P. NASLIN

PRINCIPES
des
CALCULATRICES NUMÉRIQUES AUTOMATIQUES

MONOGRAPHIES DUNOD

PRINCIPES
DES
CALCULATRICES NUMÉRIQUES AUTOMATIQUES

MONOGRAPHIES DUNOD

PRINCIPES
DES
CALCULATRICES NUMÉRIQUES AUTOMATIQUES

PAR

P. NASLIN

Ancien élève de l'École polytechnique
Ingénieur militaire en chef de l'Armement
Professeur à l'École Nationale supérieure de l'Armement,
à l'École supérieure d'Électricité et
l'Institut supérieur des Matériaux et de la Construction mécanique

PARIS

92, RUE BONAPARTE (6e)

1958

AVANT-PROPOS

Les applications du calcul numérique automatique dans les domaines scientifiques, techniques, industriels et commerciaux se développent à un rythme rapide. Afin d'être en mesure de mettre à profit ce nouvel outil, les étudiants et les élèves des grandes écoles techniques, de même que les ingénieurs et techniciens en activité, doivent être informés des principes de l'organisation des calculatrices numériques automatiques, qui n'ont guère changé depuis leur énoncé par Charles BABBAGE, des méthodes d'établissement de leurs programmes, ainsi que de la nature technologique de leurs principaux organes. Au demeurant, ces principes, méthodes et techniques débordent largement le cadre du calcul automatique proprement dit, et s'appliquent plus généralement à la conception des automatismes à séquences, qui constituent la grande majorité des automatismes industriels.

Le présent ouvrage se propose de présenter ces notions fondamentales d'une manière aussi simple que possible, dans l'esprit des remarques figurant à la fin de son introduction. Son organisation est celle des cours que nous professons à l'École Nationale Supérieure de l'Armement et à l'École Supérieure d'Électricité. Nous

TABLE DES MATIÈRES

DEUXIÈME PARTIE

Technologie des calculatrices numériques automatiques.

PRINCIPES DES CALCULATRICES NUMÉRIQUES AUTOMATIQUES

INTRODUCTION

Il n'est sans doute pas inutile, au début de cet exposé, de rappeler qu'il existe deux types bien distincts de machines à calculer : analogiques et numériques.

Les *machines analogiques* constituent en somme des *modèles* des systèmes physiques qu'elles permettent d'étudier. Les variables mathématiques y sont représentées par des grandeurs physiques, telles que tensions électriques ou déplacements angulaires, qui leur sont proportionnelles. Les opérations mathématiques — sommation, multiplication, division, intégration, différentiation, etc... — y sont effectuées par des organes électriques, électroniques ou électromécaniques plus ou moins étroitement spécialisés. Pour résoudre un système d'équations donné, les organes convenables sont interconnectés de manière à constituer un système dont le fonctionnement soit précisément régi par ces équations. Si, comme c'est généralement le cas, les équations à résoudre sont l'expression mathématique du comportement d'un système physique donné, le calculateur analogique, une fois branché pour la résolution de ce problème, constitue donc bien un modèle du système étudié. Soumis à des sollicitations du même type que celles qui seront appliquées au système réel, il réagit de la même manière. De plus, si un certain nombre de paramètres sont à la libre disposition de l'ingénieur, celui-ci peut répéter l'expérience autant de fois qu'il le désire, en donnant à ces paramètres toutes les combinaisons de valeurs possibles, de manière à parvenir au fonctionnement optimum. On voit tout le parti que

l'on peut tirer de cette « *simulation* », qui permet en quelque sorte d'imiter en laboratoire des essais dont la réalisation pratique serait à la fois longue et onéreuse. Toutefois, la précision d'un calcul analogique de quelque importance est limitée à des valeurs de l'ordre du pour-cent par celle des organes de calcul et de mesure constituant le calculateur.

Le mode de travail des *machines à calculer numériques* est totalement différent, du moins en première analyse. Elles apparaissent plutôt comme la version mécanisée d'un *bureau de calcul*, capable d'exécuter automatiquement la totalité d'un calcul numérique. L'examen du fonctionnement d'un bureau de calcul doit donc logiquement nous permettre de dégager les diverses fonctions dont doit être capable une calculatrice numérique automatique.

En premier lieu, un mathématicien spécialiste de l'*analyse numérique* doit se charger de mettre le problème posé sous une forme calculable à l'aide des machines à calculer à clavier en usage dans les bureaux de calcul, c'est-à-dire sous la forme d'une suite d'opérations arithmétiques simples : additions, soustractions, multiplications, divisions, éventuellement racines carrées. Ainsi, les équations différentielles seront remplacées par des équations aux différences finies, les intégrales par des sommes finies, etc...

La précision n'est ici limitée que par le nombre de chiffres significatifs des nombres traités. Ce travail de *préparation* est tout aussi nécessaire si le calcul proprement dit est destiné à être confié à une calculatrice automatique.

Dans le cas du calcul manuel, le plan de calculs ainsi préparé sert à l'établissement de *feuilles de calculs* susceptibles d'être exécutés par des calculateurs consciencieux, mais strictement inintelligents. Le travail de ces derniers consiste essentiellement à lire des nombres dans certaines colonnes de leur feuille de calculs, à les transcrire sur le clavier de leur machine à calculer, à déclencher l'opération indiquée sur la feuille et à recopier le résultat dans une colonne réservée à cet usage. Ces résultats seront en général repris dans un stade ultérieur du calcul, dont les instructions peuvent être ou non portées par la même feuille de calculs.

Ce mode opératoire fait apparaître trois ordres de fonctions fort différentes :

— une fonction de *calcul*, matérialisée par la machine à clavier;

— une fonction d'*emmagasinage* des données numériques et des résultats intermédiaires, dite encore fonction de *mémoire*, matérialisée par l'écriture des nombres dans les cases correspondantes de la feuille de calculs;

— une fonction d'*organisation*, ou de *programme*, matérialisée par la suite, ou séquence, des *instructions* spécifiant la nature des opérations à effectuer sur les nombres mis en mémoire; on remarquera que ces instructions sont, elles aussi, mises en mémoire sur la feuille de calcul.

Nous retrouverons ces mêmes fonctions dans une calculatrice numérique automatique, dont le schéma très général sera donc conforme, dans ses grandes lignes, à celui de la *figure 1*. Tout comme la feuille de calculs, la *mémoire* contiendra à la fois les données numériques du problème, les résultats intermédiaires et les instructions du programme. Elle sera reliée à l'organe de calcul, appelé encore *opérateur*

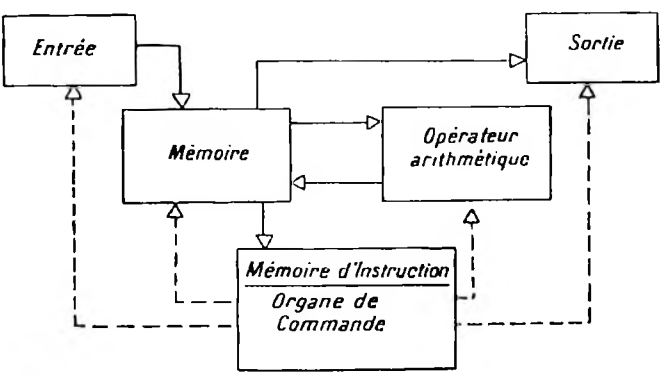


Fig. 1. — Organisation générale d'une calculatrice numérique automatique.

arithmétique, par une liaison bidirectionnelle transportant, dans un sens, les termes des opérations à effectuer et, dans l'autre, les résultats partiels à mettre en mémoire. L'*organe de commande* ou de programme transmettra aux autres organes des signaux de commande sous l'effet des instructions extraites tour à tour de la mémoire, de manière à provoquer l'exécution des instructions du programme dans leur enchaînement correct. Nous pressentons donc d'ores et déjà que l'exécution d'une instruction isolée se fera en deux temps : dans un premier temps, elle sera extraite de la mémoire et emmagasinée provisoirement dans une mémoire spéciale de l'organe de commande, appelée *mémoire d'instruction ;* dans un second temps, elle sera interprétée ou *décodée* de manière à donner naissance aux signaux de commande à appliquer aux organes de calcul et de mémoire, afin d'y provoquer le déroulement des opérations élémentaires désirées. L'enchaînement de ces temps ou *phases* sera gouverné par un organe de synchronisation appelé fréquemment *horloge* ou *générateur de rythme*. Enfin, l'utilisateur disposera d'organes lui permettant, d'une part, d'introduire dans la machine, à partir d'un clavier par exemple, les données et les instructions et, d'autre part, de prendre connaissance des résultats obtenus : ce sont les *organes d'entrée et de sortie*, qui permettent en quelque sorte à la machine de communiquer avec le monde extérieur.

Ces quelques remarques très générales étaient nécessaires afin de situer le problème du calcul numérique automatique et de servir de fil conducteur à l'exposé qui va suivre, en donnant une idée de la nature et de l'articulation des diverses fonctions qui y seront étudiées.

A partir de ce point de départ, nous tenterons de faire participer le lecteur, pas à pas, au processus intellectuel qui a conduit à l'élaboration des principes de fonctionnement des machines à calculer automatiques. Il importe surtout à cet égard de convaincre notre lecteur que ce fonctionnement est simple, logique et naturel, lorsqu'il est dépouillé de ses complications d'origine technologique.

Nous suivrons donc le conseil de DESCARTES et irons du simple au complexe en bâtissant de proche en proche

une machine à calculer numérique automatique simplifiée au moyen d'organes dont la technologie aussi simple que possible ne présente pas de difficulté en soi. Ayant ainsi dégagé les fonctions essentielles du calcul automatique et l'organisation fondamentale d'une machine mathématique numérique, nous passerons en revue les différents types d'organes effectivement utilisés dans les machines modernes

Ainsi, après avoir défini le système de *numération binaire* fondamental en la matière, nous montrerons comment les opérations élémentaires d'addition et de soustraction peuvent être effectuées simplement au moyen de circuits à relais électromagnétiques. Nous dégagerons à cette occasion les lois et le symbolisme de l'*algèbre logique*. Pour pousser plus avant, nous aurons besoin d'organes de *mémoire* susceptibles d'être sélectés à la demande et de communiquer avec notre opérateur arithmétique élémentaire. Les notions de *code* et de *programme* nous conduiront ensuite tout naturellement à l'organisation des transferts entre nos différents organes et feront apparaître la nécessité d'une *mémoire d'instruction*, d'un *compteur ordinal* et d'un *décodeur d'instruction*. Après avoir montré la nécessité des *ruptures de séquence* et donné le moyen de les réaliser, nous pourrons établir le schéma détaillée de notre machine hypothétique et préciser l'*enchaînement des phases*.

Disposant maintenant d'une machine élémentaire en ordre de marche, nous entrerons dans le détail de la confection des programmes, en prenant quelques exemples caractéristiques. Puis nous dirons quelques mots des organes d'entrée et de sortie permettant respectivement d'introduire dans notre machine les données numériques et les instructions du programme, et d'en extraire les résultats.

