur un système mécanique aux machines électroniques et aux gros ordinateurs. Ces dérniers sont essentiellement composés d'une mémoire centrale qui contient programmes et données, d'une unité centrale de traitement qui exécute le programme et d'unités d'entrées - sorties permettant des échanges avec l'extérieur. L'unite centrale est composée à son tour d'une unité arithmétique et logique capable de résoudre les opérations arithmetiques et logiques et d'une unité de contrôle qui génère les microcommandes qui vont positionner les opérateurs de l'unité arithmétique et logique. L'objet de mon travail consiste à étudier une unité arithmétique et logique pouvant résoudre les opérations arithmétiques simples à savoir l'addition, la soustraction, la multiplication et la division; et ceci avec des nbres représentés selon un format fixe (virgule fixe). Pour cela j'éssaierai de résoudre toutes les opérations pas à pas, en choisissant des algorithmes adéquats permettant une solution simple, avec une vitesse moyenne, pour établir ainsi les microprogrammes dont le déroulement génère les microcommandes commandant l'exécution de l'instruction. La microprogrammation est beaucoup utilisée de nos jours parcqu'elle introduit une souplesse et une simplification que l'on ne trouve pas en logique cablée.

CHAPITRE I

11) Organisation générale d'une calculatrice

111) Généralités : Une machine se compose essentiellement de quatre organes :

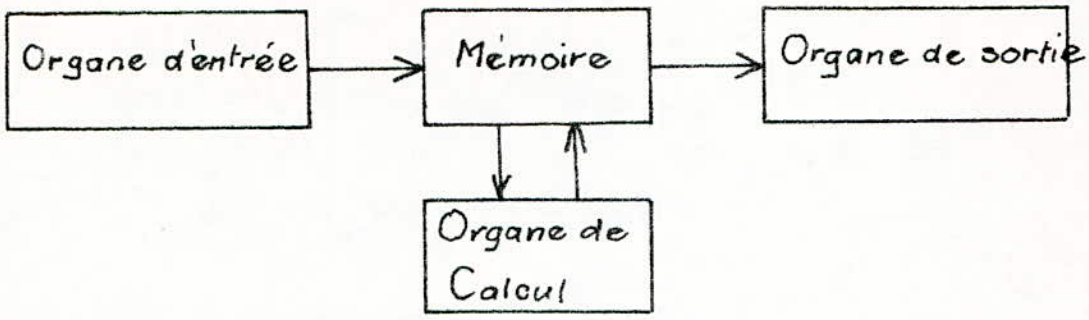


Fig 1 - Structure d'une machine

- Une mémoire centrale qui contient programmes et données
- Une unité centrale de traitement qui exécute le programme
- Des unités d'entrées. sorties permettant des échanges avec l'extérieur

112) L'unité centrale et la mémoire centrale

Le programme est enregistré en mémoire avant le début de son exécution. Cette mémoire, à partir de laquelle le programme peut être executé, est appelée memoire centrale ou mémoire principale. Les différents unités de la machine s'organisent autour de cette mémoire centrale.

La mémoire centrale contient deux types d'informations : D'une part, les instructions du programme (ou informations traitantes) que la machine devra exécuter ; d'autre part, les données, souvent appelées opérandes (ou informations traitées) sur lesquelles la machine effectuera les traitements dictés par les instructions

A ces deux types d'informations, traitante et traitée, correspondent deux unités particulières de la machine : L'unité de contrôle également appelée unité d'instructions ou unité de commande pour les informations traitantes et l'unité arithmétique et logique ou unité de traitement pour les informations traitées

On peut résumer schématiquement les fonctions principales de

l'unité de contrôle : elle extrait de la mémoire centrale la nouvelle instruction à exécuter ; elle analyse cette instruction et établit les connexions électriques correspondantes dans l'unité arithmétique et logique ; elle extrait de la mémoire centrale les données sur lesquelles porte l'instruction ; elle déclenche le traitement de ces données dans l'unité arithmetique et logique, éventuellement, elle range le résultat dans la mémoire centrale.

L'unité arithmétique et logique qui est l'objet de mon étude effectue sur les données qu'elle reçoit les traitements commandés par l'unité de contrôle.

L'ensemble de l'unité de contrôle et de l'U.A.L forme un tout dans la plupart des ordinateurs. On l'appelle unité centrale de traitement ou processeur central.

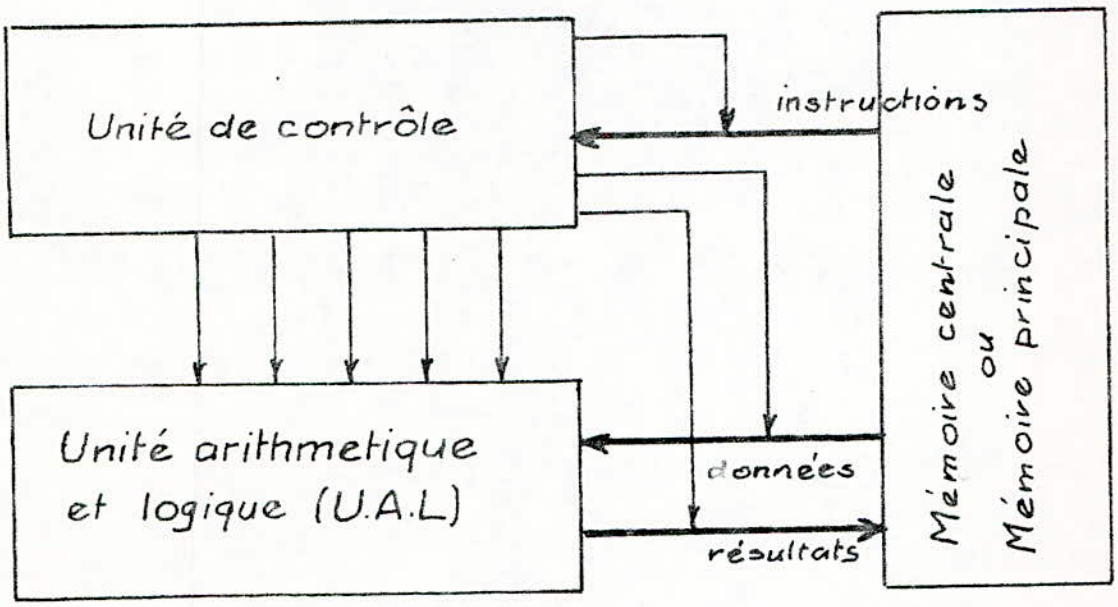


fig 2 - Organisation générale d'une unité centrale

113) <u>Les unités d'échange et les unités périphériques.</u>

Telle qu'elle a été définie jusqu'à présent, la machine doit exécuter un programme initialement enregistré en mémoire centrale, portant sur des données également enregistrées en mémoire centrale, les résultats étant rangés en mémoire centrale au fur et à mesure de leur obtention. Il faut maintenant lui donner les moyens de

communiquer avec l'extérieur : c'est le rôle des unités périphériques. Il existe deux grandes classes d'unités périphériques : les unités de communication (lecteur de cartes, imprimante, unité de visualisation osalloscopique) qui permettent le dialogue avec l'extérieur, et les mémoires auxilliaires (disques, bandes magnétiques) dont les capacités sont très nettement supérieures à celle nécéssairement limitée de la mémoire centrale. Les unités périphériques sont reliées soit à l'unité centrale, soit directement à la mémoire par l'intermediaire d'unités spécialisées dans la gestion des transferts d'informations appelées unités d'échange ou canaux. L'unité de contrôle commande les unités d'echange lorsqu'elle rencontre des instructions d'échange d'informations avec l'extérieur, appelées instructions d'entrée_sortie

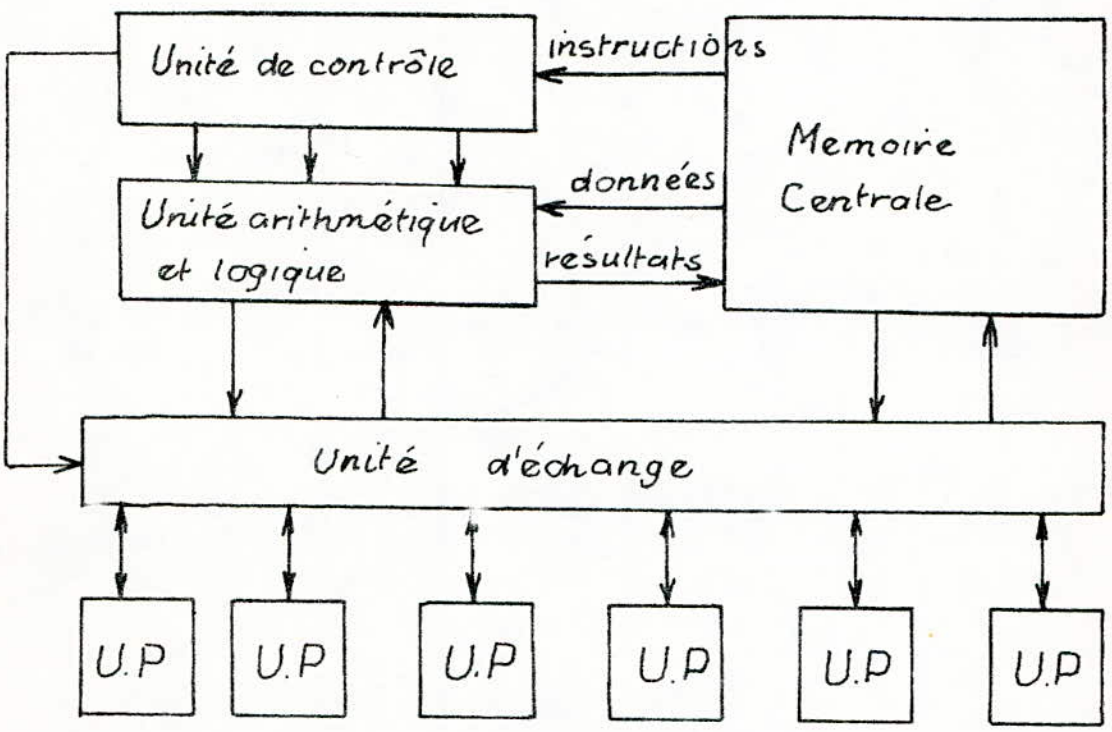


fig 3_ Schéma général d'une calculatrice

Ainsi un calculateur apparait comme un assemblage d'unités distinctes dont le fonctionnement est dicté par le programme contenu en mémoire centrale. L'unité de contrôle commande et contrôle l'exécution des opérations demandées par le programme

Celles-ci sont exécutées par l'unité arithmétique et logique s'il s'agit d'un traitement, par une unité d'échange s'il s'agit d'un transfert d'informations avec l'extérieur

12) Principe de fonctionnement

121) Les registres: On admet qu'au cours d'un traitement les différentes informations, instructions et données, peuvent être temporairement conservées dans des éléments de mémoire capables de contenir une information et appelés registres. Sur un ordre émanant généralement de l'unité de contrôle, une information peut être transférée d'un registre à un autre registre, ce transfert ne modifiant pas le contenu du premier registre

122) La mémoire centrale: Elle peut être considérée comme formée d'un ensemble de cellules, chaque cellule pouvant contenir une information: donnée ou instruction. Les cellules sont numerotées et l'unité de contrôle connait chaque cellule par son numéro, appelé adresse. Elle peut demander à lire le contenu d'une cellule d'adresse donnée ou à écrire une nouvelle information dans une cellule d'adresse donnée. Pour réaliser ces opérations, l'unité de contrôle fournit l'adresse de la cellule concernée dans un registre associé à la mémoire centrale et appelé registre d'adresse ou encore registre de selection de mémoire. Le dispositif de sélection mémoire analyse l'adresse contenue dans le registre de sélection et sensibilise la cellule adressée soit pour une lecture, soit pour une écriture. Dans le cas d'une lecture, l'information contenue dans la cellule adressée est transférée dans un deuxième registre associé à la mémoire et appelé registre d'échange ou registre mot. Dans le cas d'une écriture, il faut préalablement charger ce même registre de l'information à transférer dans la cellule adressée. L'opération de lecture ne détruit pas l'information contenue dans la cellule adressée. L'opération d'écriture la détruit en la remplaçant par une nouvelle information

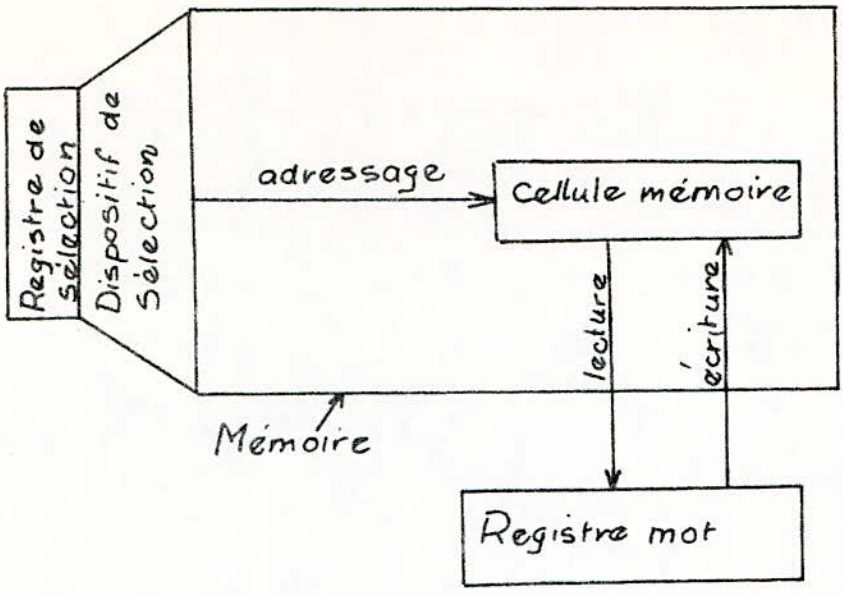


fig 4 - Lecture et écriture en mémoire

123) Le programme : Il se compose des instructions supposées rangés séquentiellement dans la mémoire. Ceci implique que normalement les instructions qui doivent s'exécuter à la suite les unes des autres soient rangées à des adresses sucessives de la mémoire : à la fin d'une instruction d'adresse A, le calculateur enchaine automatiquement sur l'instruction d'adresse A+1, sauf dans le cas d'une rupture de séquence.
Schématiquement, on distingue trois grands types d'instructions :
- les instructions de traitement portant sur des opérandes en mémoire, comprenant essentiellement les opérations arithmétiques et logiques et l'opération de rangement en mémoire
- Les instructions de rupture de séquence permettant de rompre l'enchainement séquentiel des instructions et de passer à une autre partie de programme si certaines conditions sont réalisées.
- Les instructions d'échange permettant les échanges d'informations entre le calculateur et le milieu extérieur

124) Unité arithmétique et logique

1241) Machines à 3 adresses : Pour commander au calculateur une opération arithmétique une addition par exemple, l'instruction doit lui fournir les informations suivantes

- Le type d'opération à réaliser, ici une addition ; c'est le rôle du code opération
- L'adresse de la cellule mémoire qui contient la première donnée ou premier opérande
- L'adresse de la cellule mémoire qui contient le deuxième opérande
- L'adresse de la cellule mémoire ou doit être rangé le résultat. On en déduit la forme de l'instruction qui contient un code opération et trois adresses

Code opération	Adresse 1ère opérande	Adresse 2ème opérande	Adresse résultat

La figure 5_a représente l'unité arithmétique et logique capable d'exécuter cette opération et qui est entourée de trois registres mémorisant les deux opérandes et le résultat. Les machines qui utilisent ce type d'instruction sont appelées machines à trois adresses

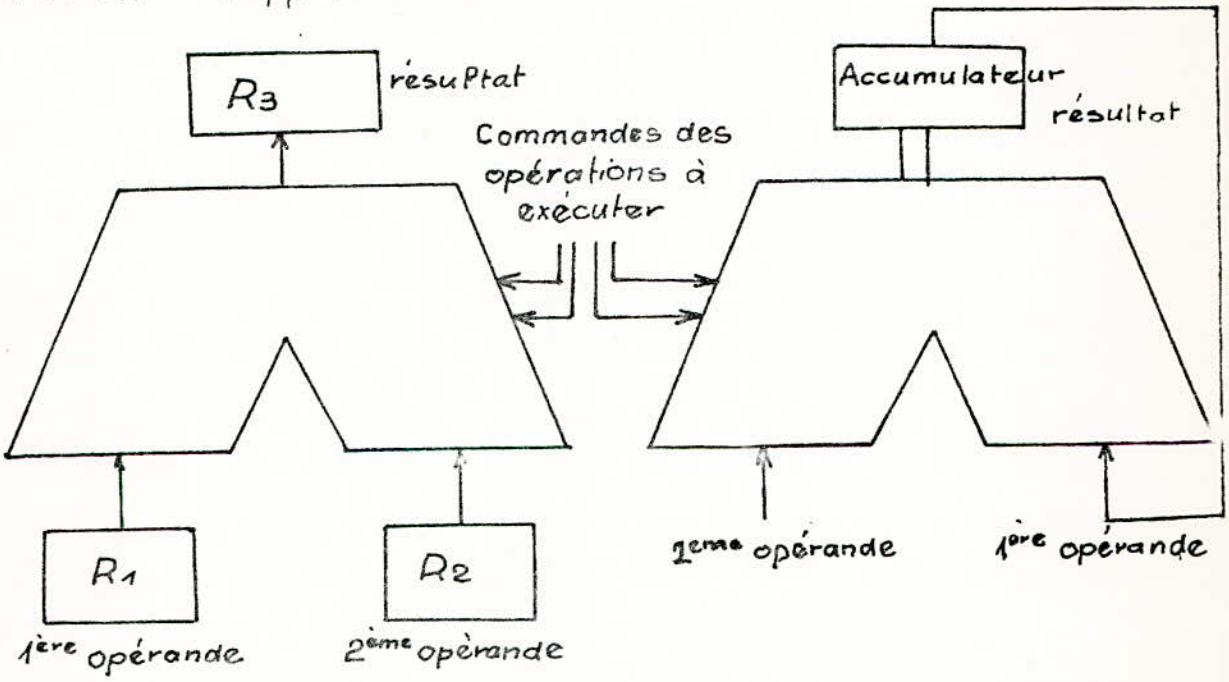


fig 5_a. Machine à trois adresses fig 5_b. Machine à une adresse

Fig 5_Unité arithmétique et logique

1242) Machine à une adresse : Cependant il existe des machines plus utilisées dites à une adresse. Leur unité arithmétique et logique contient un registre particulier appelé accumulateur qui sert à contenir à la fois le premier opérande et le résultat, ce qui permet à l'instruction de ne contenir qu'une adresse : celle du deuxième opérande.