Nous reviendrons ensuite sur les différentes fonctions ainsi présentées au fur et à mesure de leur besoin et montrerons comment elles sont réalisées pratiquement dans les diverses technologies actuelles : diodes au germanium, tores de ferrite, mémoires acoustiques, tambours magnétiques, etc...

Enfin, nous terminerons en évoquant les applications de l'algèbre de BOOLE à la résolution des problèmes de *logique*.

PREMIÈRE PARTIE

ORGANISATION LOGIQUE D'UNE CALCULATRICE NUMÉRIQUE UNIVERSELLE

Le système de numération binaire.

Nous effectuons habituellement nos calculs numériques dans le système de *numération décimale*. Dans ce système, la représentation physique d'un chiffre, qui peut prendre une valeur quelconque de 0 à 9, nécessite donc l'emploi d'un organe susceptible de pouvoir prendre *dix états* distincts caractérisant chacun l'un des dix chiffres de la numération. Tel est le cas des roues de compteurs utilisées dans les machines à calculer mécaniques à clavier et dans les caisses enregistreuses.

Mais les organes des machines mathématiques numériques modernes ne sont pas de nature mécanique, mais bien plutôt électrique, magnétique ou électronique. Or, les organes mis en œuvre dans ces techniques se caractérisent le plus souvent par l'existence de *deux états* distincts; citons, par exemple, l'état ouvert ou fermé d'un contact, l'état conducteur ou bloqué d'un tube électronique, la présence ou l'absence de courant ou de tension dans une ligne, l'aimantation dans un sens ou dans l'autre d'un matériau magnétique, etc... C'est de cette constatation qu'est née l'idée d'utiliser dans les machines à calculer le système de numération à base 2 ou *système binaire*.

Toutefois, on distingue deux types de machines. Les unes travaillent en numération binaire pure, c'est-à-dire que les nombres qu'elles manipulent sont exprimés dans le système binaire. Bien entendu, l'utilisateur doit ignorer cette circonstance : il est donc nécessaire de traduire automatiquement les données décimales en numération binaire au

moment de leur introduction dans la machine, et d'effectuer la conversion inverse lors de l'extraction des résultats. D'autres machines, au contraire, effectuent leurs opérations dans le système décimal, et ce sont alors les chiffres successifs des nombres qu'elles traitent qui sont exprimés dans le système binaire ou, à tout le moins, au moyen d'organes à caractère binaire. Afin de simplifier les choses, notre machine hypothétique sera du type binaire pur.

Dans le système de numération décimal, les chiffres successifs d'un nombre ont pour poids les puissances successives de 10 disposées en ordre croissant de droite à gauche. Ainsi, le nombre 1957 s'explicite de la manière suivante :

$$1957 = 7 \times (10)^0 + 5 \times (10)^1 + 9 \times (10)^2 + 1 \times (10)^3.$$

Le système de numération binaire ne possède que 2 chiffres, 0 et 1. Dans un nombre exprimé en numération binaire, les poids des chiffres successifs sont les puissances successives de 2 : 1, 2, 4, 8, 16, 32, 64, etc... Ainsi le nombre 1101 s'explicite comme suit :

$$1 \times (2)^0 + 0 \times (2)^1 + 1 \times (2)^2 + 1 \times (2)^3.$$

Il correspond donc au nombre décimal :

$$1 + 4 + 8 = 13.$$

Le tableau page suivante contient la liste des 16 premiers nombres binaires, ainsi que leurs équivalents décimaux.

On remarquera l'alternance des 0 et des 1 dans les colonnes successives. La colonne de droite, correspondant au plus petit poids, est constituée par une alternance de 0 et de 1. La colonne suivante contient alternativement des paires de 0 et des paires de 1. La 3^e colonne en partant de la droite est constituée par des groupes de quatre 0 et de quatre 1. Enfin, la colonne de gauche contient une suite de huit 0 et une suite de huit 1. Le nombre de chiffres de chaque groupe correspond au poids de la colonne. Cette périodicité est représentée schématiquement dans la colonne de droite du tableau ci-dessus, dans laquelle un trait représente le chiffre 1 et un blanc le chiffre 0.

Décimal	Binaire	Représentation schématique
10 1	8 4 2 1	8 4 2 1
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	
1 0	1 0 1 0	
1 1	1 0 1 1	
1 2	1 1 0 0	
1 3	1 1 0 1	
1 4	1 1 1 0	
1 5	1 1 1 1	

D'une façon générale, on peut exprimer au moyen de N chiffres binaires tous les nombres compris entre 0 et $(2^N - 1)$ inclusivement. De même, n chiffres décimaux permettent d'écrire tous les nombres jusqu'à $(10^n - 1)$ inclusivement. Si le plus grand nombre de N chiffres binaires était exactement égal au plus grand nombre de n chiffres décimaux, nous pourrions écrire :

$$2^N - 1 = 10^n - 1.$$

D'où :

$$N \log_{10} 2 = n.$$

Il faut donc environ 3,3 fois plus de chiffres binaires que de chiffres décimaux pour exprimer un même nombre. En particulier, 10 chiffres binaires fournissent une capacité

légèrement supérieure à 3 chiffres décimaux, puisque :

$$2^{10} = 1024.$$

CONVERSION DÉCIMALE-BINAIRE. — Examinons maintenant comment il est possible de convertir un nombre décimal dans la numération binaire. Voyons tout d'abord comment nous nous y prendrions pour extraire successivement les différents chiffres du nombre décimal 1957, en commençant par le chiffre des unités. Divisons-le par 10 :

$$1957 = 10 \times 195 + ⑦.$$

Le reste 7 représente le chiffre des unités. Divisons à nouveau le quotient par 10 :

$$195 = 10 \times 19 + ⑤.$$

Le reste 5 est le chiffre des dizaines. Enfin, le reste et le quotient d'une dernière division par 10 sont égaux respectivement aux chiffres des centaines et des milliers :

$$19 = 10 \times ① + ⑨.$$

De même, une suite de divisions par 2 nous fournira aisément l'équivalent binaire d'un nombre décimal, 59 dans l'exemple ci-dessous :

$$\begin{aligned} 59 &= 2 \times 29 + ① \\ 29 &= 2 \times 14 + ① \\ 14 &= 2 \times 7 + ⓪ \\ 7 &= 2 \times 3 + ① \\ 3 &= 2 \times ① + ① \end{aligned}$$

L'équivalent binaire de 59 s'écrit donc :

$$111011.$$

Vérifions qu'il en est bien ainsi :

$$1 + 2 + 8 + 16 + 32 = 59.$$

L'exécution manuelle de la conversion décimale-binaire est grandement accélérée par l'intermédiaire du système de numération *octal*, c'est-à-dire à base 8. En effet, la base de ce système étant égale à la puissance troisième de 2, l'équivalent binaire d'un nombre exprimé en numération octale s'obtient simplement en écrivant la suite des équivalents binaires de chacun de ses chiffres, exprimés chacun au moyen de 3 chiffres binaires, que l'on écrit aisément de tête. Ainsi, l'équivalent binaire du nombre octal 354 s'écrit immédiatement :

011 101 100.

Inversement, l'équivalent octal d'un nombre binaire s'obtient en séparant ses chiffres en tranches de trois, en partant de la droite, et en écrivant les équivalents de chacune de ces tranches en numération octale, d'ailleurs identiques à leurs équivalents décimaux.

Quant à l'équivalent octal d'un nombre décimal, il s'obtient facilement par divisions successives par 8. Appliquons ces considérations à la conversion du nombre décimal 1957 dans le système binaire. Cherchons tout d'abord son équivalent octal :

$$1957 = 8 \times 244 + ⑤$$
$$244 = 8 \times 30 + ④$$
$$30 = 8 \times ③ + ⑥$$

soit le nombre octal : 3645

qui s'écrit immédiatement dans le système binaire :

011 110 100 101.

Enfin, vérifions que 3645 est bien l'équivalent octal du nombre décimal 1957 :

$$5 + 4 \times 8 + 6 \times (8)^2 + 3 \times (8)^3 = 1957.$$

Nous voyons donc que, si nous devions changer la base de notre système de numération — ce qui est du reste hautement improbable — ce n'est nullement la base 12 qu'il faudrait choisir, mais bien la base 8. La base 12 n'a pas d'avantages particuliers, car il est sans grand intérêt, sauf en calcul mental, de trouver plus ou moins fréquemment des nombres ronds au cours d'un calcul numérique. Au contraire, la base 8 aurait le très grand intérêt de simplifier, quelquefois considérablement, les organes des machines à calculer automatiques, puisque l'équivalent binaire d'un nombre octal s'obtient simplement en mettant bout à bout les équivalents binaires de chacun de ses chiffres.

Addition de deux chiffres binaires, opérations logiques fondamentales.

Dans le système binaire, la table d'addition prend la forme très simple ci-dessous :

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

Dans l'addition de 2 nombres binaires, le jeu des retenues est identique à celui auquel nous sommes habitués dans le système décimal; ainsi :

$$\begin{array}{rcccc} & ① & ① & ① & \\ & 1 & 1 & 0 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 \end{array}$$

Occupons-nous d'abord de l'addition de 2 chiffres binaires, que nous désignerons par a et b; en appelant S le chiffre de plus petit poids de la somme, et R la retenue, la table d'addition écrite ci-dessus nous permet de dresser

le tableau suivant, en envisageant les 4 combinaisons possibles des valeurs 0 ou 1 des chiffres a et b :

a	b	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Nous remarquons immédiatement que $R = 1$ si a *et* b sont tous deux égaux à 1. Nous remarquons également que R est égal au produit habituel de a par b, et nous écrirons donc :

$$R = a \cdot b. \qquad [1]$$

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Notons, sans insister pour le moment, que cette opération de *coïncidence* correspond à l'opération logique d'*intersection.*

Le chiffre de plus petit poids de la somme S est égal à 1 si *a ou b* est égal à 1, mais pas les deux à la fois. C'est le « *ou* exclusif » ou « disjonctif », appelé encore *dilemme :* l'un ou l'autre, mais par les deux. Il correspond à ce que l'on appelle en arithmétique « somme modulo 2 », c'est-à-dire l'excès de la somme par rapport au multiple de 2 le plus

élevé qu'elle contient (ici, c'est simplement l'excès par rapport à 2), et on l'écrit :

$$S = a \oplus b. \qquad [2]$$

Mais le dilemme n'est pas une opération logique élémentaire. On l'exprime aisément au moyen de l'intersection (*et*) définie plus haut et de deux autres opérations élémentaires : la négation ou complément, et la réunion (*ou* inclusif).