L'opération d'addition par exemple nécessite trois instructions

- Chargement du premier opérande dans l'accumulateur
- Addition du 2ème opérande au contenu de l'accumulateur
- Rangement du contenu de l'accumulateur en mémoire,

Chacune de ces instructions comportera un code instruction et une adresse

Code opération	adresse

Chargement	adresse 1ère opérande
addition	adresse 2ème opérande
rangement	adresse résultat

L'U.A.L est schématisée sur la figure 5b

Les registres R1 et R3 sont remplacés par l'accumulateur. Le 2ème opérande peut être mémorisé pendant l'opération par le registre mot associé à la mémoire

125) L'unité de contrôle

C'est l'unité qui extrait les instructions de la mémoire et qui les analyse. Pour cela, elle doit comporter deux registres:

- a) Un registre appelé compteur d'instruction ou compteur ordinal qui contient l'adresse de la nouvelle instruction à exécuter. Son nom provient de ce que, en dehors des ruptures de séquences, ce registre doit voir son contenu augmenté de un pour passer à l'instruction suivante.
- b) Un registre, appelé registre instruction, qui contient l'instruction extraite de la mémoire. En général ce registre comprend deux parties

une partie contenant le code opération qui définit l'instruction à exécuter (addition, multiplication, branchement...) et une partie qui contient l'adresse de l'opérande. De plus l'unité de contrôle comporte un organe appelé séquenceur qui, après analyse du code opération, fournit les commandes à l'ensemble des unités de la machine (mémoire, U.A.L.) pour leur faire exécuter les différentes phases de l'instruction.

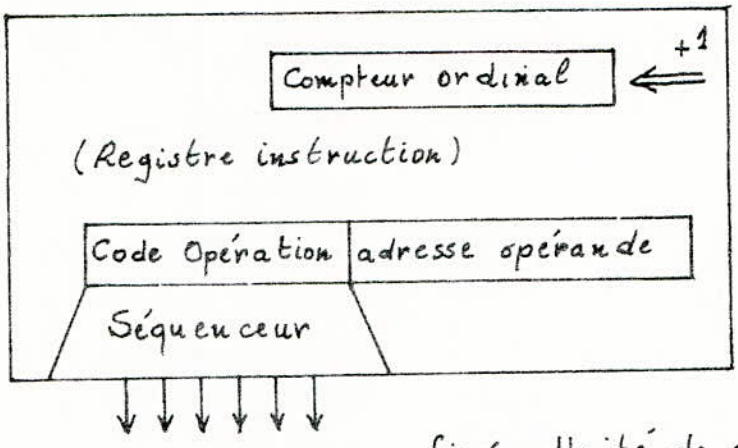


fig 6 - Unité de contrôle -

13) Présentation des nombres en machine.

131) Généralités. Un calculateur est avant tout une machine numérique capable de traiter des informations selon un programme qui lui a été fourni. Il est bien sûr nécessaire de fournir à la machine les grandeurs à traiter; ces informations d'entrée sont appelées données. Toutes ces informations circuleront dans la machine ou seront mises en mémoire selon un code : elles seront traduites en impulsions de tension, chaque impulsion représentant un bit de la grandeur convertie dans le système binaire. Introduit par Leibniz au XVII ième siècle, le système binaire ou système à base 2 est la numération qui apparait comme la plus évidente dans les machines électroniques puisqu'elles utilisent essentiellement des systèmes à deux états d'équilibre. Il faut noter aussi qu'il y a physiquement deux modes de représentation possibles, le mode série et le mode parallèle.

- Dans le mode série, les différents bits représentant une information apparaissent l'un après l'autre dans le temps, sur un même fil, dans l'ordre des poids croissants : c'est une représentation dans le temps.
- Dans le mode parallèle, ces bits apparaissent au même instant sur plusieurs fils affectés aux différents poids binaires : c'est une représentation dans l'espace.

La representation parallèle est exidement plus rapide mais elle nécessite quatre supports d'information (fils) au lieu d'un

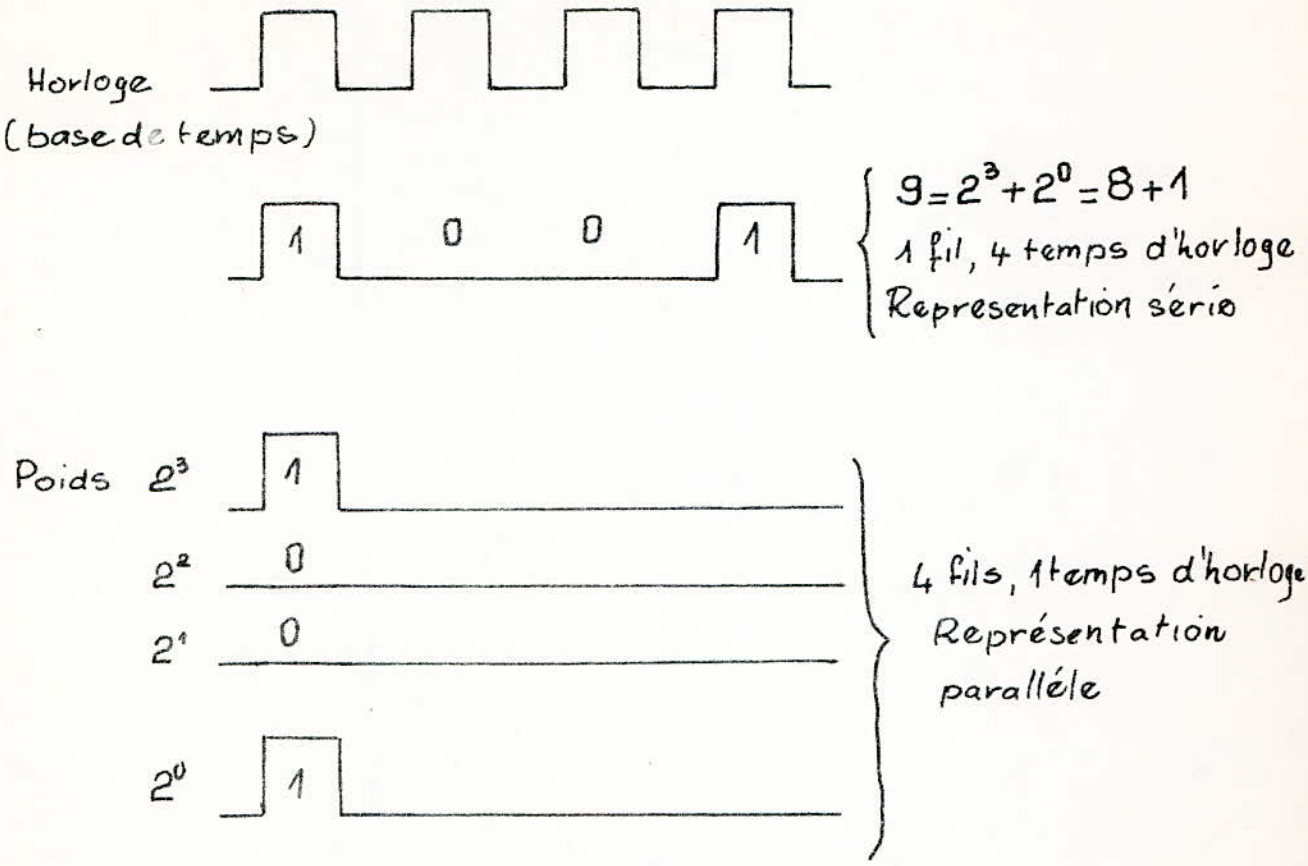


Fig 7 - Représentation du chiffre 9

132) Représentation des nombres négatifs

La solution la plus immédiate est de réserver un digit binaire pour le signe, les autres digits représentant la valeur absolue. La convention généralement adoptée consiste à représenter le signe ⊕ par le digit "0" et le signe ⊖ par le digit "1". Cette représentation impose un traitement spécial du signe et de ce fait, des circuits différents pour l'addition et la soustraction. Cet inconvénient disparait si l'on utilise la représentation des nombres négatifs sous forme complementée. On distingue le complément vrai ou complément à 2 et le complément restreint ou complément à 1. On obtient le complément restreint en substituant les 0 par des 1 et vice versa et le complément vrai en ajoutant 1 au complément restreint ainsi obtenu. Ce dernier est très utilisé car il

facilite la réalisation de certains circuits et présente l'avantage d'avoir une seule représentation du zéro tandis qu'il en existe deux dans le mode de représentation en complément restreint

133) Format

Un opérateur arithmétique ne peut traiter que des nombres qui lui sont presentés selon un format ou un nombre restreint de formats bien définis. La première composante d'un format est sa taille. Les machines mots admettent des formats de longueur fixe; suivant que l'information occupe un mot, un demi-mot, deux mots, etc....
La deuxième composante d'un format est la convention, dans laquelle le nombre est representé ou codé. Notons qu'il existe deux types de formats qui sont: d'une part le format fixe ou format en virgule fixe et le format flottant ou format en virgule flottante.

- Virgule fixe: C'est la façon la plus naturelle d'écrire un nombre binaire dans un mot mémoire: Les nombres sont considerés comme des entiers, le soin de placer la virgule etant laissé au programmeur.

- Virgule flottante: Elle consiste à représenter les nombres sous la forme $SM \times 2^E$ ou S: est le signe du nombre, M la mantisse et E l'exposant. Il y a plusieurs conventions de représentation des nombres flottants binaires en machine. De façon générale, on place les informations les plus significatives en tête: d'abord le signe, ensuite l'exposant, ensuite la mantisse

Signe	Exposant	Mantisse

CHAPITRE II

21) Addition et soustraction binaire

211) Généralités

L'unité arithmétique et logique est composée soit d'une unité capable d'exécuter tout le jeu d'instructions du calculateur, soit de plusieurs unités fonctionnelles ou opérateurs spécialisés chacun dans l'exécution d'un ou plusieurs types d'opérations. Je m'intéresserai, dans ce paragraphe aux opérations qui peuvent s'exécuter entre bits de même poids comme l'addition et la soustraction, réservant pour la suite les opérations plus complexes comme la multiplication et la division. Je précise aussi que je considère des opérateurs avec accumulateur parce qu'ils ne demandent qu'un registre source pour mémoriser l'un des opérandes, l'autre étant mémorisé par l'accumulateur pendant tout le temps de l'opération.

212) Les registres :

Pour pouvoir exécuter des opérations arithmétiques, il faut au préalable mettre les opérandes dans des registres. Pour cela faisons un bref rappel en ce qui concerne ces derniers. Un registre composé de plusieurs bistables permet de conserver une information de plusieurs digits. En effet un bistable est un système électronique possédant deux état stables, qui peut servir d'élément de mémoire

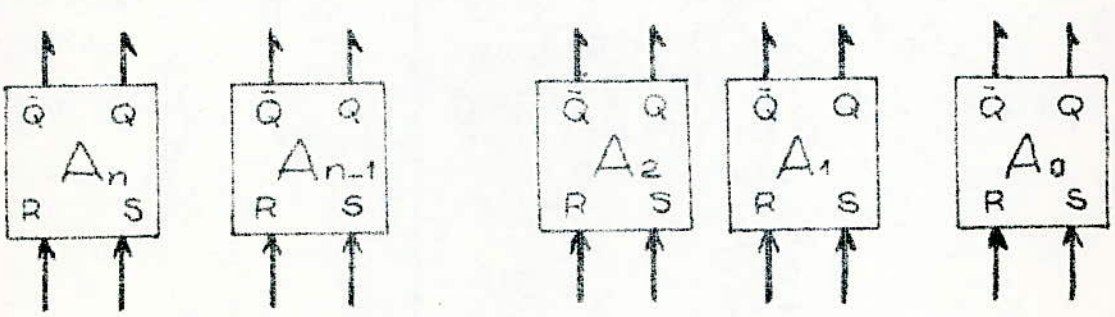
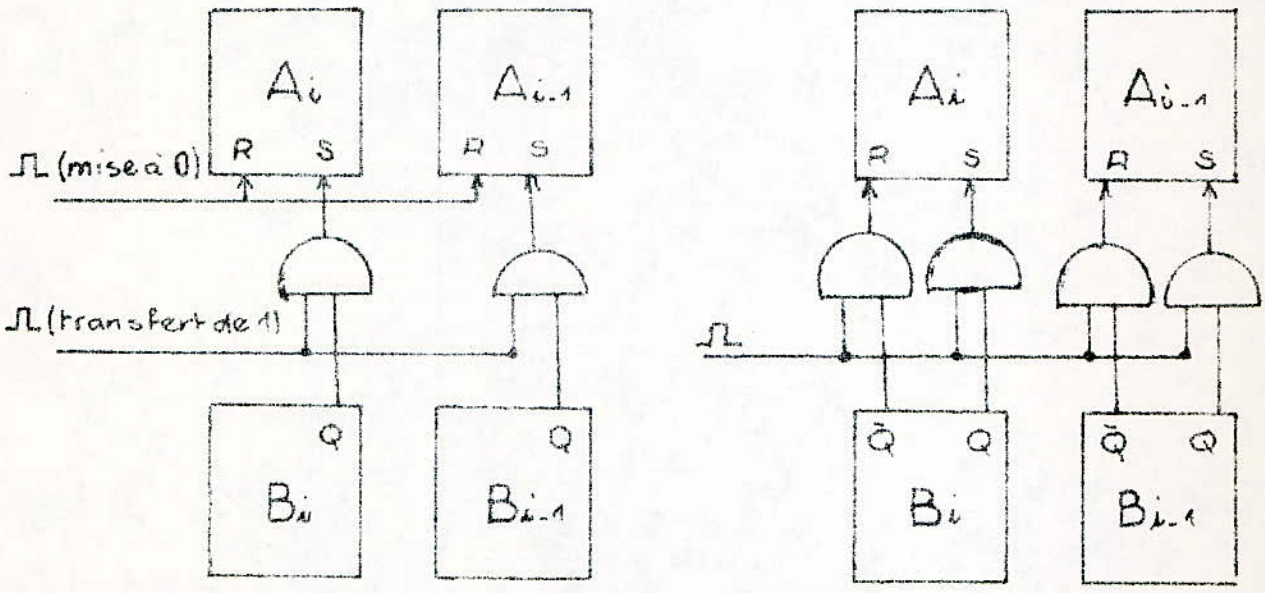


fig 8 - Registre à bistables

- Opérations sur les registres :

— On peut mettre à zéro un registre en envoyant une impulsion sur les entrées correspondantes

— On peut transférer l'information d'un registre à un autre soit par deux impulsions successives, l'une de mise à zéro, l'autre de transfert des 1, soit par entrée forcée des 0 et des 1



a) Mise à zéro préalable b) entrée forcée

fig 9) transferts registre à registre

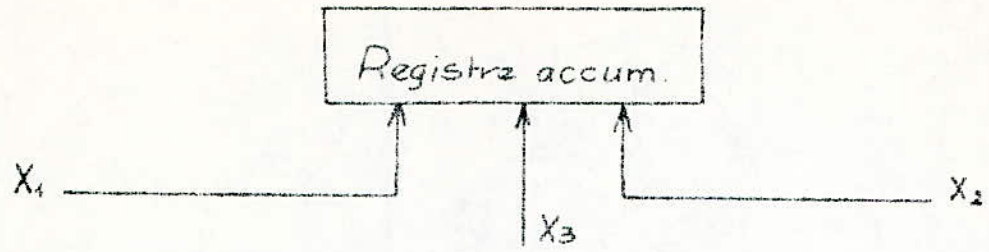
Par la suite le signal de remise à zéro de l'accumulateur sera appelé X_1 et le signal de transfert dans l'accumulateur X_2

On peut aussi transférer le complémentaire d'un registre à un autre en inversant chaque sortie des bascules. Soit X_3 ce signal

X_3 : transfert du complémentaire du registre B dans l'accumulateur

— Le transfert du contenu d'un registre à un autre peut se faire aussi à l'aide de bus. Un bus est un ensemble de fils sur lesquels les informations binaires sont maintenues sous forme de tension par l'un des registres source. Un bus comporte soit un fil par bit, soit 2 fils par bit, l'un portant la grandeur vraie, l'autre la grandeur complémentée. Le bus peut alimenter plusieurs registres destination indépendamment ou simultanément

Pour le moment on peut représenter le registre de la façon suivante



X_1 : remise à zéro du registre accumulateur

X_2: transfert d'un nombre dans le registre

X_3: transfert du complémentaire d'un nombre dans le registre

213) Addition et soustraction de 2 nombres dans le code du complément à 2

2131. Introduction : L'exécution des opérations d'addition et de soustraction peut se faire soit en valeur absolue et signe des nombres, soit en complément à 1 ou soit en complément à 2. J'ai utilisé cette dernière méthode parce qu'elle représente l'avantage de remplacer la soustraction Acc − B par l'addition Acc + B* si (Acc) et (B) sont 2 nombres positifs. Donc l'unité A et L possédera un additionneur qui pourra exécuter les deux opérations ce qui permet une économie de matériel (pas de soustracteur) et un encombrement moindre

En plus dans le code complément à 2 l'exécution des opérations d'addition et de soustraction est très simple comme on le verra par la suite

2132. Algorithme : Supposons que les 2 opérandes sont placés dans 2 registres, le premier terme dans l'accumulateur Acc, le deuxième dans le registre B. Les nombres sont introduits dans les registres de telle façon que les digits de plus bas poids soient à droite. La position la plus à gauche est destinée à contenir le signe du nombre dans le cas des nbres algébriques. Etudions quelques exemples pour pouvoir tirer des règles. Dans un souci de plus grande compréhension je vais raisonner sur des nombres binaires de 4 bits + 1 bit pour le signe

Exemple 1 :

Addition de deux nombres négatifs A = −9 et B = −4

Registre Acc (A=-9)	A*	1	0 1 1 1
Registre B (B=-4)	B*	1	1 1 0 0
Addition $X^* = A^* + B^*$		1 1	0 0 1 1

$A^* = 2^4 - 9 = 16 - 9 = ?$ $B^* = 2^4 - 4 = 16 - 4 = 12$

Notre addition nous a conduit à ajouter $2^4 - 9 + 2^4 - 4 = 2^5 - (9+4)$. Pour que le résultat $X^* = A^* + B^*$ soit exact, il faut retrancher 2^5, ce qui revient à ignorer le bit de poids 2^5, le résultat est donc $X^* = 110011$

$$|X| = \overline{X}^* + 1 = 1100 + 1 = 1101$$

Le module de X est 13 ce qui est bien le résultat cherché

Exemple 2 :

Soustraction de nombres négatifs A=-4 B=-9

Registre Acc (A=-4)	A^*=	1	1 1 0 0
Registre B (B=-9)	B^*=	1	0 1 1 1
	$X^* = A^* - B^*$=	0	0 1 0 1

Le résultat est positif on a directement $X^* = X = 5$. On dit que le résultat apparait en clair.