La *négation* ou *complément*, que l'on symbolise par une barre supérieure, se définit très simplement par le tableau suivant (on lit « *a* barre ») :

a	$\bar{a}$
0	1
1	0

Quant à la réunion, c'est le « *ou* inclusif », c'est-à-dire : l'un ou l'autre ou les deux; c'est donc le « *vel* » latin, d'où son symbole $\vee$:

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

C'est une opération d'anticoïncidence, $(a \vee b)$ n'étant nul que si *a et b* sont simultanément nuls. En d'autres termes, la négation appliquée à $(a \vee b)$ est équivalente à l'intersection (*et*) des compléments de *a et b* :

$$\overline{a \vee b} = \bar{a} \cdot \bar{b}.$$

De même, si nous nous reportons au tableau de l'intersection, nous voyons que $\overline{a \cdot b}$ est nul si *a ou b*, ou les deux, sont nuls. En d'autres termes, le complément de $a \cdot b$ est équivalent à la réunion (*ou*) des compléments de *a et b* :

$$\overline{a \cdot b} = \bar{a} \vee \bar{b}.$$

Revenons maintenant à l'expression de S. Nous pouvons l'exprimer de deux manières différentes au moyen des opérations définies ci-dessus. Tout d'abord, nous voyons que $S = 1$, soit si $a = 1$ *et* $b = 0$, soit si $a = 0$ *et* $b = 1$ (*a* seul *ou b* seul), ce qui s'écrit :

$$S = a \cdot \bar{b} \vee \bar{a} \cdot b. \qquad [3]$$

La simultanéité des deux termes du second membre est évidemment exclue. Mais nous pouvons dire encore : $S = 1$ si *a ou b* est égal à 1, mais pas *a et b*, ce qui s'écrit :

$$S = (a \vee b) \cdot (\overline{a \cdot b}). \qquad [4]$$

Il est facile de démontrer l'équivalence de ces deux expressions. Remplaçons, dans la seconde, la négation $\overline{a \cdot b}$ par sa valeur développée écrite plus haut :

$$S = (a \vee b)(\bar{a} \vee \bar{b}). \qquad [5]$$

Les opérateurs *et* et *ou* étant commutatifs et associatifs, on peut développer ce produit comme un produit de sommes ordinaires :

$$S = a\bar{a} \vee b\bar{b} \vee a\bar{b} \vee \bar{a}b.$$

Or, il est évident que :

$$a \cdot \bar{a} = 0 \qquad et \quad b \cdot \bar{b} = 0$$

puisqu'une variable binaire ne peut être simultanément

égale à 0 *et* à 1. Il est non moins évident que, d'une manière générale :

$$0 \vee x = x \text{ (1)}.$$

On peut donc supprimer les deux termes $a \cdot \bar{a}$ et $b \cdot \bar{b}$ de l'expression ci-dessus et le résultat annoncé est ainsi démontré.

A titre d'exercice, faisons subir à l'expression [3] la transformation inverse, en la développant cette fois sous la forme d'un produit de réunions :

$$a\bar{b} \vee \bar{a}b = (a \vee \bar{a})(b \vee \bar{b})(a \vee b)(\bar{a} \vee \bar{b}).$$

Or, il est évident que :

$$a \vee \bar{a} = 1 \qquad et \quad b \vee \bar{b} = 1.$$

puisqu'une variable binaire est nécessairement égale, soit à 0, soit à 1. D'autre part, il est non moins évident que, d'une manière générale :

$$1 \cdot x = x.$$

Nous pouvons donc supprimer, dans l'expression ci-dessus, les deux parenthèses $(a \vee \bar{a})$ et $(b \vee \bar{b})$ et nous retrouvons donc bien l'expression [5], équivalente à [4].

Circuits logiques à relais électromagnétiques.

Montrons maintenant comment des chaînes de contacts commandés par relais électromagnétiques permettent de réaliser très facilement les opérations logiques envisagées ci-dessus. Pour un même relais, nous désignerons par une lettre minuscule l'enroulement d'excitation et par la majuscule correspondante les contacts qu'il commande. Considérons d'abord (*fig. 2 a*) un contact « travail », c'est-à-dire ouvert

---

(1) On voit que le signe $\vee$ est assez comparable au signe $+$, à cette différence près que : $1 \vee 1 = 1$.

lorsque le relais n'est pas excité et fermé lorsqu'il est excité. Si un tel contact alimente l'enroulement b d'un second relais, ce dernier sera excité ou non en même temps que le premier. C'est l'opération *d'identité* :

$$b = a.$$

Au contraire, considérons un contact « repos » (*fig. 2 b*), c'est-à-dire fermé quand le relais n'est pas excité, et inversement. Si ce contact alimente un enroulement c, le relais c sera excité lorsque le relais a ne l'est pas, et inversement. C'est donc l'opération de *complément ou négation* :

$$c = \bar{a}.$$

C'est pourquoi, dans nos schémas, nous désignerons les

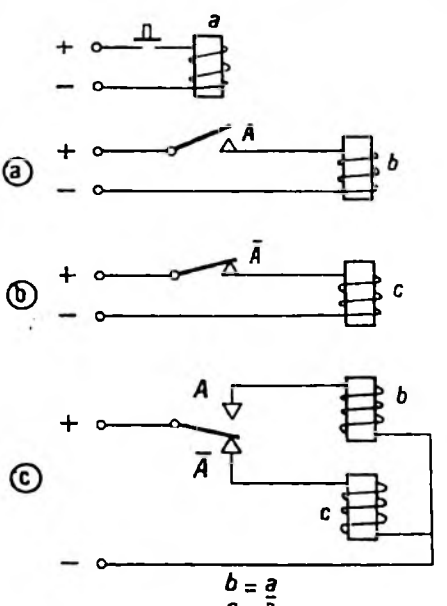


FIG. 2. — Circuits d'identité et de complément à relais électromagnétiques.

contacts « repos » par des majuscules surmontées d'une barre.

Un même relais peut comporter un grand nombre de contacts « travail » et « repos », et c'est là l'un des avantages des circuits à relais par rapport à leurs équivalents électroniques. En particulier, si l'on a besoin, dans un même circuit, d'une certaine variable et de son complément, on utilisera un *inverseur* tel que celui de la *figure 2 c*.

L'opération d'*intersection* (*et*) se réalise aisément en montant en série deux contacts « travail » A et B (*fig. 3*) : le

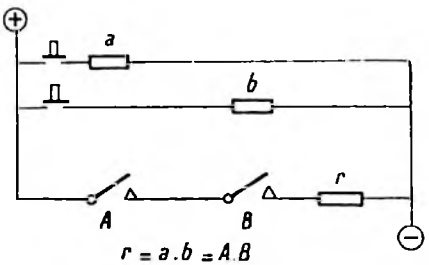


FIG. 3. — Circuit d'intersection (*et*).

relais r n'est excité que si les deux contacts sont simultanément fermés, c'est-à-dire si les deux relais a et b sont simultanément excités :

$$r = a \cdot b = A \cdot B.$$

Un tel circuit fournit donc la retenue de l'addition de 2 chiffres binaires.

De même, l'opération de *réunion* (*ou*) résulte du montage en parallèle de deux contacts « travail » A et B (*fig. 4*), puisqu'il suffit que l'un des deux contacts soit fermé, donc l'un des deux relais excités, pour que le relais c le soit également.

Examinons maintenant l'opération de *dilemme* (*ou* exclusif),

qui correspond à la formation du chiffre de plus petit poids de la somme de 2 chiffres binaires (somme modulo 2).

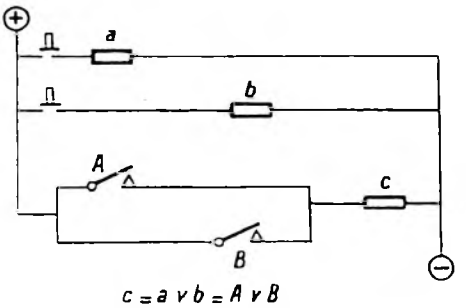


FIG. 4. — Circuit de réunion (*ou* inclusif).

Prenons d'abord la forme [3] (*fig. 5 a*). Le terme $a \cdot \bar{b}$ s'obtient par le montage en série d'un contact « travail » A et d'un contact « repos » B; de même, le terme $\bar{a} \cdot b$ résulte du mon-

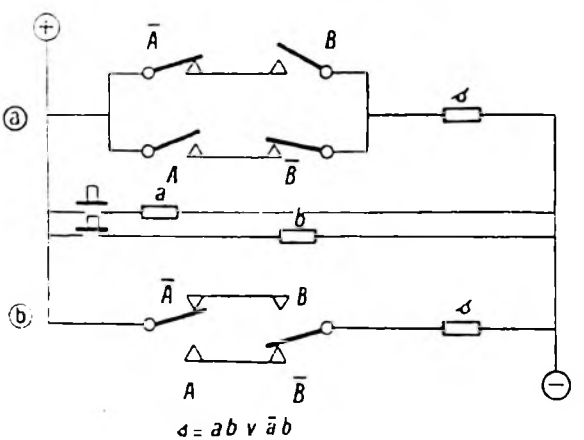


FIG. 5. — Circuit de dilemme (*ou* exclusif).

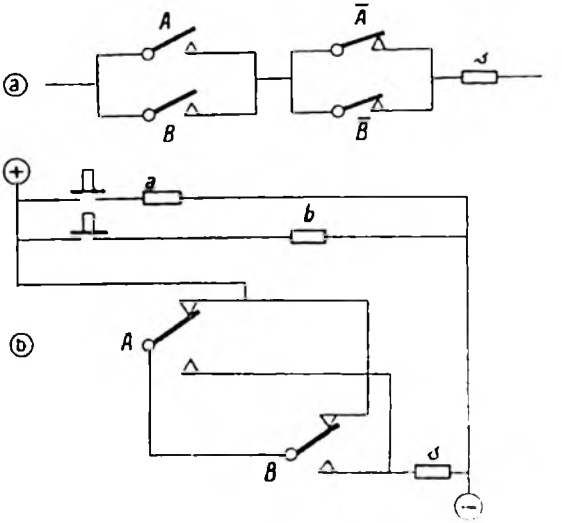


FIG. 6. — Autre circuit de dilemme
(sans avantage par rapport à celui de la *fig. 5*).

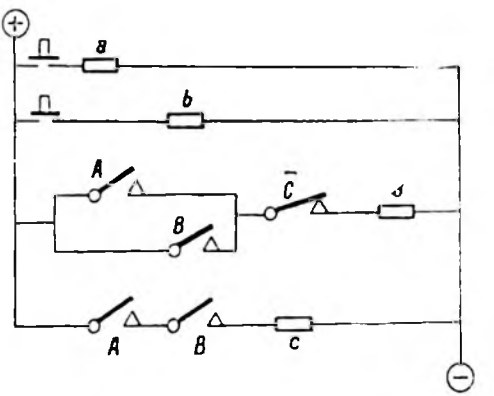


FIG. 7. — Autre circuit de dilemme (moins avantageux que celui de la *fig. 5*).

tage en série d'un contact « repos » A et d'un contact « travail » B. L'expression complète de S s'obtient enfin par la mise en parallèle de ces deux circuits. Ce circuit se simplifie par l'emploi de deux inverseurs (*fig. 5 b*). Il est bien évident que le circuit est fermé, et le relais s excité, si l'un seul des deux relais est excité; il est à nouveau coupé si les deux relais sont excités simultanément.