On peut éviter la soustraction en procédant de la façon suivante : la soustraction A-B est remplacée par l'addition A+(-B) soit $A^* + (B)^*$, ce qui revient à dire que le signe de la soustraction a pour seule conséquence de complémenter à 2 le nombre qui suit ce signe, même si ce nombre est déjà négatif.

Ces exemples nous permettent d'établir les règles suivantes :

- Le bit signe est considéré comme un bit ordinaire pour les opérations d'addition et de soustraction.
- Tout report (ou retenue) au delà du signe est systématiquement ignorée
- Pour une addition on fait ADD=1 SUB=0
  Pour une soustraction on fait SUB=1 ADD=0
- Si le bit du signe = 0 le résultat est positif et apparait en clair

Si le bit du signe = 1 le résultat est négatif et apparait en complément à 2
Je peux donc ajouter à mon U. A et L 2 autres signaux qui sont

X_4 : addition en l'accumulateur

X_5 : Soustraction en l'accumulateur

Exemple 3 :

Soit $A = +12$; $B = +8$; $X = A + B$

$A^* =$	0	1 1 0 0
$B^* =$	0	1 0 0 0
$X^* = A^* + B^* =$	0 1	0 1 0 0

Le bit du signe = 1 ce qui veut dire que $X = A + B$ est négatif. Or X est manifestement positif. Nous remarquons qu'il n'y a pas de report au delà du bit signe

Exemple 4

Soit $A = -9$; $B = 8$; $X = A - B$

$A^* =$	1	0 1 1 1
$B^* =$	0	1 0 0 0
$A^* - B^* =$	0 0	1 1 1 1

Le bit du signe = 0 ce qui veut dire que $X = A - B$ est positif. Or X est manifestement négatif.

Les exemples 3 et 4 nous permettent d'introduire la notion de dépassement de capacité de l'accumulateur.

En prenant plusieurs exemples on constate qu'on aura débordement pour

$C_1 \oplus C_2 = 1$ où C_1 étant le report dans le bit signe
et C_2 étant le report au delà du signe

C_2	C_1	Bascule débordement
0	0	0
1	0	1
0	1	1
1	1	0

D'après les exemples etudiés précédemment, je note que l'indicateur de débordement doit être positionné dans 2 cas:

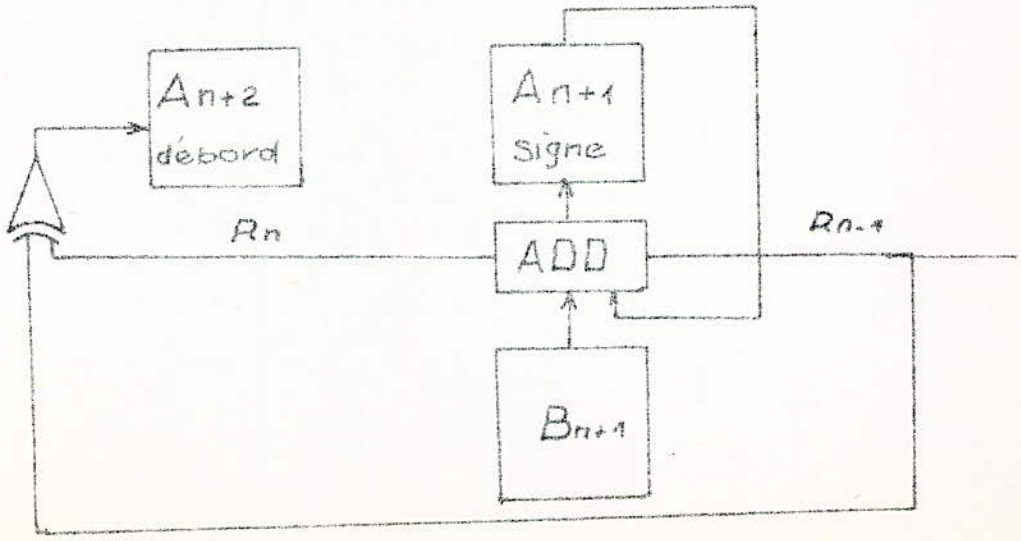
_Il n'y a pas de retenue d'ordre (n) mais il y a retenue d'ordre (n+1) en supposant que notre unité arithmétique et logique traite des nombres de (n+1) bits, le plus à gauche faisant fonction de bit de signe

_Il y a retenue d'ordre (n), mais il n'y a pas de retenue d'ordre (n+1).

Soit Xo le signal de mise à 1 de la bascule de débordement. je peux associer au registre accumulateur un (n+2)ième bit qui servira d'indicateur de débordement.

La figure suivante indique de façon très schématique le cablage du bistable de débordement

fig 10



2133) Conséquences: D'après les exemples etudiés precedemment je constate que l'exécution des opérations encomplément à 2 présente deux caractéristiques essentielles.

- La première consiste en un test du bit signe desregistres, donc il faut prévoir un signal P me renseignant sur l'état du bit signe.
- La deuxième consiste à remplacer une soustraction par une addition algébrique sous réserve de savoir calculer le complément d'un nombre, cequi se fait généralement par complémentation de l'accumulateur de la façon suivante:

On inverse chaque bit puis on ajoute un 1.

L'inversion des bits est obtenue en inversant les sorties Q des bistables composant le registre Acc puis il suffit d'ajouter un 1 à l'aide d'un additionneur type parallèle selon le schéma suivant:

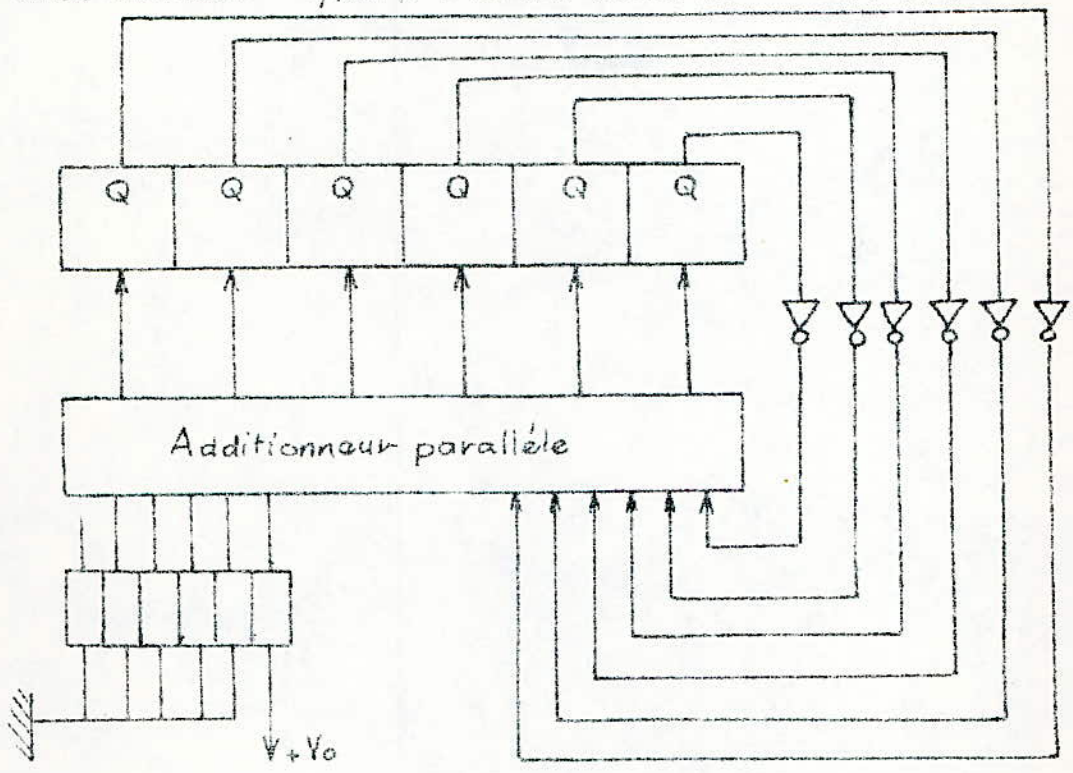


fig 11 - Complémentation à 2 du registre accumulateur

Pour une raison d'économie et d'encombrement, au lieu de munir notre UAL d'un autre étage d'additionneur parallèle, on peut éventuellement utiliser des portes logiques avec des signaux adéquats selon qu'on veut

complémenter le registre accumulateur ou bien faire une addition ou une soustraction.

Soit X_7 le signal d'inversion du registre accumulateur et X_8 le signal de mise à 1 de l'entrée retenue de l'étage additionneur de plus faible poids.

Donc pour complémenter à deux le registre accumulateur, on doit positionner les signaux X_7, X_8 et le signal X_9, X_4 comme indiqué sur la figure 13

Remarque : Le bit signe n'intervenant pas lors de la complémentation de l'accumulateur, on risque de faire des erreurs. Aussi je propose le schéma suivant pour garder le bit signe toujours à la même valeur.

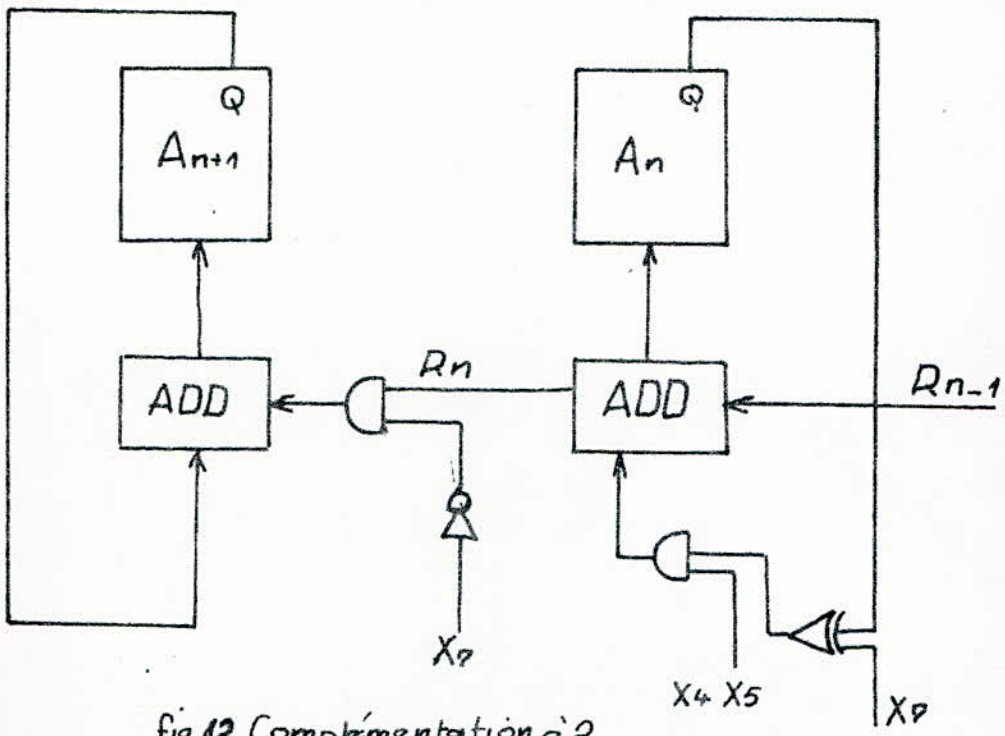


fig 12. Complémentation à 2

Maintenant pour complémenter à 2 le contenu du registre B et l'ajouter au contenu de l'accumulateur, il suffit d'inverser chaque bit du registre B et positionner un niveau 1 sur l'entrée retenue de l'étage d'additionneur de plus faible poids. On doit donc positionner les signaux X_8, X_4 et X_9 comme représenté sur la figure suivante :

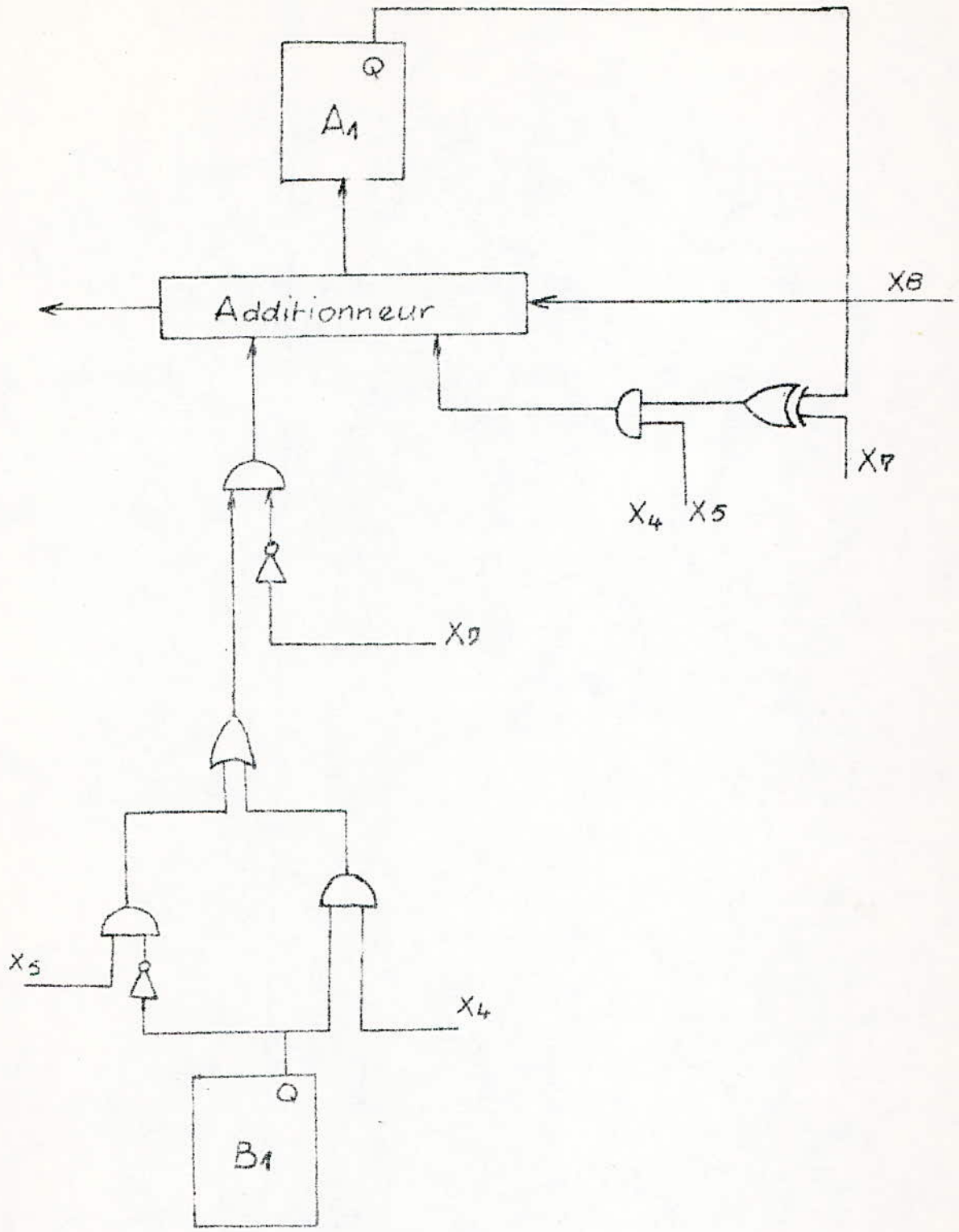


fig 13.

2134) Schémas de principe :

D'après les constatations faites précédement l'UAL pourant resoudre les opérations d'additions et de soustractions pourra être représentée selon le schéma suivant

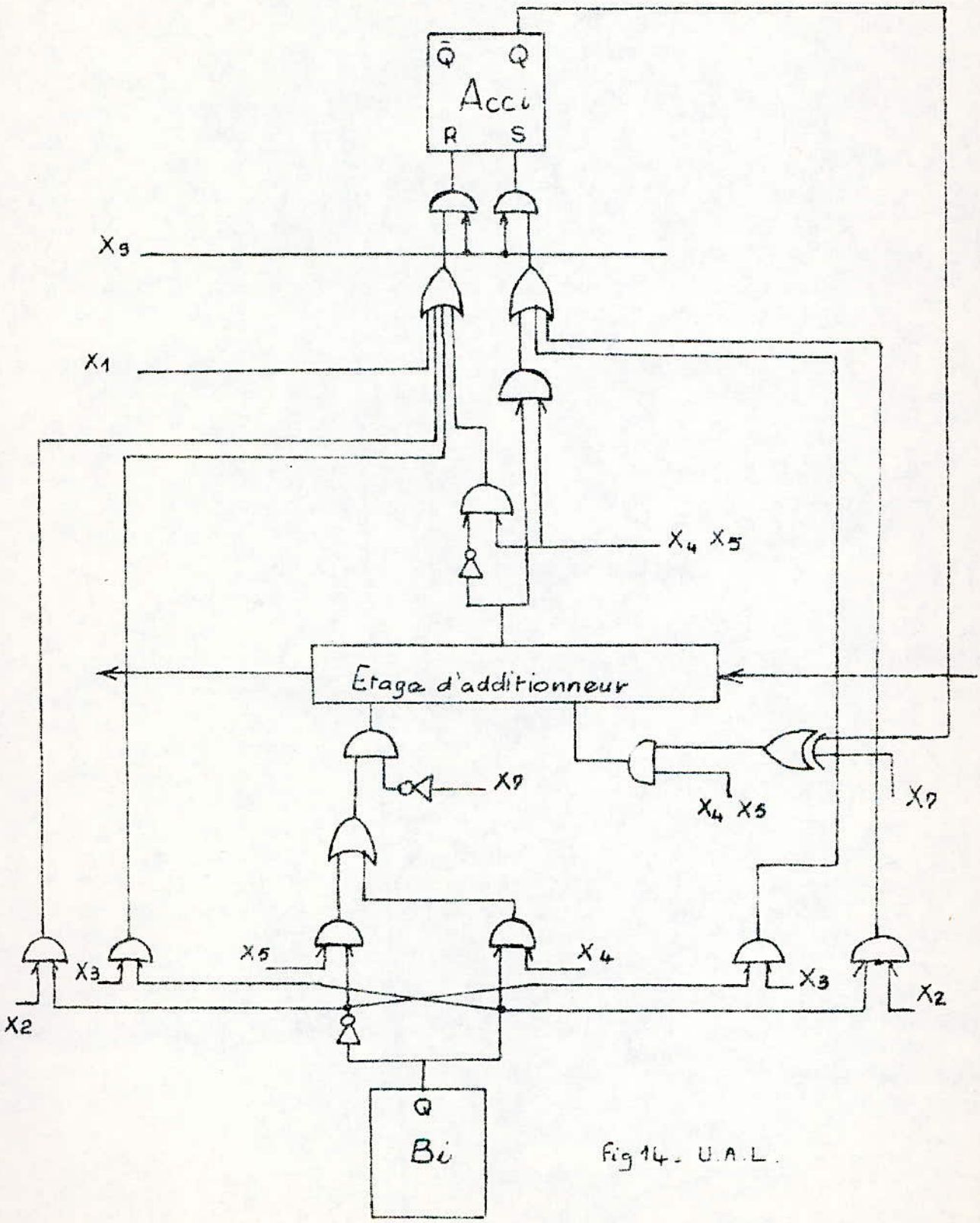


Fig 14. U.A.L.

214) Microprogrammeet présentation des registres

2141. Présentation des registres

Pour l'exécution des opérations d'addition et de soustraction, on a besoin, comme on vient de le voir, de 2 registres, le registre accumulateur et le registre B. Quand l'opération est terminée, un signal d'echantillonnage X_9 vient mettre le résultat dans le registre Accumulateur. Je peux donc représenter mon unité arithmétique et logique avec tous les signaux nécéssaires à l'exécution des opérations d'addition et soustraction. Ces signaux bien entendu seront générés par l'unité de contrôle de la machine

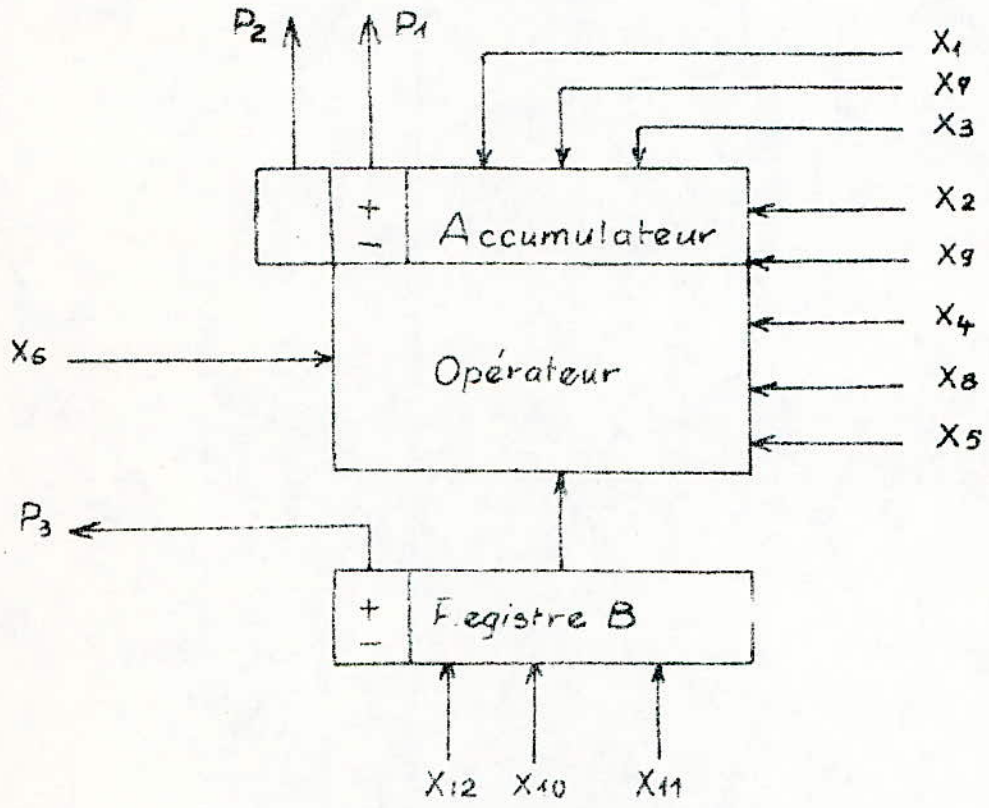


Fig 15. Schéma synoptique de l'UAL

X_1 : Acc:= 0 Remise à zéro du registre accumulateur
X_2 : Acc:= RegB Transfert du contenu du registre B dans Acc
X_3 : Acc:= $(RegB)_{inv}$ Transfert de l'inverse du contenu de B dans Acc
X_4 : ADD
X_5 : SUB
X_6 : SD:= 1 Mise à 1 de la bascule débordement
X_7 : Acc:= $(Acc)_{inv}$ Inversion du contenu de l'accumulateur
X_8 : R_0:=1 Mise à 1 de la retenue de l'étage de plus faible poids
X_9 : Echantillonnage du résultat dans l'accumulateur
X_{10} : RegB:= Y transfert d'un nombre Y dans le registre B
X_{11} : RegB(n) = RegB[n]$_{inv}$ Changement du signe du registre B
X_{12} : RegB:= 0 Mise à zéro du registre B

P_1 : Signal du signe de l'accumulateur
P_2 : Signal du débordement
P_3 : Signal du signe du registre B

2142. Microprogramme

Maintenant on peut élaborer le microprogramme des instructions que la machine doit poursuivre pour l'exécution des opérations d'addition et soustraction.

La microprogrammation consiste à remplacer le séquenceur câblé par un séquenceur microprogrammé. Les instructions du microprogramme sont appelées micro-instructions et l'exécution d'une micro-instruction génère une ou plusieurs microcommandes.

Les calculateurs microprogrammés ont un jeu d'instructions facilement modifiable, le changement ou l'adjonction d'un microprogramme étant beaucoup plus simple que des modifications de câblage.

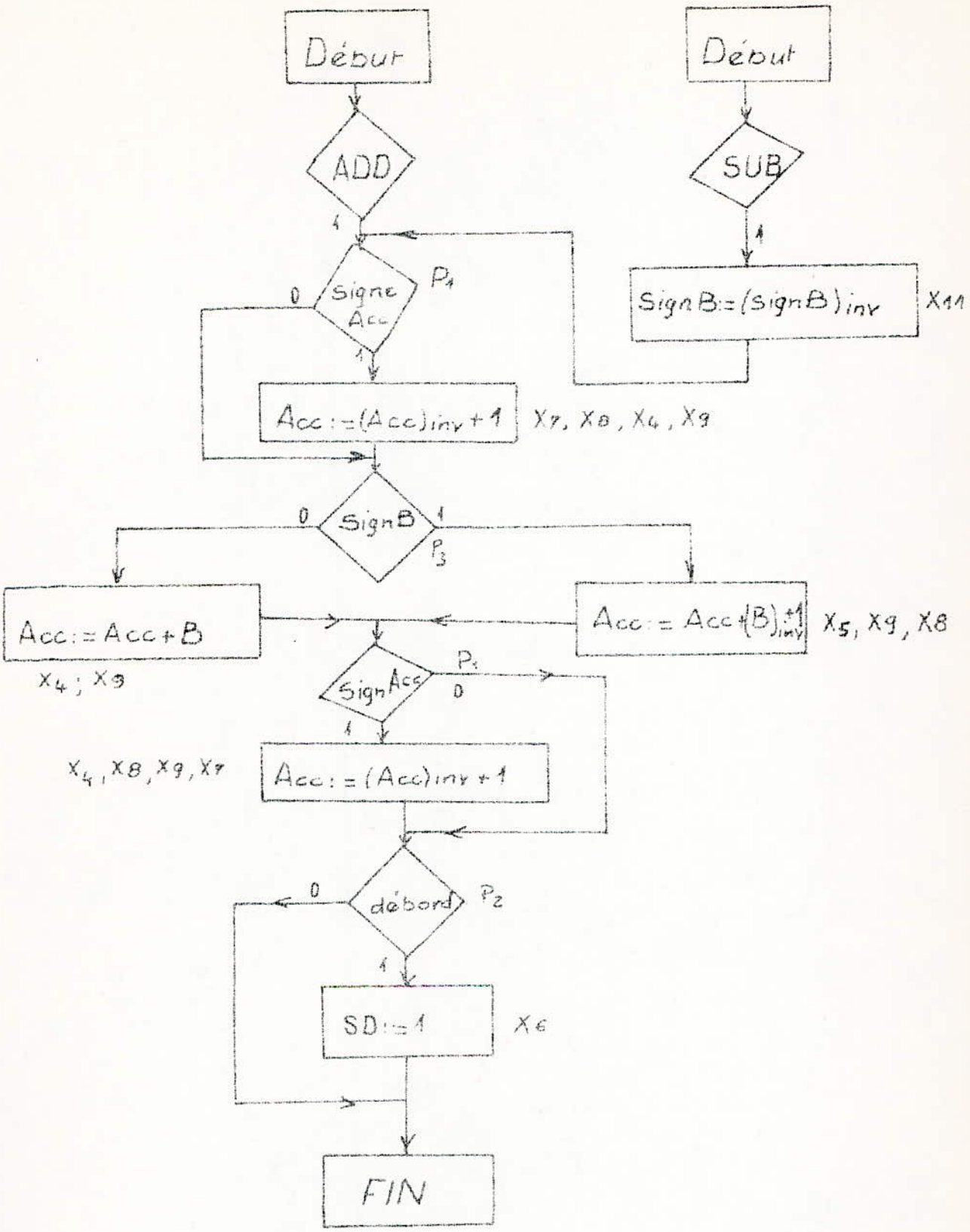


fig 15. Microprogramme de l'addition - Soustraction.

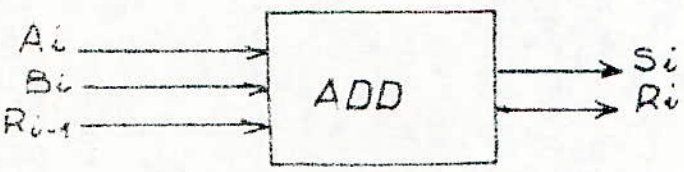
215. Les additionneurs

2151. Généralités :

Le problème est de réaliser un circuit effectuant la somme de deux digits binaires en fournissant le digit somme et le digit de report éventuel, c'est à dire un circuit connaissant la table d'addition binaire. Pour cela on a le choix entre deux modèles d'additionneurs. L'additionneur série ou l'additionneur parallèle. Puisque je veux une unité A et L pouvant résoudre des opérations avec une vitesse pas trop faible, je choisirai un additionneur parallèle. En effet ce dernier présente l'avantage d'être plus rapide que l'additionneur série au prix d'une augmentation du materiel à utiliser

2152. Réalisation de l'étage d'additionneur

Pour réaliser l'addition de deux nombres binaires, il faut tenir compte lors de l'addition de deux digits de rang "i" de la retenue de rang "i-1"
L'opérateur qui réalise l'addition de 2 digits binaires, en tenant compte d'une eventuelle retenue en entrée, s'appelle étage additionneur.
Il possède trois entrées A, B, R' et deux sorties S et R



Dressons le tableau de vérité pour pouvoir tirer les équations qui génèrent notre additionneur ainsi défini

Entrées			Sorties	
A	B	R'	R	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

fig17. Tableau de vérité

De ce tableau, on peut déduire les opérations suivantes:

$$S = R'(\bar{A}\bar{B} + AB) + \bar{R}'(\bar{A}B + A\bar{B})$$

$$R = \bar{A}BR' + A\bar{B}R' + AB\bar{R}' + ABR'$$

On constate que si on a pas une retenue, on aura un ou exclusif entre les entrées A et B mais si on a une retenue, on aura le complément d'un ou exclusif.

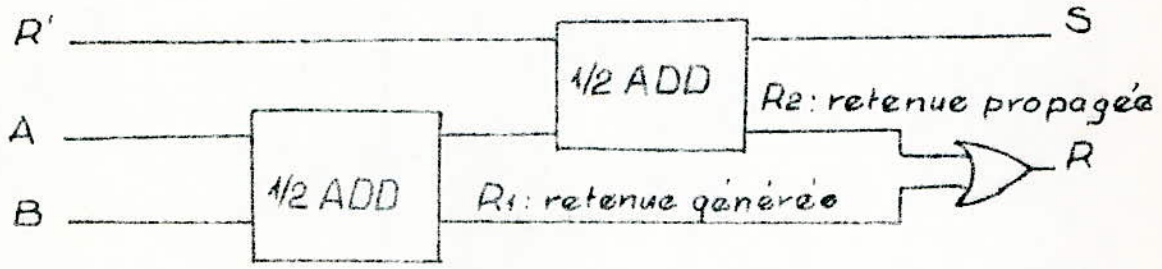
En faisant des regroupements sur le tableau de Karnaught j'obtiens pour la retenue $R = (\bar{A}BR' + ABR') + (A\bar{B}R' + ABR') + (AB\bar{R}' + ABR')$

d'ou $R = (A + B)R' + AB$

La retenue comporte donc deux termes:

$R_1 = AB$ qui représente la retenue générée dans l'étage d'additionneur

$R_2 = (A+B).R'$ qui représente la retenue propagée par l'étage d'additionneur je peux construire un étage d'additionneur de la façon suivante: je fais les sommes de A et B dans un premier demi-additionneur, au résultat S_1 obtenu j'ajoute la retenue R' provenante de l'étage précédent j'obtiens ainsi la somme S et deux retenues: R_1 générée dans l'étage, R_2 propagée par l'étage, je les réunis par un "ou" logique pour obtenir la retenue (fig 18). Comme R_1 et R_2 ne peuvent être présentés simultanément, le circuit "ou" peut être remplacé par une simple connexion



fig(18). Etage d'additionneur réalisé à l'aide de 1/2 ADD

Sur le tableau de vérité de la fig 17 on note que $S = \bar{R}$, sauf si $A+B+R' = 0$ auquel cas $S=0$ ou encore si $ABR' = 1$ auquel cas $S=1$, ce qui permet d'écrire: $S = ABR' + \bar{R}(A+B+R')$

On en déduit la réalisation de l'étage d'additionneur présentée sur la figure suivante:

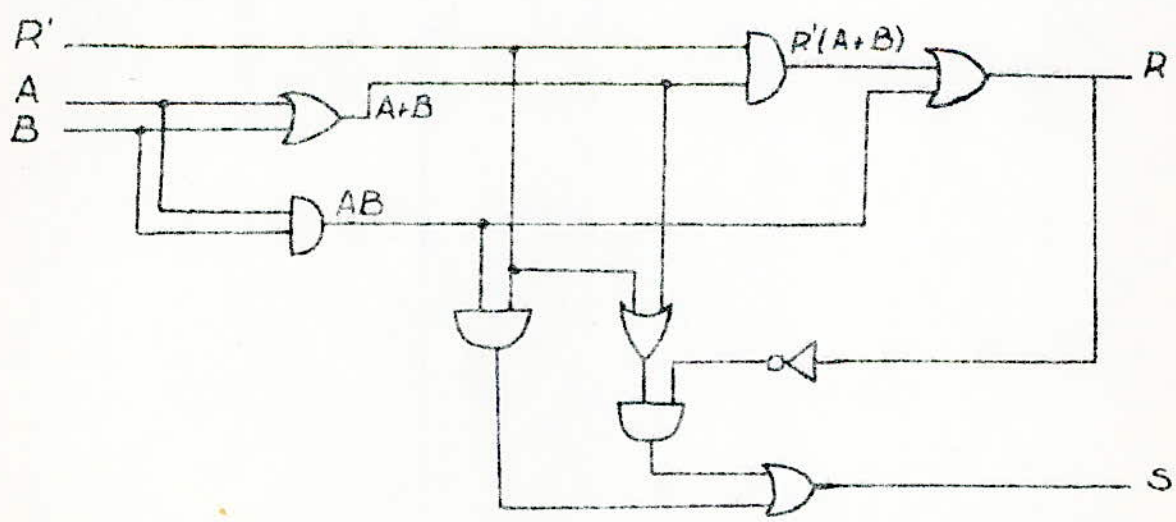


fig 19. Réalisation de l'étage d'additionneur

2153 Additionneur parallèle.

_Schéma structural: Fondamentalement, un additionneur binaire parallèle est formé par une chaine d'étages d'additionneurs, l'étage de rang i transmettant la retenue éventuelle à l'étage de rang i+1. L'étage 0 peut n'être formé que d'un demi additionneur, cependant je préfère utiliser un additionneur complet parce que, comme je l'ai dit avant, j'aurai besoin de positionner un niveau 1 sur l'entrée retenue pour la complémentation à 2 des nombres. La figure suivante représente un élément d'additionneur parallèle cablé entre un registre source et un accumulateur