Utilisons maintenant l'expression [5], qui correspond au montage de la *figure 6 a*. Ici encore, on peut avantageusement utiliser deux inverseurs (*fig. 6 b*). Mais ce circuit ne présente aucun avantage par rapport à celui de la *figure 5 b*.

Enfin, cherchons à traduire la forme [4]. Pour ce faire, nous avons besoin d'un relais supplémentaire c muni d'un contact repos et excité par $a \cdot b$, afin d'obtenir le complément de cette quantité (*fig. 7*). Ce circuit est donc manifestement désavantageux par rapport aux deux précédents.

Additionneurs binaires.

Nous venons de voir comment constituer un circuit à relais capable d'effectuer l'addition de 2 chiffres binaires. Mais un tel circuit ne constitue pas un additionneur binaire. En effet, pour chaque rang binaire, la retenue provenant de l'étage précédent peut venir s'ajouter à la somme des deux chiffres de ce rang. C'est donc en réalité 3 chiffres binaires que l'on doit additionner.

ADDITIONNEUR A DEUX DEMI-ADDITIONNEURS. — Une première solution consiste à procéder en deux temps (*fig. 8 a*). Pour le rang binaire n, on commence par effectuer l'addition des 2 chiffres a_n et b_n de ce rang; on obtient ainsi une retenue $a_n \cdot b_n$ et une somme modulo 2 :

$$c_n = a_n \oplus b_n = a_n \overline{b_n} \vee \overline{a_n}\, b_n.$$

Puis, en un deuxième temps, on procède à l'addition de cette somme c_n avec la retenue r_{n+1} en provenance du rang

de poids immédiatement inférieur (1). On obtient ainsi la somme définitive S_n et une seconde retenue $(a_n \oplus b_n)\ r_{n+1}$. On remarque que les deux retenues s'excluent mutuellement, car on ne peut pas avoir simultanément $a_n \cdot b_n$ (a_n *et* b_n)

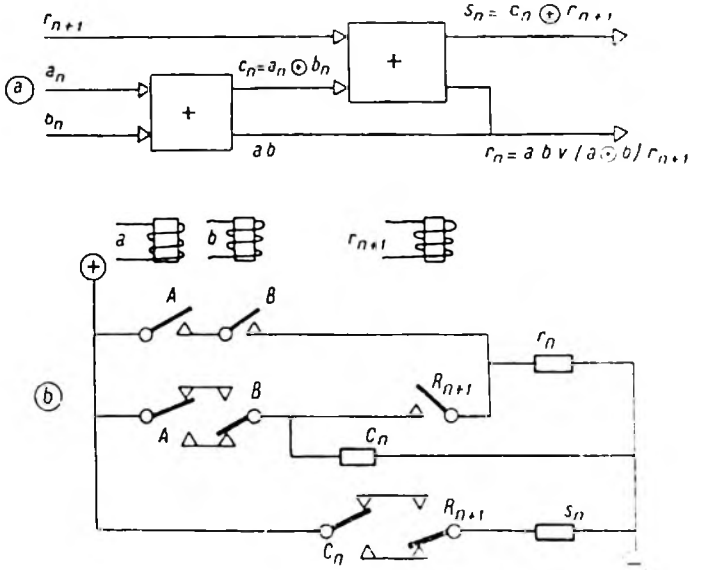


FIG. 8. — Additionneur binaire à deux demi-additionneurs : (*a*) schéma logique; (*b*) réalisation à relais électromagnétiques.

---

(1) On voit que nous numérotons les chiffres successifs d'un même nombre de gauche à droite, c'est-à-dire dans le sens dans lequel nous avons l'habitude d'écrire les nombres. Avec cette convention, l'ordre des indices est donc l'ordre inverse des poids. La retenue r_{n+1} est donc bien celle provenant du rang de poids immédiatement inférieur au rang n considéré. Cela n'est peut-être pas très logique, mais il fallait choisir entre cet illogisme conforme à nos habitudes ou la logique qui eût voulu que nous numérotions les chiffres dans l'ordre inverse de leur écriture usuelle, ce qui n'eût guère été plus commode.

et ($a_n \oplus b_n$) (a_n *ou* b_n mais pas les deux à la fois). Il est donc possible de les mélanger directement sans précaution.

La nature de ce schéma justifie le nom de *demi-additionneur* donné au circuit d'addition de 2 chiffres binaires, puisqu'il faut utiliser deux tels circuits pour constituer un additionneur complet.

La *figure 8 b* représente le branchement des relais pour effectuer cette opération.

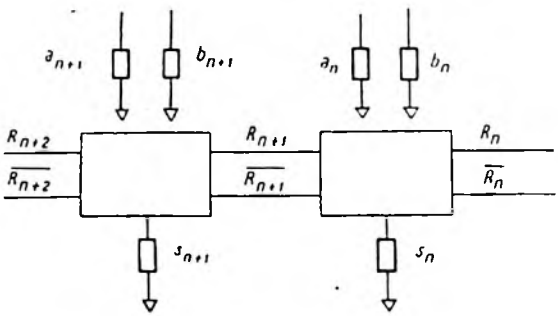
Ce schéma très simple présente toutefois un grave inconvénient. Dans les circuits à relais, on s'efforce, dans toute la mesure du possible, de positionner les contacts avant d'y faire passer du courant. Or, si cela est possible pour les relais a_n et b_n, il n'est pas de même pour les relais de retenue r_{n+1}. La retenue de rang n ne peut être correcte qu'après que l'étage précédent a pris son état définitif. Elle peut, à son tour, modifier la retenue dans l'étage suivant, et ainsi de suite. Dans le cas le plus défavorable, une retenue provenant de l'addition des 2 chiffres de plus petit poids peut se propager jusqu'au rang de plus fort poids :

$$\begin{array}{r} \textcircled{1}\,\textcircled{1}\,\textcircled{1}\,\textcircled{1}\,\textcircled{1}\,\textcircled{1} \\ 1\;1\;1\;1\;1\;1 \\ +\;0\;0\;0\;0\;0\;1 \\ \hline 1\;0\;0\;0\;0\;0\;0 \end{array}$$

Or, le changement d'état d'un relais n'est pas un processus instantané. Sauf disposition particulière, il faudra donc réserver, pour l'addition de deux nombres binaires de n chiffres, un intervalle de temps correspondant à n fonctionnements de relais en cascade.

ADDITIONNEUR A TROIS ENTRÉES. — Nous allons voir qu'il est possible d'éviter cet inconvénient au moyen d'un circuit dans lequel les retenues ne sont pas représentées par l'état de relais, mais par la fermeture ou l'ouverture de circuits de retenue reliant les étages entre eux. Le schéma général d'un tel additionneur est conforme à la *figure 9*. Deux étages successifs sont reliés par deux lignes R et $\bar{R}$, dont l'une ou

l'autre est sous tension selon qu'il y a une retenue à transmettre ou non.



IG. 9. — Principe d'un additionneur binaire à relais à retenues simultanées.

Commençons par établir le tableau des diverses combinaisons possibles des 3 chiffres a_n, b_n et r_{n+1} en portant notre attention sur la présence ou l'absence de r_{n+1} :

a_n	b_n	r_{n+1}	s_n	r_n
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Le simple examen de ce tableau nous permet d'écrire, immédiatement :

$$s_n = r_{n+1}\,(a_n\,b_n \vee \overline{a_n}\,\overline{b_n}) \vee \overline{r_{n+1}}\,(a_n\,\overline{b_n} \vee \overline{a_n}\,b_n) \qquad [6]$$

$$r_n = a_n\, b_n \vee r_{n+1}\,(a_n\, \overline{b_n} \vee \overline{a_n}\, b_n) \qquad [7]$$

$$r_n = \overline{a_n}\;\overline{b_n} \vee \overline{r_{n+1}}\,(a_n\, \overline{b_n} \vee \overline{a_n}\, b_n). \qquad [8]$$

Dans l'expression [7], on remarquera que $a_n\, b_n$ traduit à la fois la quatrième et la huitième ligne du tableau en ce qui concerne la valeur de r_n. De même, le terme $\overline{a_n}\;\overline{b_n}$ de l'expression [8] contient à la fois la valeur de r_n à la première et à la cinquième ligne.

Afin d'en être bien convaincus, écrivons par exemple l'expression développée de r_n, en lisant le tableau ligne par ligne :

$$r_n = a_n\, b_n\, \overline{r_{n+1}} \vee \overline{a_n}\, b_n\, r_{n+1} \vee a_n\, \overline{b_n}\, r_{n+1} \vee a_n\, b_n\, r_{n+1}.$$

Regroupons les termes :

$$r_n = a_n\, b_n\,(r_{n+1} \vee \overline{r_{n+1}}) \vee r_{n+1}\,(a_n\, \overline{b_n} \vee \overline{a_n}\, b_n).$$

Or, il est évident que :

$$r_{n+1} \vee \overline{r_{n+1}} = 1.$$

ce qui nous donne bien l'expression [7].

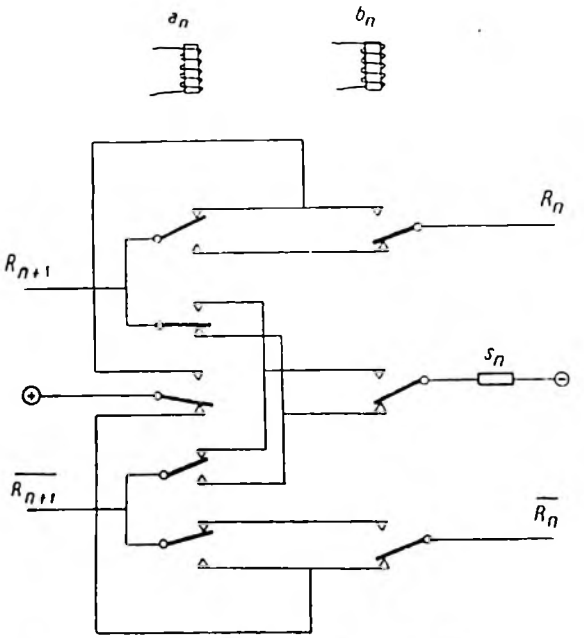
Enfin, il faut encore préciser que, pour le rang de plus faible poids, la retenue incidente est nulle. Si donc nous travaillons sur des nombres de N chiffres binaires, nous écrivons :

$$r_{N+1} = 0, \qquad \overline{r_{N+1}} = 1. \qquad [9]$$

La *figure 10* représente un étage d'additionneur binaire à relais. Ce schéma n'est pas autre chose que la traduction des équations [6], [7] et [8], en profitant toutefois du fait que s_n peut être formé au moyen d'un inverseur b_n unique.

Répétons que, dans ce circuit, la valeur 1 ou 0 des variables r_{n+1} et $\overline{r_{n+1}}$ ne correspond pas à l'état des contacts de relais, mais à la présence ou à l'absence de courant sur les lignes

correspondantes R_{n+1} et $\overline{R_{n+1}}$ apportant la retenue en provenance de l'étage précédent. C'est l'état des contacts de ce dernier, ainsi que la nature de la retenue qu'il reçoit lui-même de l'étage qui le précède, qui détermine laquelle d'entre ces deux lignes est sous tension : R_{n+1} s'il y a une nouvelle retenue, $\overline{R_{n+1}}$ dans le cas contraire. Dans un tel circuit, on voit donc que, dès que les relais a et b des différents étages ont pris leurs positions, l'état de toutes les retenues



$$\begin{cases} s_n = R_{n+1}\,(a_n b_n \vee \overline{a_n}\,\overline{b_n}) \vee \overline{R_{n+1}}\,(a_n \overline{b_n} \vee \overline{a_n} b_n) \\ R_n = a_n b_n \vee R_{n+1}\,(a_n \overline{b_n} \vee \overline{a_n} b_n) \\ \overline{R_n} = \overline{a_n}\,\overline{b_n} \vee \overline{R_{n+1}}\,(a_n \overline{b_n} \vee \overline{a_n} b_n) \end{cases}$$

FIG. 10. — Étage d'addition binaire à relais.

se trouve immédiatement déterminé, ainsi par conséquent que celui des chiffres de la somme.

Les circuits représentant les parenthèses en facteur avec $\overline{r_{n+1}}$ ou r_{n+1} sont donc alimentés par les lignes R_{n+1} ou $\overline{R_{n+1}}$ respectivement, tandis que les circuits d'intersection correspondant aux termes $a_n\, b_n$ et $a_n\, b_n$ sont connectés directement à la source d'alimentation, supposée ici positive.

Faisons encore une remarque à propos du dessin de la *figure 10*. Nous n'y avons pas marqué par des points noirs les jonctions de plusieurs fils en un même point. Nous ne le ferons pas non plus dans la suite, car il est très facile d'oublier un point ou deux dans la reproduction d'un dessin, qui devient alors inintelligible. Il ne peut du reste y avoir ambiguïté que dans les cas où deux fils se croisent. Ainsi, la *figure 11* représente en (*a*) deux fils qui se croisent

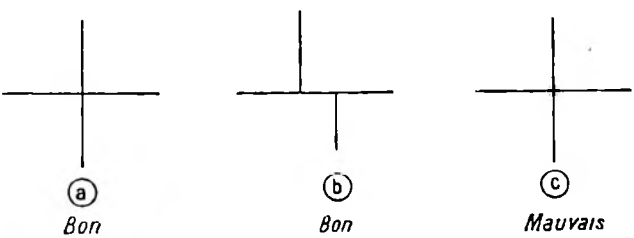


FIG. 11. — Convention de schématique : deux fils qui se croisent ne se. jamais en contact électrique (*a*); une jonction quadruple est représentée comme en (*b*); la représentation (*c*) n'est jamais utilisée dans ce livre.

sans contact, et en (*c*) avec jonction représentée selon le symbolisme courant, et très malheureusement normalisé. Pour notre part, nous nous interdisons de faire usage de la représentation (*c*), et représentons toujours une jonction quadruple sous la forme (*b*). L'usage des points devient alors totalement inutile. Il convient de souhaiter que la norme incriminée plus haut soit rapidement amendée.

Soustraction.

La question de la soustraction se ramène en réalité à celle de la représentation des nombres négatifs.

A cette fin, on peut adopter deux conventions différentes :

— la première, dite *représentation algébrique pure*, consiste simplement à juxtaposer la valeur absolue et le signe, à la manière habituelle, le signe étant, comme les chiffres, une variable binaire; cette convention est très commode pour les opérations de multiplication et de division, puisqu'il suffit d'effectuer le produit ou le quotient des valeurs absolues et de combiner les signes à la manière usuelle; elle est peu commode, par contre, pour la somme algébrique ou la soustraction, car il faut commencer par comparer les deux valeurs absolues afin de savoir lequel des deux nombres doit être retranché de l'autre.

— la seconde convention, dite *représentation algébrique mixte*, utilise le code des compléments, qui nous conduit tout d'abord à préciser la position de la virgule.

Position de la virgule. — Elle peut être choisie arbitrairement. Néanmoins, il est commode, pour diverses raisons qui apparaîtront dans la suite, de l'imaginer située à droite du rang binaire de poids le plus élevé, c'est-à-dire de traiter des nombres toujours inférieurs à l'unité en valeur absolue. Ceci implique évidemment le choix d'une échelle convenable pour les variables du problème à traiter. Nous verrons toutefois ultérieurement que cet inconvénient peut être levé par l'emploi d'une virgule dite « flottante ».

Limitons-nous, pour fixer les idées, à 5 chiffres binaires. Nous rencontrerons par exemple le nombre :

$$0,1\,1\,0\,1.$$

Les poids des chiffres situés à droite de la virgule sont les puissances négatives successives de 2 :

$$\frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{8} \quad \frac{1}{16}$$

soit encore :

$$\frac{8}{16} \quad \frac{4}{16} \quad \frac{2}{16} \quad \frac{1}{16}.$$

Le nombre écrit plus haut est donc l'équivalent binaire de 13/16.

CODE DES COMPLÉMENTS : SOUSTRACTION. — Considérons toujours le nombre 0,1101 et imaginons que nous lui ajoutions l'équivalent binaire de 2 :

$$\begin{array}{r} 0,1101 \\ +\ 10,0000 \\ \hline \textcircled{1}\,0,1101 \end{array}$$

Notre machine ignorant le second rang binaire à gauche de la virgule, de poids 2, la représentation de notre nombre demeure donc inchangée après cette opération.

Envisageons alors l'opération $(a - b)$, a et b étant tous deux inférieurs à l'unité. Cette différence a la même représentation binaire que l'expression :

$$2 + (a - b) \qquad \text{soit encore :} \quad a + (2 - b).$$

Nous voyons donc que, pour soustraire le nombre b du nombre a, il suffit d'ajouter à ce dernier le complément à 2 de b.

La question qui se pose maintenant est donc de savoir former commodément ce complément. Cherchons donc le complément à 2 de 0,1101, équivalent binaire de 13/16 :

$$\begin{array}{r} 10,0000 \\ -\ 0,1101 \\ \hline 1,0011 \end{array}$$

Ce dernier nombre est l'équivalent binaire de 19/16, qui est bien le complément à 2 de 13/16. Le chiffre à gauche de la virgule caractérise le signe : 0 pour un nombre positif, 1 pour le complément à 2 d'un nombre positif, qui nous sert de représentation du même nombre changé de signe.

Au lieu du complément à 2, cherchons le complément par rapport au plus grand nombre que nous puissions représenter avec nos 5 chiffres binaires, c'est-à-dire 1,1111. Ce dernier nombre est l'équivalent binaire de (2 — 1/16), soit en binaire :

$$\begin{array}{r} 10,0000 \\ -\ 0,0001 \\ \hline 1,1111 \end{array}$$

Cherchons donc le complément à 1,1111 de 0,1101 :

$$\begin{array}{r} 1,1111 \\ -0,1101 \\ \hline 1,0010 \end{array}$$

Ce complément est appelé « *complément restreint* », par opposition au « *complément vrai* » ou complément à 2. Sa formation est particulièrement aisée, puisqu'elle consiste simplement à changer les 0 en 1 et inversement.

D'après ce qui vient d'être dit, le complément vrai se forme donc en ajoutant au complément restreint une unité de plus petit poids. Ainsi :

$$\begin{array}{rr} \text{C.R. } 0,1101 : & 1,0010 \\ & +\ 0,0001 \\ \hline \text{C.V. } 0,1101 : & 1,0011 \end{array}$$

On peut encore dire, plus simplement, que le complément vrai s'obtient en prenant les compléments des chiffres successifs, en partant des forts poids, c'est-à-dire de la gauche en écriture normale, jusqu'à ce que l'on arrive au chiffre « 1 » de plus petit poids, que l'on conserve inchangé.

Appliquons ces considérations à la formation de la différence (13/16 — 7/16), dans le système binaire. L'équivalent binaire de 7/16 est 0,0111. Formons son complément vrai par l'intermédiaire du complément restreint :

$$\begin{array}{rr} \text{C.R. } 0,0111 : & 1,1000 \\ & +\ 0,0001 \\ \hline \text{C.V. } 0,0111 : & 1,1001 \end{array}$$

Ajoutons ce dernier nombre à l'équivalent binaire de 13/16 :

$$\begin{array}{r} 0,1\,1\,0\,1 \\ +\,1,1\,0\,0\,1 \\ \hline \textcircled{1}\,0,0\,1\,1\,0 \end{array}$$

Comme nous ignorons le second rang binaire à gauche de la virgule, notre résultat est 0,0110; c'est bien l'équivalent binaire de la différence 6/16.

Effectuons maintenant la différence inverse (7/16 — 13/16). Pour cela, ajoutons à l'équivalent binaire de 7/16 le complément de 13/16, que nous avons formé plus haut :

$$\begin{array}{r} 0,0\,1\,1\,1 \\ +\,1,0\,0\,1\,1 \\ \hline 1,1\,0\,1\,0 \end{array}$$

Le « 1 » à gauche de la virgule caractérise un nombre négatif, dont nous trouverons la valeur absolue en cherchant son complément vrai :

$$\begin{array}{lr} \text{C.R. } 1{,}1010 : & 0,0\,1\,0\,1 \\ & +\,0,0\,0\,0\,1 \\ \text{C.V. } 1{,}1010 : & 0,0\,1\,1\,0 \end{array}$$

Nous trouvons bien la valeur absolue attendue 6/16. La *figure 12* précise bien la correspondance entre les

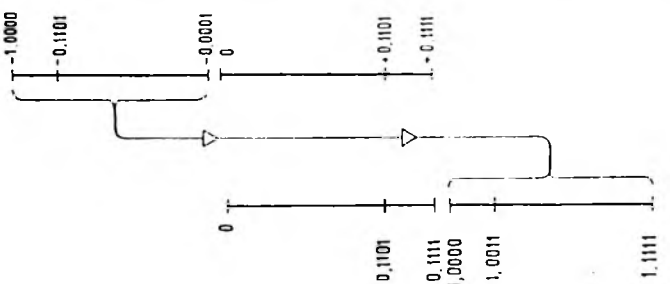


FIG. 12. — Correspondance entre les nombres négatifs (— 1 inclus) et leurs compléments à 2.

nombres négatifs et leur représentation par le complément à 2 de leur valeur absolue. Nos 5 chiffres binaires nous permettent de représenter tous les nombres de 4 chiffres après la virgule compris entre 0 et 1,1111 (2 — 1/16). Nous avons divisé cette gamme en deux tranches :

— la tranche comprise entre 0 et 0,1111 (1 — 1/16), que nous affectons aux nombres positifs inférieurs à l'unité;

— la tranche comprise entre 1 et 1,1111, composée des compléments à 2 des valeurs absolues des nombres négatifs compris entre — 0,0001 et — 1.