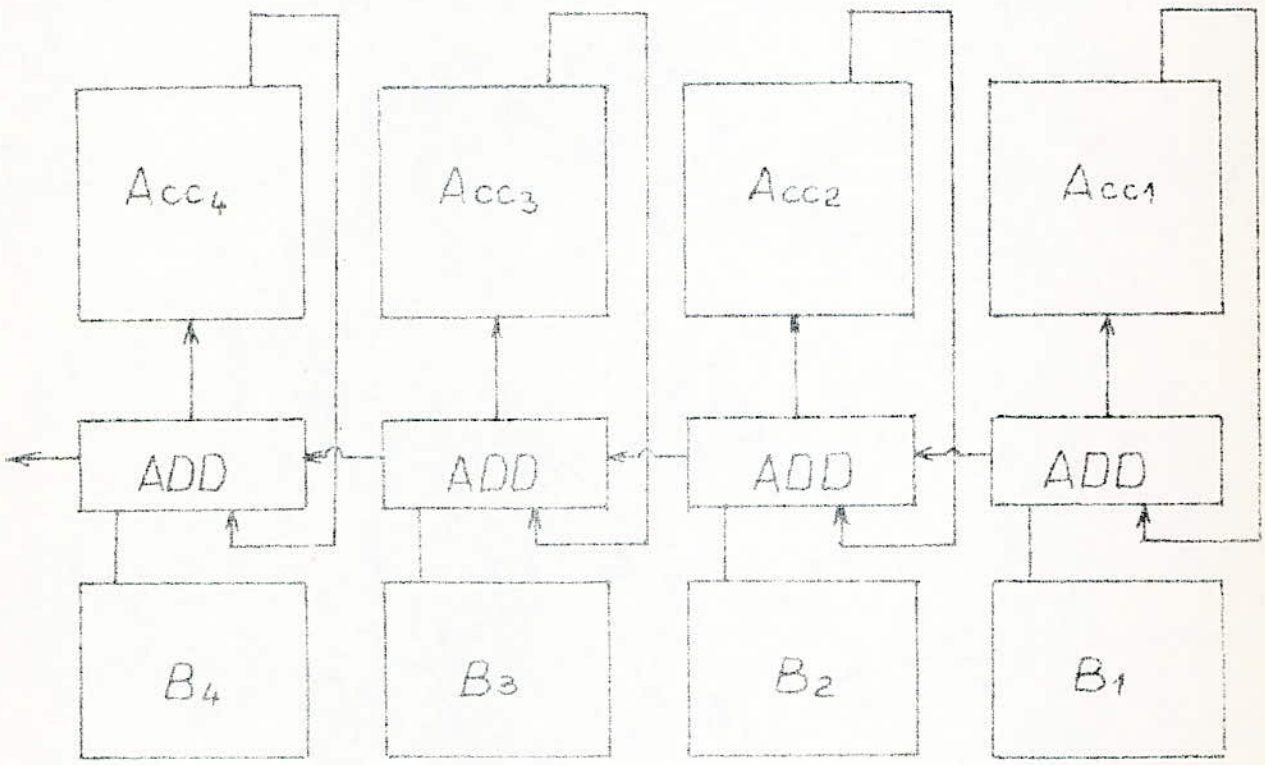


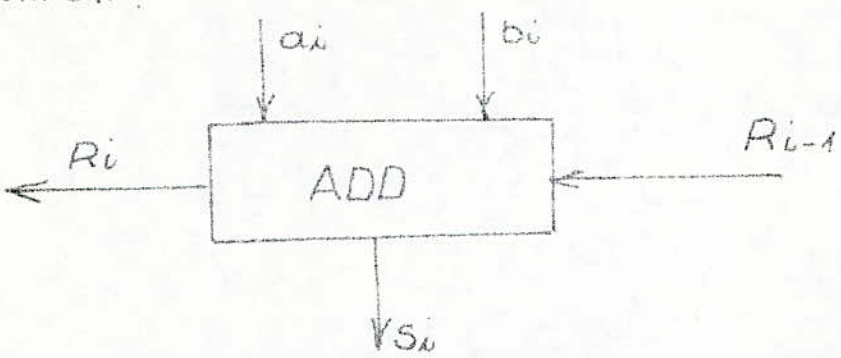
Fig 20. Additionneur parallèle

_Déroulement d'une addition:

Soient 2 nombres binaires de 4 digits: a_1, a_2, a_3, a_4 et b_1, b_2, b_3, b_4 que nous voulons additionner. Ces nombres sont contenus dans les registres Acc et B dont les sorties sont commandées en parallèle. La ligne "départ" donne les informations a_1, a_2, a_3, a_4 et b_1, b_2, b_3, b_4 simultanément aux additionneurs. La ligne "résultat" permet

d'enregistrer la somme dans un nouveau registre ou bien comme on l'a vu précédemment dans l'accumulateur

Cependant il doit exister entre le top "départ" et le top "résultat" un décalage permettant aux additionneurs de générer la retenue si elle existe et de la faire intervenir dans l'addition. A cet effet trois circonstances peuvent se présenter lorsque l'on fournit deux digits a_i et b_i à l'entrée d'un étage d'additionneur.



— La somme $a_i + b_i$ donne un report égal à 1 et un report transmis $R_{i-1} = 1$ changera le digit somme S_i sans changer R_i. Ceci se produit lorsque la quantité $G_i = A_i B_i$ est égale à 1

— La somme $A_i + B_i$ donne une somme nulle et un report transmis $R_{i-1} = 1$ changera le digit de somme S_i sans changer R_i qui restera nul. Ceci se produit lorsque la quantité $G_i = \bar{A}_i . \bar{B}_i$ est égale à 1

— La somme $A_i + B_i$ est égale à 1 et un report transmis $R_{i-1} = 1$ changera le digit de somme ainsi que R_i qui deviendra égal à 1 Ceci se produit lorsque la quantité $P_i = A_i \bar{B}_i + \bar{A}_i B_i$ est égale à 1

les deux premières circonstances sont intéressantes, car elles permettent de séparer les cellules en sous ensembles qui se comporteront indépendamment les unes des autres, lors de la transmission des reports, puisque l'addition de ceux-ci ne modifie pas les reports envoyés sur les étages suivants. Au contraire, les cellules pour lesquelles $P_i = 1$ ne prendront leur valeur définitive que lorsque le report présenté à l'entrée sera définitivement fixé, ce qui peut demander un certain temps, si une chaine de telles cellules existe. Dans ce dernier cas, il y a propagation d'un report sur toute la chaine

et le cas le plus défavorable est celui où la cellule de plus bas poids est du premier type tandis que toutes les autres cellules sont du 3ème type.

2154.) Méthode d'accélération de l'addition

On note sur l'additionneur parallèle représenté figure 23 que le temps de stabilisation des sorties des étages d'additionneur après positionnement des bistables est variable selon les opérandes à additionner : comme je l'ai dit précédemment le cas le plus défavorable est celui ou une retenue générée sur le bit de plus faible poids est propagée jusqu'au bit de plus fort poids. C'est donc la durée de la propagation de la retenue à travers l'ensemble des étages d'additionneurs qui doit être pris comme temps d'addition, l'échantillonnage ne pouvant être fait qu'après l'écoulement de cette durée, si l'on veut être sûr d'avoir obtenu la stabilisation des sorties.

La solution proposée pour accélérer l'opération d'addition est la suivante : l'additionneur est subdivisé en sections (ou sous additionneurs), portant chacune sur un petit nombre de bits. Dans une première phase, les additions sont réalisées au niveau de chaque section, sans tenir compte de la retenue eventuellement générée dans la section immediatement à gauche. Dans une 2ème phase, on propage les retenues. La retenue générée dans la section $i-1$ doit conduire à ajouter 1 à la section i. Cette dernière opération peut conduire la section i à propager elle même, cette retenue vers la section $i+1$ toute l'astuce du procédé revient à détecter, sans avoir à réaliser la propagation dans la section i, si cette retenue existe et doit être transmise à la section $i+1$ si oui, la retenue générée par la section $i-1$ sera transmise à la section $i+1$ en sautant la section i; elle n'aura donc qu'une porte à traverser.

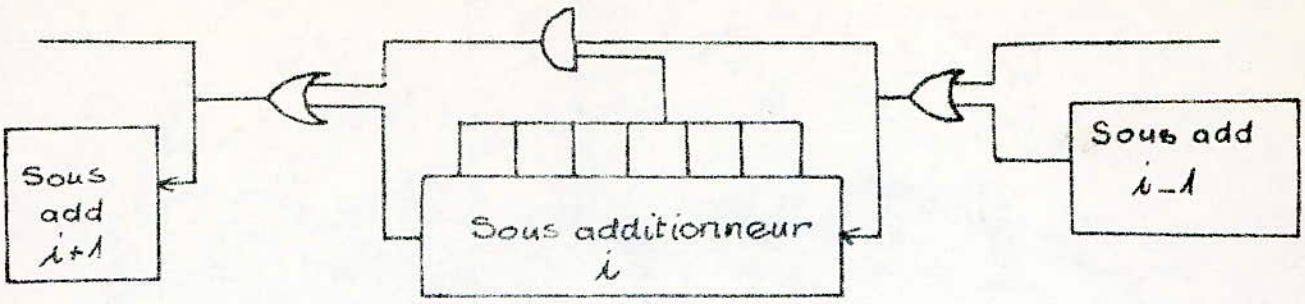


fig 21 - Addition accélérée (echéma simplifié)

Il s'agit maintenant de trouver la condition qui permettra de savoir, sans avoir à réaliser effectivement la propagation, si la retenue doit ou non sauter la section. On se convaincra rapidement qu'elle s'enonce ainsi : la retenue est arrêtée si les deux opérandes présentent deux digits binaires identiques dans au moins une des positions (ou si le résultat obtenu dans la première phase de l'opération possède au moins un digit à zéro).

22) Multiplication binaire

221) Généralités :

Je me propose de décrire les mécanismes couramment utilisés pour réaliser des opérations de multiplication ou division. Bien qu'il existe des algorithmes permettant d'exécuter ces opérations sur des nombres algébriques representés en complémentation, je me bornerai aux opérations sur les valeurs absolues, le traitement du signe étant fait à part. Je partirai de la méthode utilisée lorsque l'on fait l'opération à la main pour en déduire des algorithmes sequentiels simples

222) Multiplication par addition décalage

2221) Algorithme : J'analyserai les régles de la multiplication binaire par comparaison avec celles de la multiplication décimale. Soit alors à multiplier 110 par 101.

Je commence par multiplier le multiplicande 110 par le poids le

plus faible du multiplicateur, c'est à dire 1. Il en résulte un premier produit partiel $110_2 \times 1 = 110$. Ensuite je multiplie le multiplicande par 0 et j'ajoute le deuxième produit partiel au premier mais avec un décalage vers la gauche, ce qui revient à multiplier ce deuxième produit partiel par la base, c'est à dire 2. Je remarque aussi que la multiplication partielle peut être réduite à une addition partielle du fait que chaque bit du multiplicateur ne peut être que 0 ou 1.

La multiplication de 110 par 101 donne un résultat exprimé avec 5 chiffres binaires (11110). Toute fois il faut en prévoir 6, car

$111 \times 111 = 110001$

On doit donc adjoindre, a notre unité "UAL", un complément à l'accumulateur, généralement appelé "Multiplicateur. Quotient" ou par abrévation MQ parce qu'il contient le multiplicateur dans le cas de la multiplication, le quotient dans le cas de la division

2222) Processus. On juxtapose deux registres, l'accumulateur Acc et un registre multiplicateur MQ. Soit alors à multiplier 1001 par 1011, l'opération est initialisée de la façon suivante :

- Changement du multiplicateur dans l'Acc
- Décalage droite sur l'ensemble Acc_MQ, ce qui a pour but de mettre 0 dans l'accumulateur et le multiplicateur dans MQ
- Changement du multiplicande en B.

B	1	0	0	1					
Acc	0	0	0	0	1	0	1	1	MQ

Les régles sont alors les suivantes :

- Chaque fois que la dernière position de MQ sur la droite (poids le plus faible) est occupé par un bit 1, le contenu du registre B est ajouté à celui de l'accumulateur puis l'ensemble accumulateur _MQ est decalé en bloc d'une position vers la droite.
- Chaque fois que la dernière position du registre MQ est occupée

par un bit 0 l'ensemble Acc. MQ est décalé en bloc d'une position vers la droite sans addition de B.

Appliquons ces deux règles à notre exemple

1	Poids faible de MQ = 1 d'où addition du multiplicande puis décalage d'un rang vers la droite	1 0 0 1	
		+ 0 0 0 0	1 0 1 0
		1 0 0 1	1 0 1 1
		0 1 0 0	1 1 0 1
2	Poids faible de MQ = 1 addition puis décalage	+ 1 0 0 1	
		1 1 0 1	1 1 0 1
		0 1 1 0	1 1 1 0
3	Poids faible MQ = 0 décalage d'un rang vers la droite	0 0 1 1	0 1 1 1
4	Poids faible de MQ = 1 addition et décalage	+ 1 0 0 1	
		1 1 0 0	0 1 1 1
	Résultat	0 1 1 0	0 0 1 1

Le poids 2ème étant affecté à la case la plus à droite de MQ, nous avons comme résultat 1 + 2 + 32 + 64 = 99 ce qui est bien le produit de 11 par 9.

Le dernier décalage introduit nécessairement un 0 dans le poids le plus fort de l'Acc. Or si la dernière addition effectuée possède un report 1, la 8ème case de Acc. MQ devra contenir le bit après le dernier décalage. Ceci nécessite que le registre accumulateur comporte 5 cases au lieu de 4

2223.) Signe du produit :

Lorsque les nombres sont représentés en module et en signe, les signes du MD et du MT ne participent pas à l'élaboration du produit. La règle de multiplication des signes donne les résultats rassemblés dans le tableau suivant

SMD	SMT	SR
0	0	0
1	0	1
0	1	1
1	1	0

SMD : Signe du multiplicande

SMT : Signe du multiplicateur

SR : Signe du résultat

Le tableau de vérité nous donne immédiatement $SR = SMD \oplus SMT$

La multiplication de deux nombres de signe quelconque représentés en module et signe s'effectue donc en deux opérations indépendantes :

- L'élaboration du signe du résultat par la fonction logique

$$SR = SMD \oplus SMT$$

- L'élaboration du module du résultat soit R par l'algorithme décrit ci dessus $|R| = |MD| \times |MT|$

223.) Multiplication entière et fractionnaire

Pour pouvoir établir le microprogramme d'exécution de l'opération multiplication, je dois considérer les 2 cas possibles c'est à dire le cas ou les opérandes sont des entiers et le cas ou ils sont fractionnaires.

Pour cela je traite quelques exemples pour pouvoir énumérer tous les signaux qui seront nécessaires à l'exécution de l'opération et établir ainsi le microprogramme d'exécution.

2231) Multiplication entière.

Soit à multiplier 10101 par 11101. La disposition des registres est la suivante. La virgule est supposée immédiatement à droite du dernière bit des operandes.

Reg B	1	0	1	0	1

Reg Acc	0	0	0	0	0	0

Reg MQ	1	1	1	0	1

je commence à tester le dernier bit du registre MQ. Si MQ[n] = 1 j'ajoute le multiplicande au contenu de l'accumulateur et je décale à droite sinon je décale seulement à droite. Pour cela il est nécessaire que les registres Acc et MQ soient des registres à décalages. Je note aussi que pour le registre MQ les décalages seront effectués à partir du bit n Le bit(n+1) de signe ne doit pas participer au décalage pour pouvoir à la fin de l'opération elaborer le signe du résultat a cet effet il faut prévoir un signal P_4 qui nous renseigne sur le signe du MQ. Les décalages à droite seront effectués à l'aide de signaux adéquats de la manière suivante

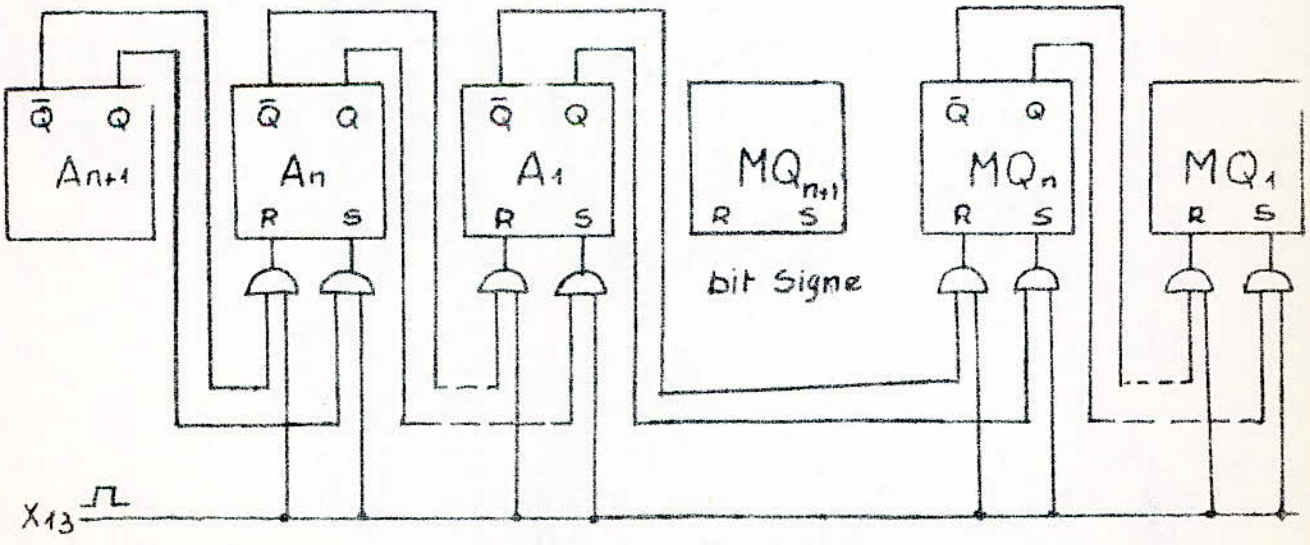


fig 22.) Décalage à droite des registres Acc - MQ

Donc aux signaux précedemment mentionnés pour l'opération d'addition, je dois ajouter un autre signal de décalage à droite X3.
Un signal Ps doit nous donner l'état du bit MQ[1], par ailleurs je note que je n'aurai plus besoin des signaux de complementation parce que les opérandes sont considérés comme positif, le signe étant traité à part.
Notre opération se déroule de la maniere suivante