Dans ce système, — 1 se trouve donc représenté par 1,0000, qui est évidemment son complément à 2. La signification de « 1 » à gauche de la virgule, qui — nous l'avons signalé — caractérise les nombres négatifs, se trouve donc conservée. Mais le nombre + 1 ne peut être représenté, le plus grand nombre positif représentable étant 0,1111 (15/16).

Si l'on voulait pouvoir représenter le nombre + 1, il faudrait adopter la correspondance schématisée sur la *figure 13*, dans laquelle le nombre + 1 s'écrit 1,0000. Mais alors c'est le nombre — 1 qui n'est plus représentable, le plus petit nombre négatif étant — 0,1111, représenté par 1,0001. Ce système a toutefois l'inconvénient d'introduire une exception dans l'interprétation du chiffre à gauche de la virgule.

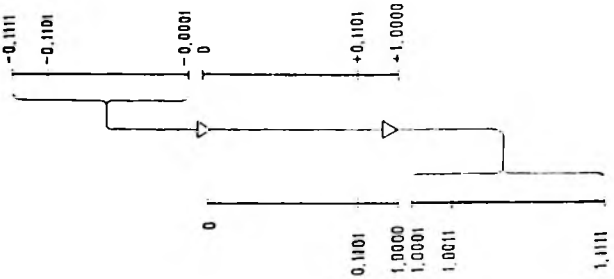


FIG. 13. — Autre correspondance possible entre les nombres négatifs (— 1 exclus) et leurs compléments à 2.

Additionneur-soustracteur binaire à relais (*fig.* 14).

Ce qui précède nous permet de former aisément la différence (a — b) de deux nombres de chiffres successifs a_n et b_n. Cette différence s'obtient, comme nous venons de le voir, en ajoutant à a le complément vrai de b. Mais il revient au même — et il est beaucoup plus simple techniquement — d'ajouter à a le complément restreint de b, et d'ajouter au résultat une unité de plus petit poids. Ce dernier résultat s'obtient automatiquement en donnant à la retenue arrivant dans l'étage de plus faible poids la valeur 1, au lieu de la valeur 0 dans le cas d'une addition (voir équation [9]). Si donc nous traitons des nombres de N chiffres binaires, nous aurons pour la soustraction :

$$r_{N+1} = 1; \qquad \overline{r_{N+1}} = 0. \qquad [10]$$

Quant au passage de b à son complément restreint, il résulte simplement, pour chaque rang binaire, de la substitution de $\overline{b_n}$ à b_n, et inversement, dans les équations [6], [7] et [8].

D'où, pour l'étage de rang n :

$$s_n = \overline{r_{n+1}}\,(a_n\, b_n \vee \overline{a_n}\,\overline{b_n}) \vee r_{n+1}\,(a_n\,\overline{b_n} \vee \overline{a_n}\, b_n). \qquad [11]$$

$$r_n = a_n\,\overline{b_n} \vee r_{n+1}\,(a_n\, b_n \vee \overline{a_n}\,\overline{b_n}). \qquad [12]$$

$$\overline{r_n} = \overline{a_n}\, b_n \vee \overline{r_{n+1}}\,(a_n\, b_n \vee \overline{a_n}\,\overline{b_n}). \qquad [13]$$

Du point de vue technique, le remplacement de b_n par son complément restreint s'obtient très simplement grâce à trois nouveaux inverseurs complémentaires des précédents, les contacts « travail » des inverseurs de soustraction étant reliés aux contacts « repos » des inverseurs d'addition, et inversement. Les circuits correspondants sont représentés en traits interrompus sur la *figure 14*. La commutation entre les sorties d'addition et de soustraction est assurée par un relais a_c, qui doit être excité pour la soustraction et au repos pour l'addition. Ce relais sera lui-même excité au moment opportun par le décodeur d'instructions.

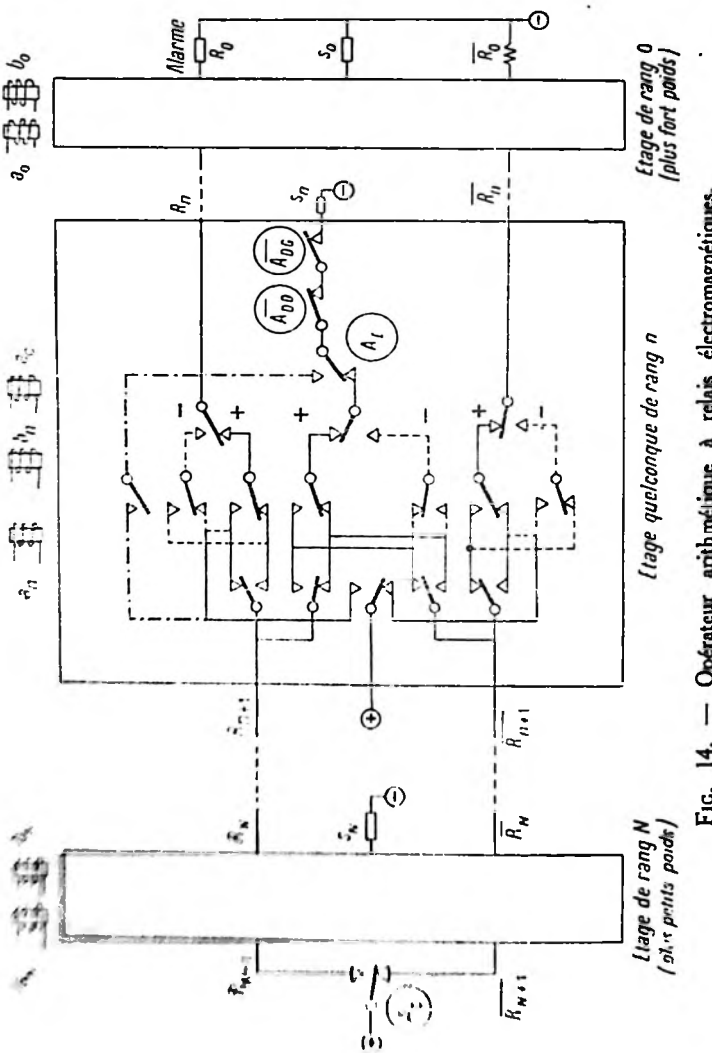


FIG. 14. — Opérateur arithmétique à relais électromagnétiques.

——— Circuit d'addition.

- - - - - - - - - Circuit de soustraction (commandé par A_S).

— . — . — . Circuit d'intersection (commandé par A_I, voir plus loin).

A_{DD} et A_{DG} : Commandes de décalage (voir plus loin).

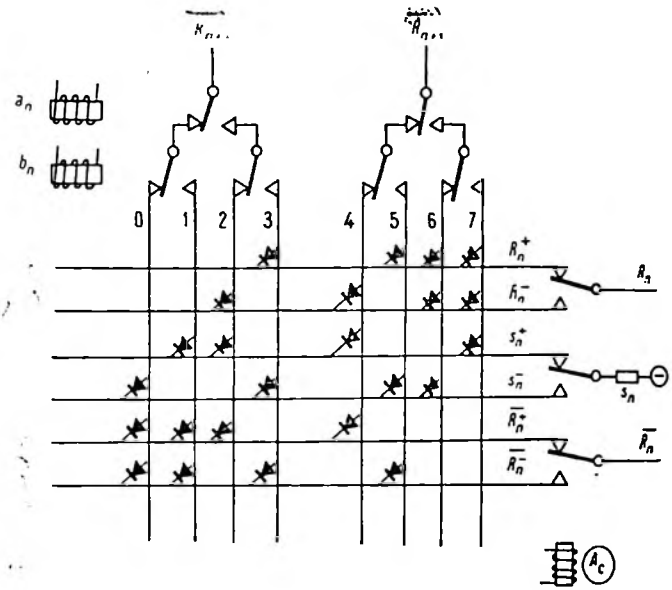
Nous ne nous occuperons pas, pour le moment, du circuit dessiné en trait mixte sur la *figure 14*, ni des contacts A_J, A_{DD} et A_{DG}, qui nous serviront dans la suite.

Enfin, pour l'étage de plus petit poids, de rang N, un nouvel inverseur A_C permet de satisfaire, soit les équations [6] pour l'addition, soit les équations [10] pour la soustraction.

Dans l'étage de plus fort poids, de rang 0, les circuits de formation de la retenue peuvent être supprimés. Toutefois, il peut être avantageux d'utiliser la ligne R_0, pour actionner un circuit d'alarme, qui indique que la capacité de l'opérateur a été dépassée et arrête automatiquement la machine.

AUTRE FORME D'ADDITIONNEUR-SOUSTRACTEUR. — Si l'on consent à utiliser, outre les relais, des redresseurs, on obtient le schéma remarquablement simple de la *figure 15*. Les 8 combinaisons possibles des trois variables, a_n, b_n et r_{n+1} sont rappelées dans le tableau de cette figure. Pour chacune de ces combinaisons, l'une des 8 lignes verticales numérotées de 0 à 7, comme les lignes du tableau, est excitée à l'exclusion des 7 autres. Considérons, par exemple, la combinaison 4. L'examen du tableau nous montre que les lignes s_n^+, R_n^+ et R_n^- doivent être mises sous tension. Il est clair que cette connexion ne peut se faire par de simples points de jonction, car alors le courant pourrait se propager d'une ligne horizontale vers une autre ligne verticale et, de là, vers d'autres lignes horizontales qui seraient alors excitées hors de propos. Il faut donc effectuer les connexions indiquées par le tableau au moyen de redresseurs, qui n'autorisent le passage du courant que dans le sens allant des lignes verticales aux lignes horizontales. C'est ce qui est représenté sur la *figure 15*. En outre, ici encore, un jeu de trois inverseurs commandés par un relais a_C permet de sélecter les sorties d'addition ou de soustraction sous la conduite du décodeur d'instructions.

Nous voyons, sur cet exemple, que le grand avantage d'une telle « *matrice de commutation* » ou de *codage* réside dans le fait qu'elle constitue la traduction fidèle du tableau des conditions logiques auxquelles doit obéir le système considéré.



	a_n	b_n	R_{n+1}	S_n^+	R_n^+	S_n^-	R_n^-
0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0
2	1	0	0	1	0	0	1
3	1	1	0	0	1	1	0
4	0	0	1	1	0	0	1
5	0	1	1	0	1	1	0
6	1	0	1	0	1	1	1
7	1	1	1	1	1	0	1

FIG. 15. — Additionneur-soustracteur à relais et redresseurs.

Registres de mémoire.

Nous venons de constituer, au moyen de relais, l'organe de calcul fondamental d'un opérateur arithmétique capable d'exécuter les opérations d'addition et de soustraction, à partir desquelles peuvent être construites toutes les opérations arithmétiques. Mais les deux nombres a et b doivent donc pouvoir être emmagasinés provisoirement dans des organes capables d'alimenter convenablement les enroulements de commande des relais a_n et b_n des étages successifs. De même, un organe similaire doit être prêt à recevoir le résultat du calcul.

Nous voyons ainsi apparaître la nécessité d'une fonction nouvelle : la fonction de *mémoire*. Nous appellerons *registre de mémoire* un organe capable d'emmagasiner un nombre de N chiffres binaires.

Un registre de mémoire sera donc constitué par N organes ou cellules binaires susceptibles de prendre 2 états définis (et 2 seulement) caractéristiques des chiffres 0 et 1 respectivement. Nous passerons en revue ultérieurement les différents phénomènes physiques permettant de constituer de telles mémoires élémentaires. Contentons-nous pour le moment de les imaginer, selon le symbolisme de la *figure 16*, comme étant constituées chacune de deux moitiés susceptibles de prendre, soit l'état (+), soit l'état (—), mais toujours de

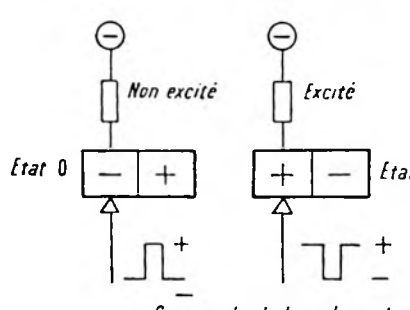
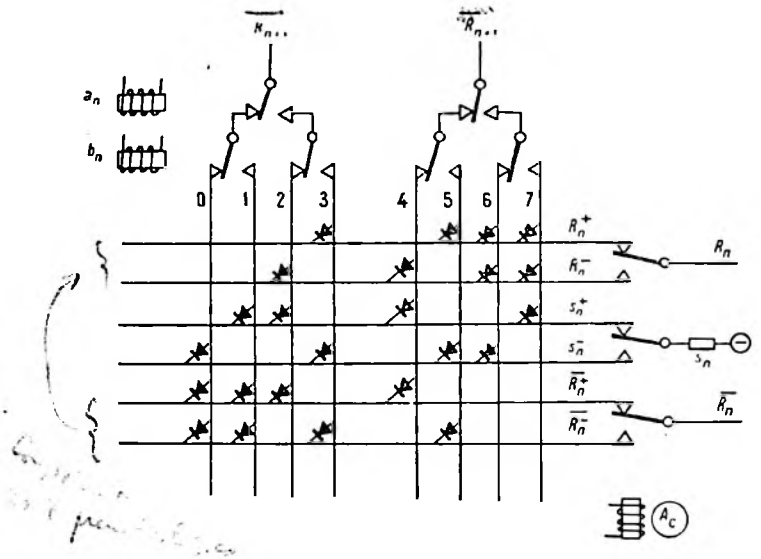


FIG. 16. — Représentation schématique d'une cellule de mémoire.



	a_n	b_n	R_{n+1}	S_n^+	R_n^+	S_n^-	R_n^-
0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0
2	1	0	0	1	0	0	1
3	1	1	0	0	1	1	0
4	0	0	1	1	0	0	1
5	0	1	1	0	1	1	0
6	1	0	1	0	1	1	1
7	1	1	1	1	1	0	1

FIG. 15. — Additionneur-soustracteur à relais et redresseurs.

Registres de mémoire.

Nous venons de constituer, au moyen de relais, l'organe de calcul fondamental d'un opérateur arithmétique capable d'exécuter les opérations d'addition et de soustraction, à partir desquelles peuvent être construites toutes les opérations arithmétiques. Mais les deux nombres a et b doivent donc pouvoir être emmagasinés provisoirement dans des organes capables d'alimenter convenablement les enroulements de commande des relais a_n et b_n des étages successifs. De même, un organe similaire doit être prêt à recevoir le résultat du calcul.

Nous voyons ainsi apparaître la nécessité d'une fonction nouvelle : la fonction de *mémoire*. Nous appellerons *registre de mémoire* un organe capable d'emmagasiner un nombre de N chiffres binaires.

Un registre de mémoire sera donc constitué par N organes ou cellules binaires susceptibles de prendre 2 états définis (et 2 seulement) caractéristiques des chiffres 0 et 1 respectivement. Nous passerons en revue ultérieurement les différents phénomènes physiques permettant de constituer de telles mémoires élémentaires. Contentons-nous pour le moment de les imaginer, selon le symbolisme de la *figure 16*, comme étant constituées chacune de deux moitiés susceptibles de prendre, soit l'état (+), soit l'état (—), mais toujours de

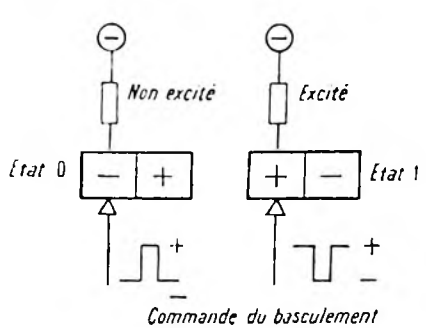


FIG. 16. — Représentation schématique d'une cellule de mémoire.

signes contraires. La configuration (— +) représentera, par exemple le chiffre 0 et la configuration (+ —) le chiffre 1. Si l'enroulement d'un relais est connecté entre la moitié gauche et le pôle (—), il sera donc excité dans l'état 1, et non excité dans l'état 0. En outre, chaque moitié possède une borne de commandes schématisée par une flèche. Nous admettrons que, pour faire passer cette mémoire élémentaire de l'état 0 à l'état 1, il suffit de porter, pendant un court instant, la borne de commande de la moitié gauche au niveau (+); inversement, on passera de l'état 1 à l'état 0 en portant cette borne un instant au niveau (—). Le même effet serait obtenu par une action symétrique sur la moitié droite. Il est bien entendu qu'en l'absence de telles manœuvres, l'état du système se conserve indéfiniment.

On remarquera que ce mode de fonctionnement est essentiellement celui des « *bascules* » électroniques ou *multivibrateurs bistables*, constitués par deux tubes interconnectés de telle manière que le système présente deux états stables dans chacun desquels l'un des tubes est conducteur tandis que l'autre est bloqué. Le système change d'état si l'on applique une impulsion positive à la grille du tube bloqué ou une impulsion négative à celle du tube conducteur.

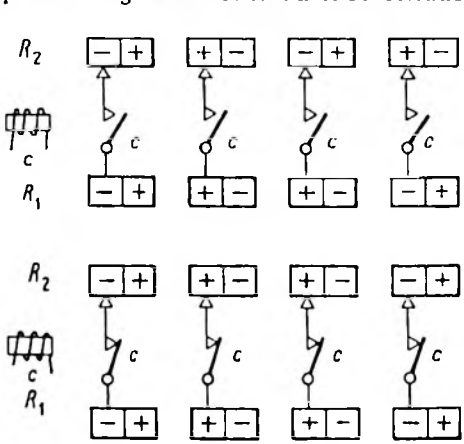


FIG. 17. — Transfert entre deux registres par fermeture des contacts C.

Mais ne nous égarons pas dans la technologie et revenons à notre modèle schématique. Pour transférer un nombre d'un registre R_1 à un registre R_2 (*fig. 17*), il suffit donc de fermer pendant un instant les contacts C commandés par le relais c. Il est entendu que les actions ne se produisent que dans le sens des flèches et que, par conséquent, l'état du registre R_1 n'est pas modifié par celui de R_2. Dans cette opération, on dit que le registre R_1 est « lu » et que son contenu est « écrit » dans le registre R_2. On voit donc que

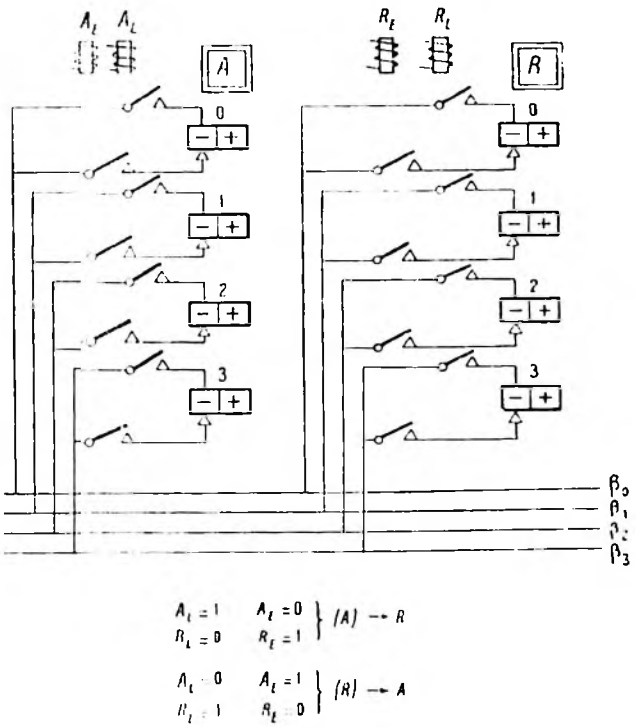


FIG. 18. — Transfert entre deux registres par l'intermédiaire d'un jeu de barres.

les propriétés essentielles d'un registre de mémoire sont, d'une part, de conserver son état à la lecture et, d'autre part, à l'écriture, de voir son ancien contenu remplacé par le nombre incident.

Dans une calculatrice numérique automatique, les données numériques du calcul, ainsi que les résultats intermédiaires, sont emmagasinées dans une *mémoire* comprenant souvent un nombre considérable de registres — quelques dizaines de milliers par exemple. Les registres de la mémoire ne communiquent pas directement entre eux, mais ils sont en liaison avec le ou les registres de l'opérateur arithmétique par l'intermédiaire d'un *jeu de barres* (*fig. 18*) comprenant autant de lignes β que de chiffres dans les nombres traités. Ainsi, pour transmettre un nombre du registre R au registre A, il faut fermer les contacts R_L et A_E provoquant respectivement la lecture de R et l'écriture dans A. La liaison inverse résulte de la fermeture des contacts de lecture A_L et d'écriture R_E. Sur la figure, nous avons noté (A) le contenu du registre A.