1	0	1	0	1

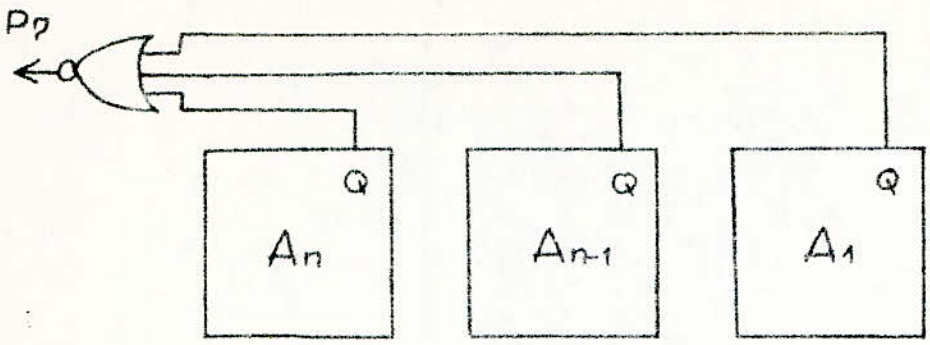
0	0	0	0	0	0

1	1	1	0	1

1er pas	add	0	1	0	1	0	1		1	1	1	0	1
	déc	0	0	1	0	1	0		1	1	1	1	0
2ème pas	déc	0	0	0	1	0	1		0	1	1	1	1
3ème pas	add	0	1	1	0	1	0		0	1	1	1	1
	déc	0	0	1	1	0	1		0	0	1	1	1
4ème pas	add	1	0	0	0	1	0		0	0	1	1	1
	déc	0	1	0	0	0	1		0	0	0	1	1
5ème pas	add	1	0	0	1	1	0		0	0	0	1	1
	déc	0	1	0	0	1	1		0	0	0	0	1

A la fin le résultat se trouve dans le registre Accumulateur MQ. On a ceci $(0100110000 1)_2 = (609)_{10}$
c'est bien le résultat de 29×21

Eventuellement on peut tester le contenu du registre accumulateur



$P_7 = 1$ si $(Acc) = 0$

fig 23. Indicateur de Zéro

A la fin de l'opération si Acc = 0 on peut transférer le contenu du registre MQ dans Acc à l'aide d'un signal de transfert X18. Comme on vient de le voir pour effectuer la multiplication de 2 nbres de 5 bits, il a fallu 5 pas (c'est à dire 5 décalage) pour trouver le résultat. Pour cela on utilise un décompteur de décalages qui sera chargé en début d'opération d'une quantité égale au nombre de digits des opérandes. A chaque décalage, il sera décrémenté d'une unité. A la fin il doit fournir un signal P_6 de fin d'opération.

Soient X_{14} et P_6 les signaux suivants

X_{14} = Changer le DD

$P_6 = 1$ si DD = 0

Pour trouver le signe du résultat, on doit réunir les sorties des bascules signes de MQ et du registre B par un "ou" exclusif fournissant un signal P_8. Si le résultat est négatif on doit mettre à 1 la bascule du bit signe de Acc. Soit X_{17} ce signal

Maintenant, je peux établir le microprogramme d'instruction pour la résolution de la multiplication de deux nombres entiers

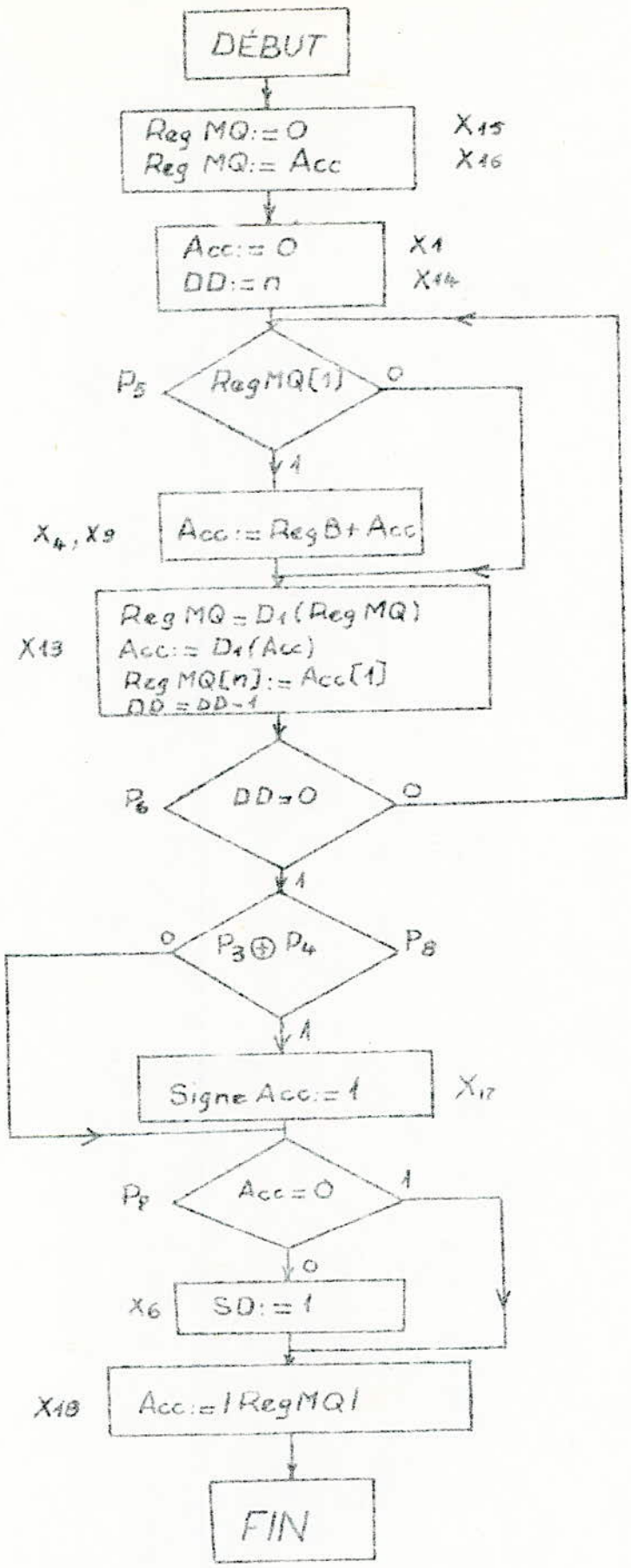


Fig24: Microprogramme de la multiplication entière

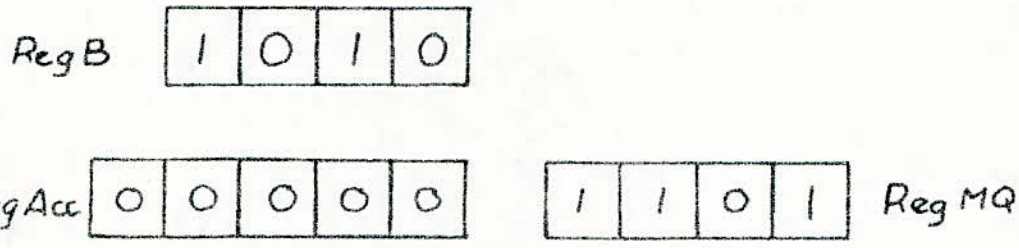
2232.) Multiplication fractionnaire

L'exécution de la multiplication des nombres fractionnaires est basée sur le même principe que celui de la multiplication des nombres entiers. Cependant la virgule est supposée immediatement à gauche du premier bit des nombres et le résultat obtenu à la fin de l'opération pourra exentuellement être arrondi.

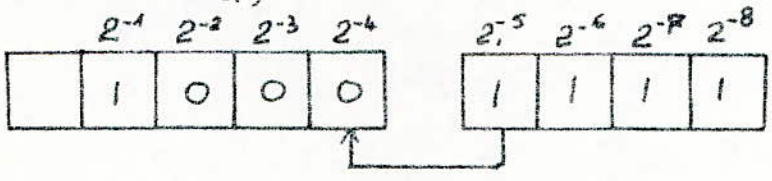
Prenons l'exemple suivant:

Soit à multiplier 0,1011 par 0,1101

La disposition des registres est toujours la même, soit:



Aprés une série de décolages et d'additions, on trouve le résultat dans les registres Acc_MQ, soit:



La virgule étant supposée immédiatement à gauche le résultat sera donc 0,1000 1111.

Au lieu de trainer un nombre aussi large on pourrai l'arrondir par excés en ajoutant 2^{-n} (2^{-4} dans notre exemple) au contenu de l'accumulateur et le résultat 0,1001 sera contenu dans l'Acc en libérant ainsi le Reg MQ Pour cela il faut donc prévoir un signal X19 de mise à 1 du bit 1 de l'accumulateur et un signal Pg pour donner l'état du bit MQ[n]. Le microprogramme d'instruction pour la multiplication des nombres fractionnaires pourra être etabli de la façon suivante:

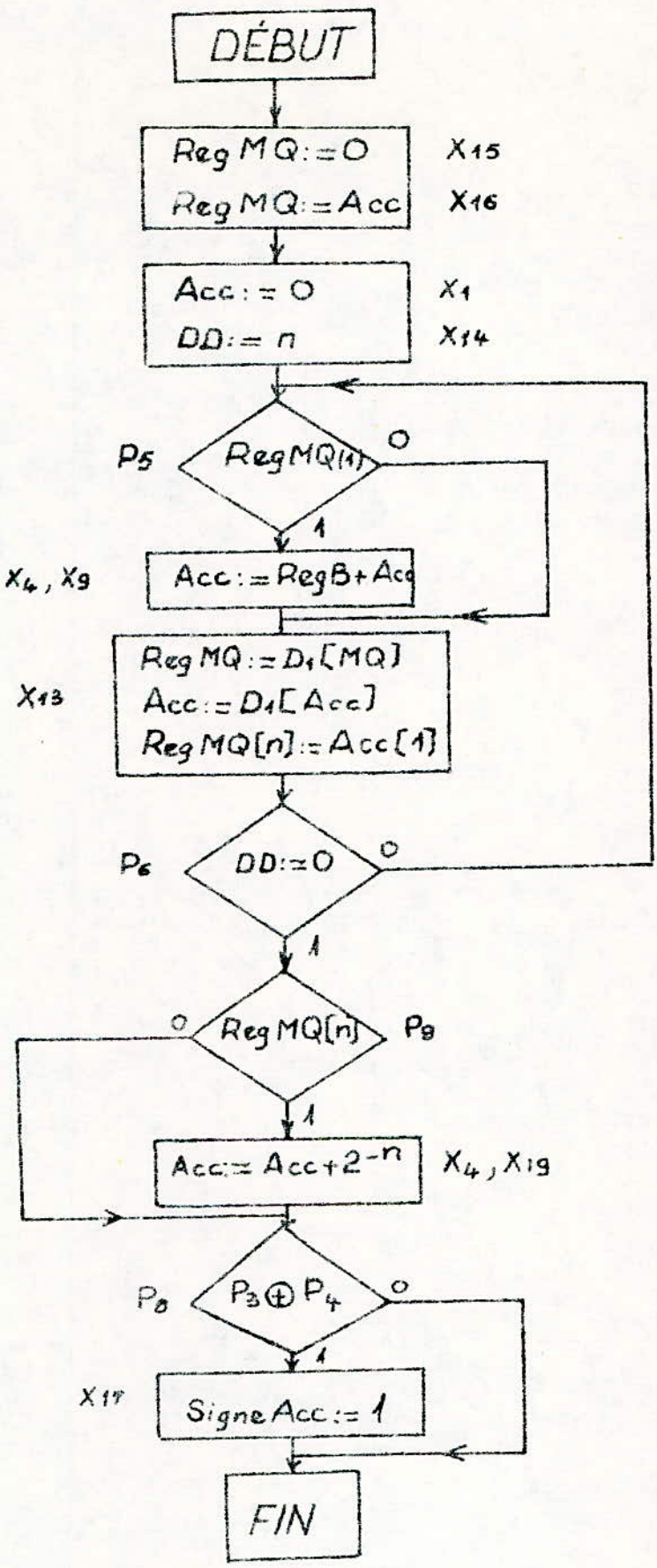


Fig 25 : Microprogramme de la multiplication fractionnaire

224.) Dispositif et microprogramme complet

2241.) Dispositif:

Pour pouvoir exécuter une opération de multiplication de deux nombres, on doit disposer de registres et de microcommandes

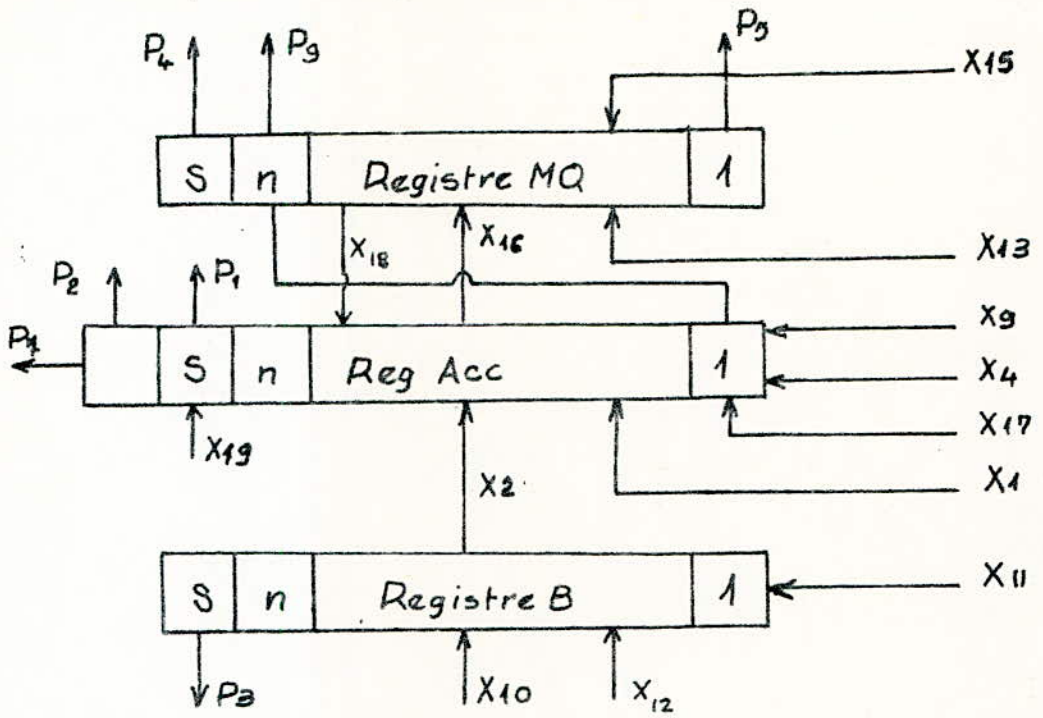
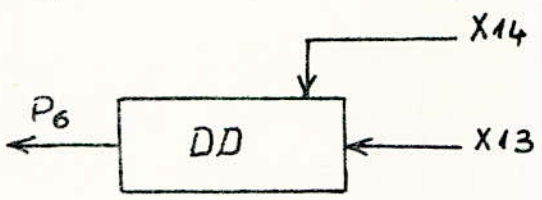


fig 26 - Microcommandes



L'exécution de la multiplication de deux nombres nécessite les microcommandes suivantes:

X13 : Signal de décalage à droite des registres Acc et MQ
X14: DD:=n
X15: Reg MQ:=0
X16: Reg MQ:=Acc
X17: Signe Acc:=1
X18: Acc:=|Reg MQ|
X19: Acc[1]:=1
P5: Test du Reg MQ[1]
P6=1 si DD=0

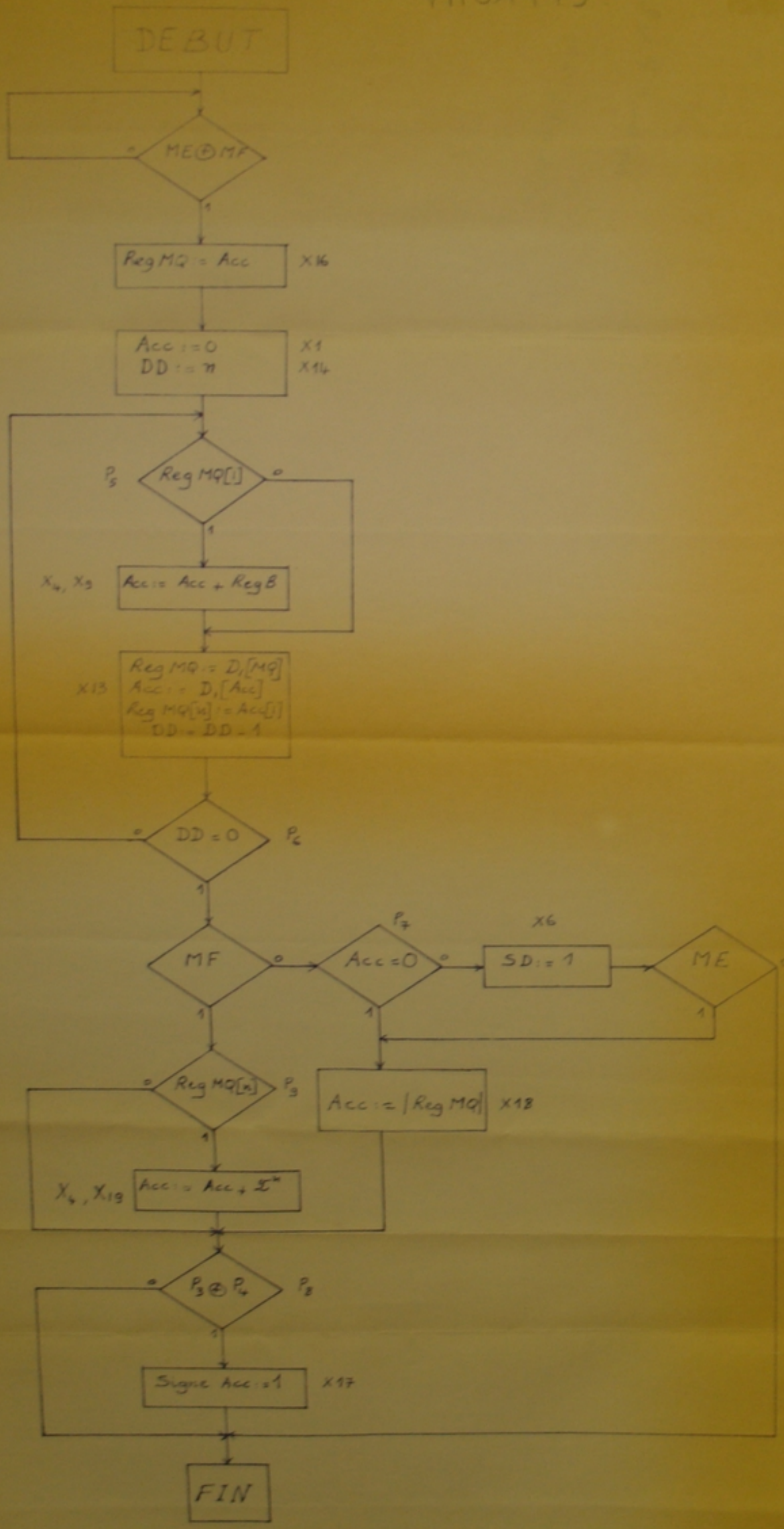


Fig 27 - Microprogramme de la multiplication -

P_7= 1 si Acc=0

$P_8 = P_3 \oplus P_4$

P_9 : test du registre MQ[n]

2242.) Microprogramme général (voir fig 27)

23. Division binaire

231.) Généralités:

Dans le cas de la multiplication, les produits partiels étaient égaux au multiplicande ou nuls, suivant que le bit correspondant du multiplicateur était 1 ou 0. Les produits partiels étaient au fur et a mesure totalisés après décalage d'une position vers la gauche de chaque produit partiel par rapport au précèdent.
Dans le cas de la division, on retranche au dividende soit le diviseur, soit 0, suivant que le bit correspondant du quotient est 1 ou 0. On recommence au pas suivant après avoir fait subir un décalage à droite du diviseur por rapport au dividende.
Ainsi donc multiplication et division sont des opérations très comparables. Il suffit de changer les additions en soustractions et d'inverser le sens des décalages

232.) Division sans restauration.

Elle correspond à un algorithme qui permet d'éviter la phase de restauration après une soustraction donnant un résultat négatif, ce qui entraine un gain de temps. L'algorithme de la division avec restauration partielle nécessite d'effectuer une addition à chaque fois que le reste est négatif pour restaurer le précedent reste partiel. Il faut ensuite effectuer un décalage suivi d'une soustraction. Ces opérations successives nous donnent alors en remarquant que les restes partiels sont contenus dans l'accumulateur Acc

- Soustraction Acc − B
- Addition (si reste négatif) Acc − B + B = Ac
- Décalage d'un rang vers la gauche 2 Ac

- Prestation 2Ac-B

Pour obtenir ce résultat il est possible de procéder autrement en ne restaurant pas le reste. La soustraction, avec reste négatif est suivie d'un décalage d'un rang vers la gauche et l'opération suivante est une addition ce qui produit à 3 le nbre d'opérations à exécuter au lieu de 4

- Soustraction: Acc - B
- Décalage: 2 (Acc - B)
- Addition: 2 (Acc - B) + B = 2 Acc - B

23.21.) Division des nombres supérieurs à un.

Généralement l'opération est initialisée de la façon suivante:

- Changement du dividende dans l'accumulateur
- Décalage droite sur l'ensemble Accumulateur-MQ ce qui a pour but de remettre à zéro l'accumulateur et de changer le dividende dans le registre MQ.
- Changement du diviseur en B.

On procède à un premier décalage d'un rang vers la gauche de l'ensemble Acc. MQ puis on soustrait le diviseur de l'accumulateur

- Si la retenue est égale à zéro on fait MQ[1] = 0 et la prochaine opération à effectuer est une addition: Acc = Acc+B.

Si la retenue est égale à zéro on fait MQ[1] = 1 et la prochaine opération à effectuer est une soustraction. Avant d'effectuer l'addition (ou la soustraction selon la valeur de la retenue) on effectue un décalage d'un rang vers la gauche.

Le processus continue jusqu'à ce que le nombre de décalages effectué soit égal à n si le dividende et le diviseur sont exprimés par n bits. Si la dernière opération effectuée donne une retenue 0 le reste est exact. Sinon il faut effectuer l'opération complémentaire pour restaurer le dernier nombre. Celui-ci est exprimé par les (n-1) bits de plus faible poids de l'accumulateur

Exemple: Soit à diviser 14 par 6

Le diviseur se trouve dans le registre B et le dividende dans MQ

Acc est mis à zéro

	B	Acc	MQ	DD
	0 1 1 0	0 0 0 0 0	1 1 1 0	
- Décalage vers la gauche		0 0 0 0 1	1 1 0 0	3
- Soustraction Acc - B		- 0 1 1 0		
- Acc[n+1] = 1 → MQ[1] = 0		① 1 0 1 1	1 1 0 0	
- Décalage vers la gauche		1 0 1 1 1	1 0 0 0	2
- Addition Acc + B		+ 0 1 1 0		
		① 1 1 0 1	1 0 0 0	
- Décalage vers la gauche		1 1 0 1 1	0 0 0 0	1
- Addition Acc + B		+ 0 1 1 0		
- Acc[n+1] = 0 → MQ[1] = 1		⓪ 0 0 0 0	0 0 0 1	
- Décalage vers la gauche		0 0 0 1 0	0 0 1 0	0
- Soustraction Acc - B		- 0 1 1 0		
- Acc[n+1] = 1 → MQ[1] = 0		① 1 1 0 0	0 0 1 0	
- Addition Acc + B		+ 0 1 1 0		
		0 0 0 1 0	0 0 1 0	
		reste = 2	Quotient = 2	

D'après cet exemple on constate que le bit de plus faible poids de MQ doit être mis à 1 ou à 0 selon que la retenue est 0 ou 1. Le transfert pourra se faire à l'aide d'un signal X20 de la façon suivante

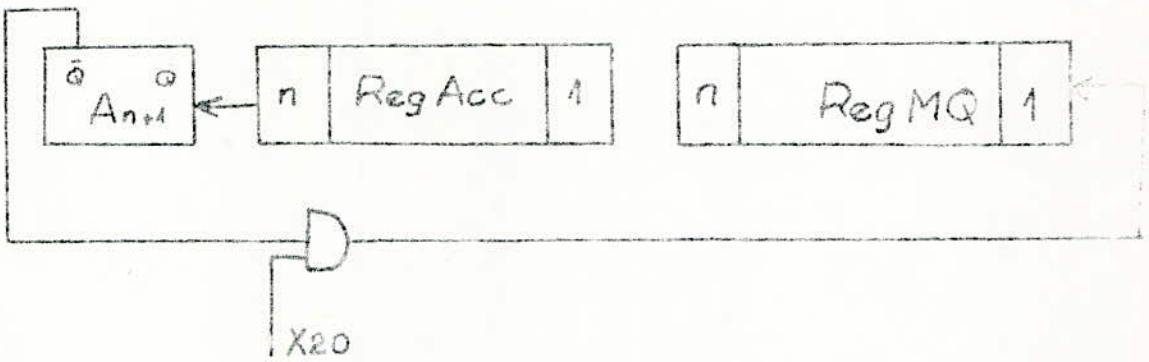


fig 28.

Les décalages d'un rang vers la gauche des registres Acc et MQ sont assurés par le positionnement d'un signal X21 pour Acc et X22 pour MQ

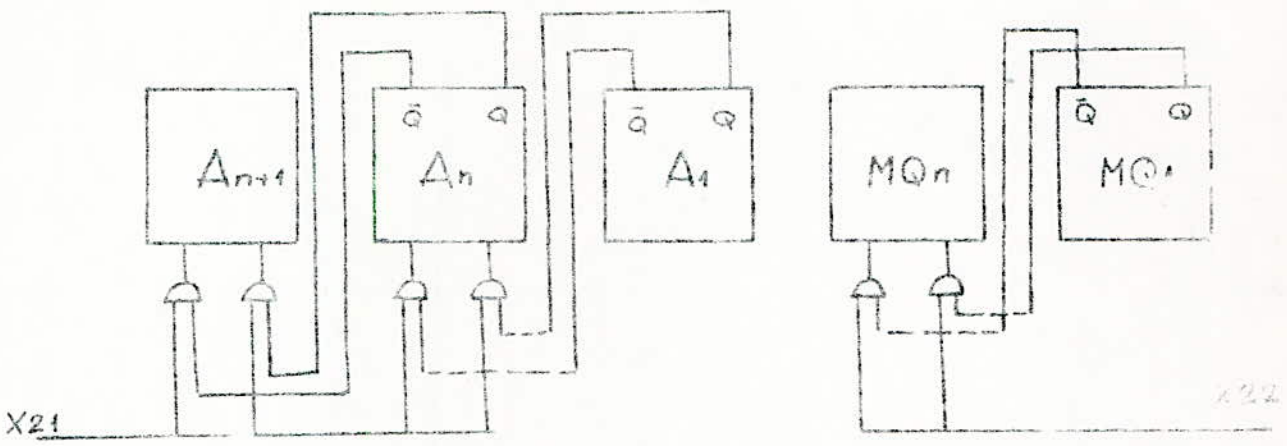


fig 29. Décalage à gauche de Acc et MQ

Le résultat de l'opération précedente, comme on le voit, est formé chiffre par chiffre dans le registre quotient MQ. Notons que l'on peut utiliser un dividende de longueur double du diviseur, les chiffres de plus faible poids étant en MQ. Le dividende pouvant être le résultat d'un produit. Cependant on risque dans ces conditions, de faire une erreur si le contenu de l'accumulateur est plus grand que le diviseur : le quotient excéderait alors la capacité du registre MQ, et ses bits de poids forts seraient perdus. Si le diviseur est bien plus petit que le dividende, lors des premiers décalages à gauche, on risque de perdre les premiers

chiffres du dividende et même il peut ainsi apparaître un 1 en position signe de l'accumulateur, or ceci indique que le résultat de la soustraction est négatif, c'est à dire que le diviseur est supérieur au dividende. On évite cette erreur par une comparaison initiale entre le contenu de l'Acc et celui du registre B (c'est à ce niveau que l'on peut détecter les :/0)
Il conviendrait donc mieux de toujours partir d'un diviseur supérieur au dividende, et pour avoir un résultat le plus précis possible (c'est à dire qu'il n'y ait pas une série de zéros dans les plus forts poids du résultat) on s'arrange pour que le diviseur soit compris entre le dividende et le double de ce dernier pour des nombres en virgule fixe

2322.) Division des nombres inférieurs à 1

La disposition des registres est toujours la même et l'opération est initialisée de la façon suivante:
- Le dividende est initialement chargé dans le registre accumulateur, le diviseur dans le registre B.
- Le registre MQ contient 0

Le processus à suivre est alors identique à celui que nous venons de voir pour des nombres entiers supérieurs à 1, à la seule condition de commencer par une soustraction au lieu d'un décalage pour éviter de faire des erreurs. En effet si on débute par un décalage on risque de perdre le bit de plus fort poids de l'accumulateur.
Traitons quelques exemples pour pouvoir tirer des règles et établir le microprogramme d'exécution de l'opération division

Exemple 1 :

Soit à diviser 0,0111 (7/16) par 0,1110 (14/16)

		2^{-1}	2^{-2}	2^{-3}	2^{-4}		2^{-0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}		2^{-1}	2^{-2}	2^{-3}	2^{-4}	DD
Contenu initial	B	1	1	1	0	Acc	0	0	1	1	1	MQ	0	0	0	0	4
_Soustraction Acc - B							_	1	1	1	0						
							①	1	0	0	1		0	0	0	0	

Décalage	1 0 0 1 0	
Addition Acc+B	+ 1 1 1 0	
Acc[n+1]=0 ⇒ MQ[1]=1	⓪ 0 0 0 0	0 0 0 1
Décalage	0 0 0 0 0	0 0 1 0 2
Soustraction	− 1 1 1 0	
Acc[n+1]=1 ⇒ MQ[1]=0	① 0 0 1 0	0 0 1 0
Décalage	0 0 1 0 0	0 1 0 0 1
Soustraction	− 1 1 1 0	
Acc[n+1]=1 ⇒ MQ[1]=0	1 0 1 1 0	0 1 0 0
Décalage	0 1 1 0 0	1 0 0 0

le résultat est obtenu dans le registre MQ soit 0,1000
Ce qui est bien egal à 1/2

Exemple 2.

Soit à diviser 0,0110 (6/16) par 0,1010 (10/16)

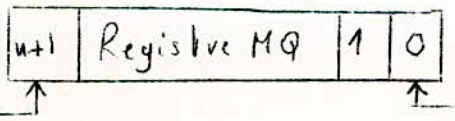
1 0 1 0	0 0 1 1 0	0 0 0 0 4
- Soustraction	− 1 0 1 0	
	① 1 1 0 0	
- Décalage	1 1 0 0 0	0 0 0 0
- Addition	+ 1 0 1 0	
- Acc[n+1]=0 ⇒ MQ[1]=1	⓪ 0 0 1 0	
- Décalage	0 0 1 0 0	0 0 0 1 3
- Soustraction	− 1 0 1 0	
	① 1 0 1 0	

- Décalage	10100	0010	2
- Addition	+1010		
	①1110		
- Décalage	11100	0100	1
- Addition	+1010		
	⓪0110		
- Décalage	01100	1001	0

Le résultat obtenu est 0,1001 c'est à dire 9/16 ou 0,56 On peut éventuellement continuer le processus, ce qui donne, si le registre MQ possède un $(n+2)^{ième}$ bit (bit 0), un résultat de 5 bits, soit dans notre cas 10011. Dans ce cas on peut alors arrondir le résultat trouvé par excés.
L'arrondissement nous donne un résultat qui est:
1001 + 1 = 1010
Comme la virgule est supposée immédiatement à gauche du bit de plus fort poids, on aura donc comme résultat final

0,1010 soit 1/2 + 1/8 = 8/16 + 2/16 = 10/16 = 0,625

Notre division avec arrondissement nous a amenés à trouver un résultat meilleur, plus proche du résultat réel qui est 0,6 Aussi on peut adjoindre à notre registre MQ un bit supplémentaire qui servira à un arrondissement éventuel.



233.) Schéma de principe et microprogramme

2331.) Schéma de principe

Comme on vient de le voir notre opération de division nécessite, des décalages à gauche des registres, des additions et des soustractions selon que le bit signe (n+1) de Acc est à 1 ou à 0, ainsi que des tests. Pour cela présentons nos registres ainsi que tous les signaux positionnant notre opérateur ainsi défini.

Aux signaux précédemment mentionnés, on peut encore ajouter les signaux suivants :

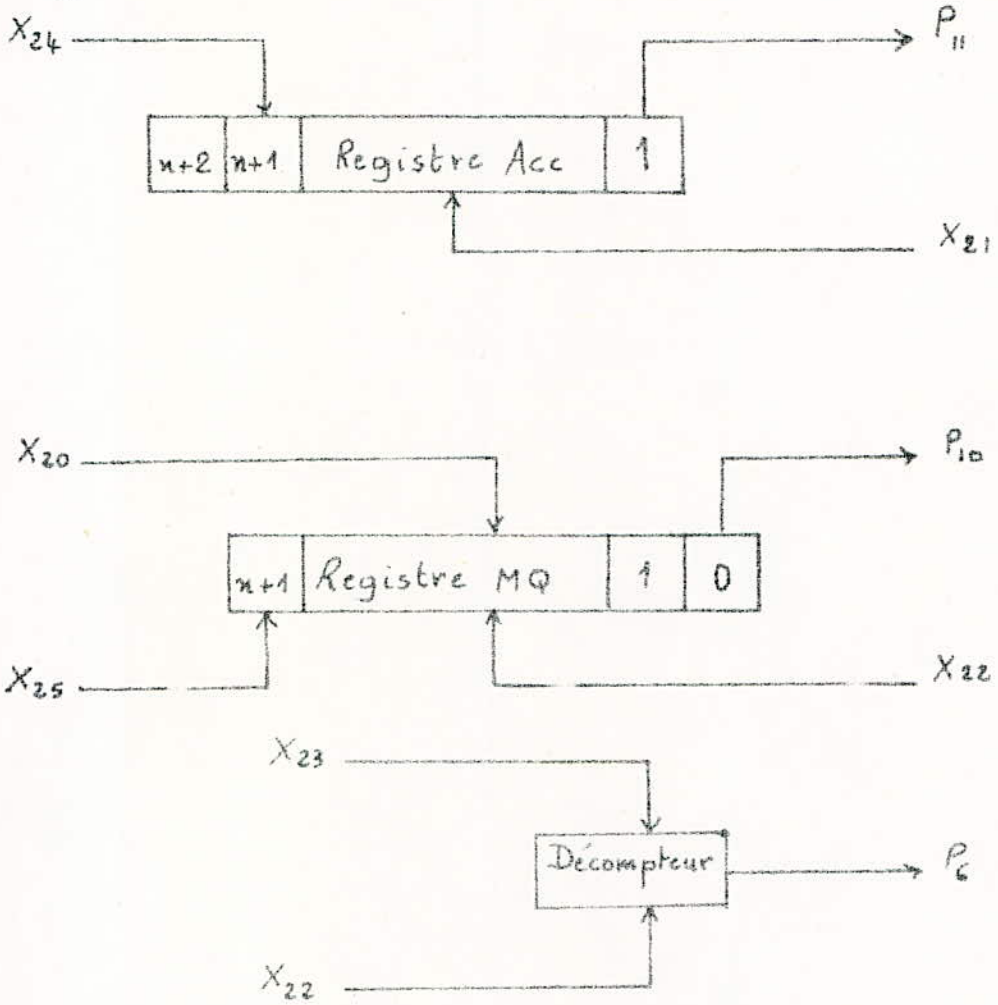


Fig 30 - Présentation des microcommandes -

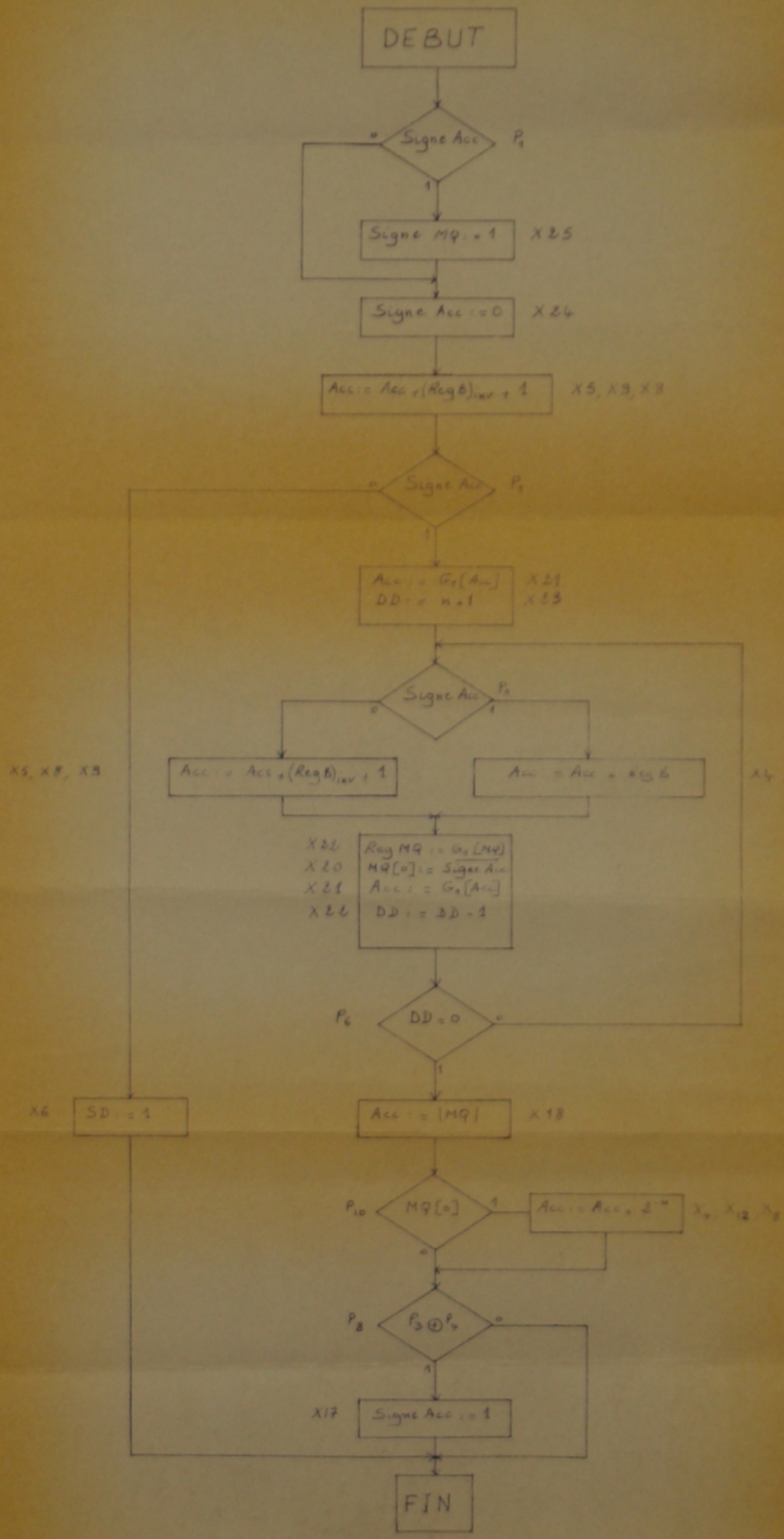


Fig 31 - Microprogramme de la division -

X20 : Reg MQ[0] := $\overline{\text{Signe Acc}}$

X21 : décologe à gauche du registre accumulateur

X22 : décolage à gauche du registre MQ et décrémentation du décompteur de décalages

X23 : DD := n+1

X24 : Signe Acc := 0

X25 : Signe MQ := 1

P_{11} : Etat du bit 1 de l'accumulateur

P_{10} : Etat du bit 0 du registre MQ

2332.) Microprogramme

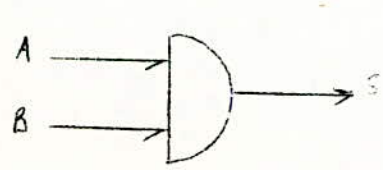
La division de deux nombres sera éxécuté selon le microprogramme suivant (Fig 31)

Lorsque le dividende est supérieur au diviseur, on prévoit un débordement, dans le cas contraire on éffectue l'opération normalement.

CHAPITRE 4 Réalisation d'un opérateur de division

Pour la réalisation de l'opérateur de division, on utilise des circuits simulog qui sont les seuls éléments disponibles dans le laboratoire de notre école.
Pour l'éxécution de l'opération de division, j'ai opté pour l'algorithme sans restauration qui nécessite des décalages ainsi que des additions et soustractions.
les registres "Accumulateur" et "multiplicateur quotient" seront donc des registres à décalage et une impulsion est nécessaire à chaque décalage d'un rang vers la gauche. On peut éventuellement utiliser un décompteur de décalages qui sera chargé en début d'opérations d'une quantité égale au nombre de bits des opérandes.
L'addition des nombres sera éxécutée à l'aide d'étages d'additionneurs, cependant comme on ne dispose pas de circuits soustracteurs, la soustraction sera remplacée par une addition après complementation à deux du nombre à soustraire.
L'opération sera éxécutée bien entendu selon le microprogramme de la division de la figure 31.
A la fin de l'opération, le résultat sera dans le registre MQ et pourra éventuellement être transferé dans le registre Acc.
Pour la réalisation de cet opérateur, on utilise le circuits logiques suivants :

Circuit ET. Il est constitué de 2 entrées et d'une sortie



La sortie se trouve à 1 si les 2 entrées A et B sont toutes deux à 1. la sortie sera à zéro dans tous les autres cas.

Ce circuit réalise le produit logique

Circuit OU. Il est représenté de la façon suivante :

La sortie S est à 1 si l'une des entrées au moins est à 1
Elle est à zéro si les 2 entrées sont à zéro.

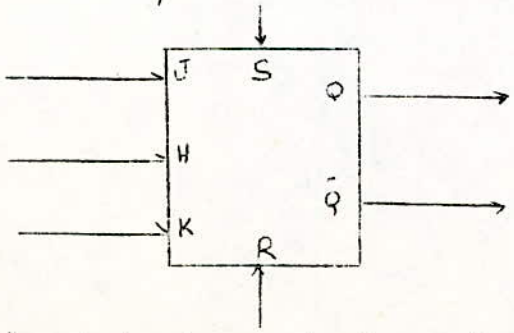
Ce circuit réalise l'addition logique.
Son tableau de vérité est le suivant

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Circuit PAS le symbole graphique de l'inversion logique est

Toute variable A traversant un inverseur logique possède à la sortie de cet inverseur l'état opposé ou complémentaire de son état à l'entrée de l'inverseur. Cet état complémentaire est Ā

Bistable JK. Il possède deux entrées J et K et deux sorties Q et Q̄
En fait il y a une troisième entrée qui est l'horloge. En effet, une propriété importante de cette bascule est qu'elle ne peut changer d'état que lorsque l'entrée horloge passe du niveau 1 au niveau 0.
Cette bascule possède aussi deux autres entrées de forçage R et S respectivement de mise à 0 et 1.



le déclenchement d'une telle bascule se fait en 2 temps Applications des états logiques aux entrées J et K et application du signal d'horloge

- CONCLUSION -

Lors de mon étude, j'ai été amené à utiliser des circuits logiques pour une meilleure compréhension de l'éxecution des opérations arithmétiques. Cependant je dois dire que l'exécution de ces opérations peut se faire aussi à l'aide de circuits beaucoup plus évolués tels que les microprocesseurs 6800 ou 8080 de Motorola qui sont beaucoup plus utilisés dans les grandes industries. L'évolution de la technologie à permi aussi de réduire au maximum les dimensions des composants et d'élaborer ainsi des unités arithmétiques et logiques à l'aide de circuits intégrés.

L'utilisation d'une machine à une adresse est une première approche dans le domaine de la réduction et de l'encombrement parce qu'un seul registre est nécessaire pour stocker le deuxième opérande.

La microprogrammation apporte une aide considérable quant à la synthèse des circuits logiques. Les machines microprogrammées sont beaucoup plus souples et de plus on peut ajouter ou modifier une instruction facilement, alors que pour un système câblé il faut reprendre tout le séquenceur.

Cependant je précise que la représentation des nombres en virgule fixe présente des inconvénients par rapport à la représentation en virgule flottante. En effet pour un même nb de bits d'un registre la valeur d'un nb. en virgule fixe est beaucoup plus petite et le nb. de chiffres significatifs pour les nbs inférieurs à un est beaucoup plus faible. Ainsi la représentation en virgule flottante est beaucoup plus utilisée actuellement dans les calculateurs.

Pour la rapidité d'une machine, on peut éventuellement augmenter la vitesse d'éxecution de ces opérations soit en

conservant l'aspect séquentiel et en diminuant le nombre d'opérations partielles successives, soit en réalisant un opérateur cellulaire permettant une opération quasi parallèle. Cependant le coût d'une telle machine est beaucoup plus élevé parce que le nombre de composants aura augmenté. Dans le monde industriel on a toujours un compromis entre le coût et la vitesse d'exécution.

Pour terminer, j'espère que ce modeste travail pourra éventuellement servir de base ou d'introduction aux futurs ingénieurs pour une bonne compréhension des unités arithmétiques et logiques des calculateurs électroniques.

ANNEXE

Lors de mon étude ainsi que pour la réalisation de l'opérateur de division, j'ai considéré des circuits logiques sans avoir donner une idée de leur réalisation et des circuits élémentaires dont ils sont composés. A cet effet je donne un bref aperçu.

1. Représentation électriques des signaux logiques. Le traitement de l'information dans un calculateur consiste à traiter, transférer, mémoriser des signaux électriques. Dans les calculateurs digitaux, qui nous intéressent ici, l'information traitée qui est de type binaire est représentée par des signaux électriques à deux états qui peuvent être rendus très distincts l'un de l'autre, par exemple un niveau de tension représentant l'état 1 et un autre très différent représentant l'état 0.

1.1 - Logique à niveau. On peut représenter l'information digitale élémentaire sur une ligne électrique en maintenant une tension V_1 pour représenter le 1 logique et une tension V_2 pour représenter le 0 logique.

Sur la figure 34 on remarque que le 1 logique est représenté par un niveau de tension de 6 volts, tandis que le zéro logique est représenté par un niveau de tension de -6 volts.

On dit que l'on a affaire à une logique positive si la tension représentant le 1 logique est supérieure à celle représentant le 0 logique et à une logique négative dans le cas contraire. Très souvent le 0 logique est représenté par la mise à la masse, soit par la tension 0 volt (figure 35)

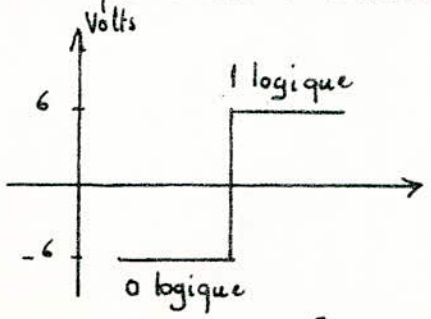


Fig 34

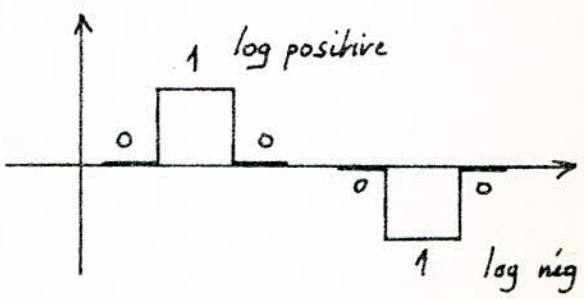


Fig 35

1.2 - Logique impulsionnelle . On parle de logique impulsionnelle lorsque la grandeur électrique qui présente l'information dure un temps très court. Généralement on emploie des impulsions positives (négatives) et on s'oblige à lire les informations à des tops donnés. Si un top coincide avec une impulsion, il s'agit d'un 1 logique, dans le cas contraire d'un 0 logique.

2 - Représentation électrique des circuits logiques.

2.1 - Circuit ET.

Considérons le circuit suivant à deux entrées A et B et une sortie S

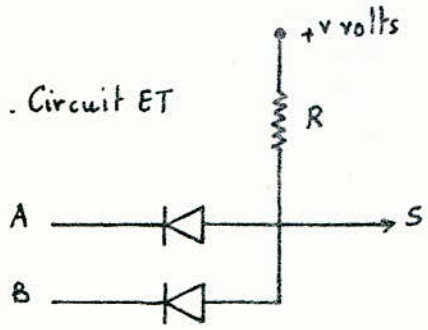


Fig 36 . Circuit ET

Dans ce circuit la sortie S est évidemment à v volts si les 2 entrées sont à v volts. En revanche, si une des entrées est à 0 volt, alors un courant s'établit entre le v volts et cette entrée, la diode correspondante conduit et sa resistance interne est faible devant la résistance de charge R. La différence de Potentiel se retrouve donc presque intégralement aux bornes de la résistance et le niveau de sortie est proche de 0 volt.

2.2 - Circuit OU

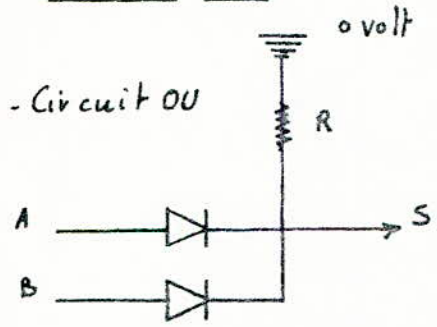


Fig 37 - Circuit OU

Dans ce circuit la sortie S est à 0 volt si les 2 entrées sont à 0 volt. En revanche si une des entrées est à v volts, la sortie sera à v volts.

2.3 - Opérateur complémentation. Considérons le circuit suivant à une entrée A et une sortie S.

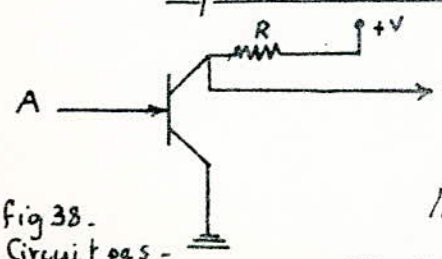


Fig 38. - Circuit pas -

Si l'entrée A est à v volts, le transistor est saturé, sa résistance interne est négligeable devant la charge R qui absorbe donc presque toute la différence de potentiel entre l'alimentation et l'émetteur. La sortie S est donc approximativement à 0 volt. Si l'entrée A est à 0 volt, le transistor est bloqué et la sortie est à la même tension que l'alimentation +v volts.

2.4 - Le bistable JK . Fondamentalement il est constitué par deux circuits NOR montés en opposition et de deux circuits ET montés aussi en opposition . On peut le représenter par le schéma suivant :

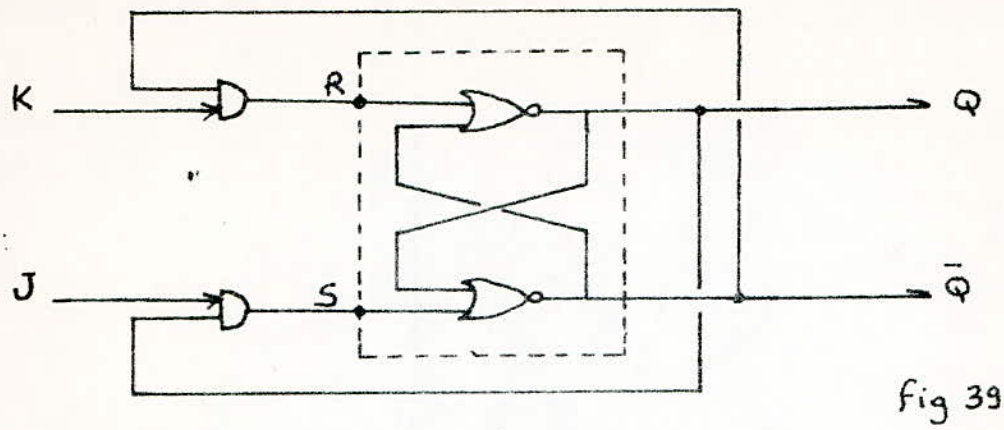


Fig 39 - bistable JK

Les entrées J et K sont impulsionnelles

La particularité de ce bistable est qu'il bascule systématiquement en cas d'impulsions simultanées sur J et K, sous réserve que les impulsions d'entrée soient courtes vis à vis du temps de basculement.

Soit Q_0 la valeur de Q avant une éventuelle impulsion d'entrée, on peut représenter la transition de cette bascule selon que l'on a une impulsion ou pas selon le tableau suivant :

J	K	Q_0	Q	transition
___	___	0	0	inchangé
___	___	1	1	inchangé
⎍	___	0	1	mise à 1
⎍	___	1	1	reste à 1
___	⎍	0	0	reste à 0
___	⎍	1	0	retour à 0
⎍	⎍	0	1	changement à 1
⎍	⎍	1	0	changement à 0

BIBLIOGRAPHIE

P. DEBRAINE : Machines de traitement de l'information
Editions Masson & Cie Tome 1

P. DEBRAINE : Machines de traitement de l'information
Editions Masson & Cie Tome 2

M. AUMIAUX : Logique binaire et ordinateurs
Editions Masson & Cie Tome 1

M. AUMIAUX : Logique binaire et ordinateurs
Editions Masson & Cie Tome 2

J.P. MEINADIER : Structure et fonctionnement des ordinateurs
Editions Larousse

G. BOULAYE : Logique et organes des calculatrices numériques.
Editions Dunod.