Mémoire élémentaire a relais. — La *figure 19* montre, à titre d'exercice, comment une cellule élémentaire de mémoire peut être constituée au moyen de deux relais p et q, dont l'un, p, comporte deux enroulements de commande p_1 et p_2. La barre β, qui lui correspond, est ici constituée

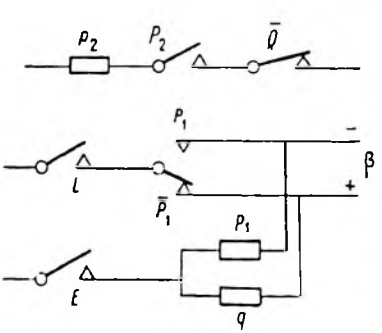


Fig. 19. — Cellule de mémoire à relais.

par une paire de fils dont l'un est au (+) et l'autre au (—), la situation représentée sur la figure correspondant à l'état 0 de la barre. Ainsi donc, si l'on ferme le contact de lecture L, la barre prend le même état que le relais p, sans modifier ce dernier. Si l'on ferme le contact d'écriture E, alors que la barre est dans l'état 0, le relais q se trouve excité, le contact $\overline{Q}$ s'ouvre et le relais p se trouve donc désexcité au cas où il était excité préalablement; au contraire, si la barre est dans l'état 1, c'est le relais p qui se trouve excité par son enroulement p_1, et le contact P_2 d'entretien se ferme; le contact $\overline{Q}$ étant fermé, puisque q n'est pas excité, on voit que l'enroulement d'entretien p_2 se trouve alimenté, et que les contacts P conservent donc leur état (contraire à celui de la figure), même après l'ouverture du contact d'écriture E.

ORGANISATION D'UN TOTALISATEUR. — Il est fréquent, et nous supposerons que c'est le cas dans notre machine hypothétique, que la fonction d'addition soit remplacée par la *totalisation*. Le *totalisateur* est un organe qui possède un registre de mémoire propre A, dont le contenu s'ajoute à tout nombre qu'il reçoit par ailleurs du jeu de barres, la somme obtenue devant alors se substituer à l'ancien contenu du registre. On peut ainsi totaliser une suite de nombre en les envoyant successivement sur le jeu de barres.

A partir de maintenant, nous supposerons que notre machine traite des nombres de 10 chiffres binaires (1), y compris le chiffre à gauche de la virgule. L'opération de totalisation se fait en deux temps (*fig. 20*). A l'instant t_1 où le pôle (+) de l'additionneur-soustracteur est alimenté, la somme du contenu du registre A et du nombre porté à cet instant par les barres se forme et s'inscrit dans le registre auxiliaire A', car les contacts $\overline{X'}$ sont alors fermés. Puis, à l'instant t_2 où le relais x' est alimenté, les contacts X' s'ouvrent, ce qui a pour effet d'isoler le registre A' des

(1) Dans les machines réelles, les nombres traités comprennent plusieurs dizaines de chiffres binaires.

circuits de calcul, et les contacts X' se ferment, provoquant le transfert du contenu de A' dans A. Mais l'alimentation du relais x' est subordonnée à la fermeture du contact X, car ce bouclage n'est à réaliser que pour un certain nombre d'opérations bien déterminées.

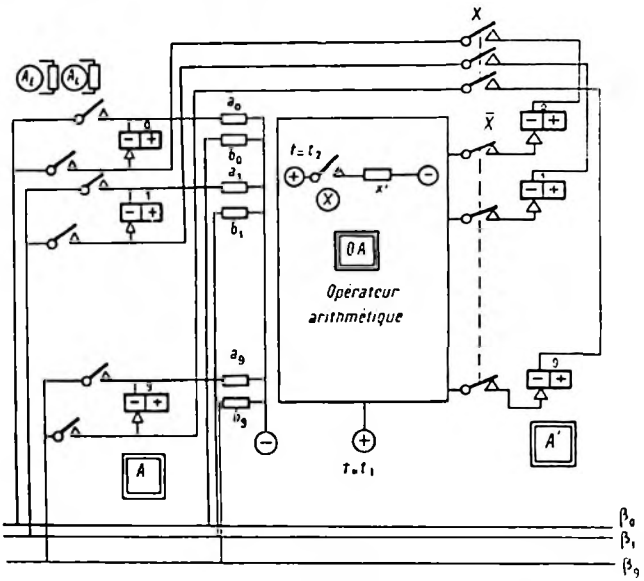


FIG. 20. — Organisation générale d'un opérateur arithmétique.

Bien entendu, des contacts A_L et A_E permettent, le moment venu, de lire le contenu de A ou d'y inscrire un nombre lu dans la mémoire.

Pour la première fois, nous voyons apparaître dans ce circuit un fonctionnement séquentiel faisant intervenir la variable temps. Nous verrons progressivement, dans la suite, toute l'importance de l'organisation des phases dans le fonctionnement d'une calculatrice numérique automatique.

ORGANISATION DE LA MÉMOIRE (*fig. 21*). — Supposons, pour fixer les idées que la mémoire comprenne 64 registres de 10 chiffres chacun (*fig. 21*), numérotés de 0 à 63, capacité d'ailleurs tout à fait insuffisante pour une calculatrice automatique. Les bornes « lecture » et « écriture » du registre

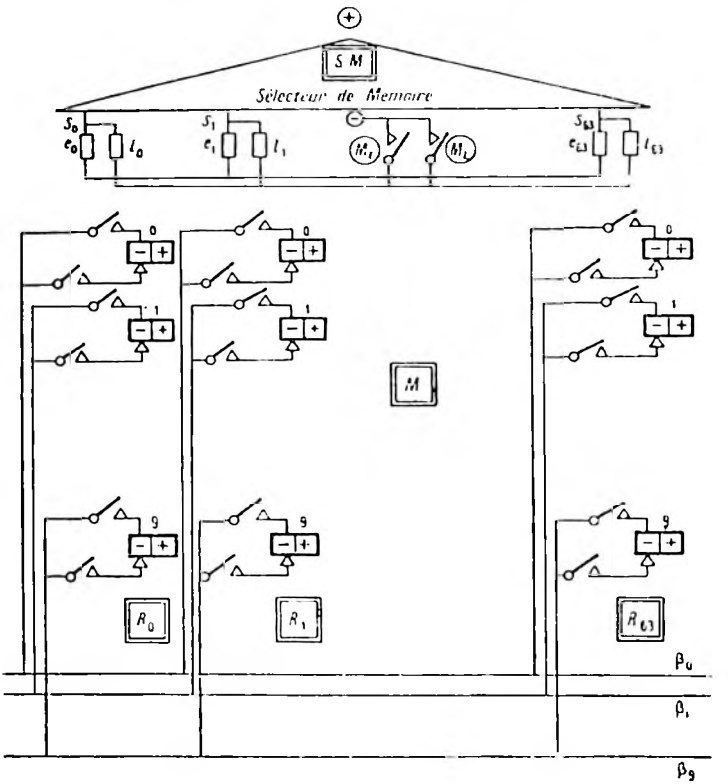


FIG. 21. — Organisation de la mémoire : la fermeture de l'un des contacts M_L ou M_E, commandés par le décodeur, entraîne la liaison avec les barres des bornes de lecture ou d'écriture des cellules du registre correspondant à la sortie du sélecteur se trouvant sous tension.

de rang i peuvent être reliées au jeu de barres par les contacts L_i et E_i respectivement (*fig. 21*). Ces contacts sont eux-mêmes commandés par les enroulements l_i et e_i.

L'utilisation de cette mémoire exige que l'on puisse sélecter l'une des mémoires et provoquer, soit son écriture, soit sa lecture. A cet effet, chaque couple d'enroulements l_i et e_i est relié, d'une part, à l'une des lignes S_i sortant du *sélecteur de mémoire*, d'autre part à une paire de contacts M_L et M_R communs à tous les enroulements. Le fonctionnement du sélecteur, que nous étudierons dans un instant, est tel que la ligne S_i correspondant à la mémoire sélectée soit mise sous tension. A ce moment, cette mémoire est lue ou écrite selon que l'on ferme le contact de lecture M_L, qui ferme le circuit de l'enroulement l_i, ou le contact d'écriture M_E qui ferme le circuit de l'enroulement e_i. L'état de toutes les autres mémoires non sélectées reste inchangé.

SÉLECTEUR DE MÉMOIRE. — Le numéro d'ordre d'un registre dans la mémoire est appelé son « *adresse* », c'est-à-dire sa position dans la mémoire.

Une adresse comprise entre 0 et $(2^p - 1)$ s'exprime au moyen de p chiffres binaires, que l'on peut inscrire dans un

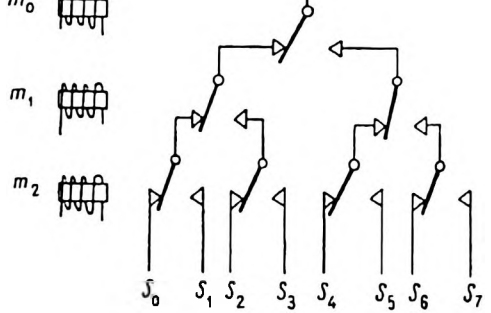


FIG. 22. — Sélecteur à 8 voies.

registre de p cellules élémentaires alimentant chacune un enroulement de commande de relais. Dans notre machine hypothétique, p est égal à 6 (en effet, $2^6 = 64$). Notre registre de sélecteur comprend donc 6 mémoires élémentaires alimentant 6 enroulements de relais appelés m_4 à m_9 sur la *figure 18*. Nous verrons ultérieurement la raison de cette numérotation.

Pour simplifier les choses, examinons le fonctionnement d'un sélecteur à 8 positions S_0 à S_7 commandé par 3 relais m_0, m_1 et m_2 alimentés par un registre à 3 chiffres binaires (*fig. 22*). Ce sélecteur doit donc satisfaire les huit équations logiques :

$$S_0 = \overline{m_0}\ \overline{m_1}\ \overline{m_2}$$
$$S_1 = \overline{m_0}\ \overline{m_1}\ m_2$$
$$S_2 = \overline{m_0}\ m_1\ \overline{m_2}$$
$$S_3 = \overline{m_0}\ m_1\ m_2$$
$$S_4 = m_0\ \overline{m_1}\ \overline{m_2}$$
$$S_5 = m_0\ \overline{m_1}\ m_2$$
$$S_6 = m_0\ m_1\ \overline{m_2}$$
$$S_7 = m_0\ m_1\ m_2$$

Les seconds membres représentent l'état des relais pour chacun des états du registre correspondant aux nombres de 0 à 7.

Ce système d'équations est satisfait par la pyramide d'inverseurs représentée sur la *figure 22*, comme on peut facilement le vérifier.

En effet, les contacts $\overline{M}_0$ peuvent être communs aux lignes S_0 à S_3, et les contacts M_0 communs aux lignes S_4 à S_7. De même, un premier contact $\overline{M}_1$ peut être commun aux lignes S_0 et S_1, et un premier contact M_1 commun aux lignes S_2 et S_3, cependant qu'un second contact $\overline{M}_1$ peut

être commun aux lignes S_4 et S_5, et un second contact M_1 commun aux lignes S_6 et S_7. Pour terminer, il faut évidemment autant d'inverseurs M_2 qu'il y a de paires de lignes de sortie. Nous voyons là un premier exemple de la réduction du nombre des contacts qui peut être obtenue par l'examen systématique d'un système d'équations logiques tel que

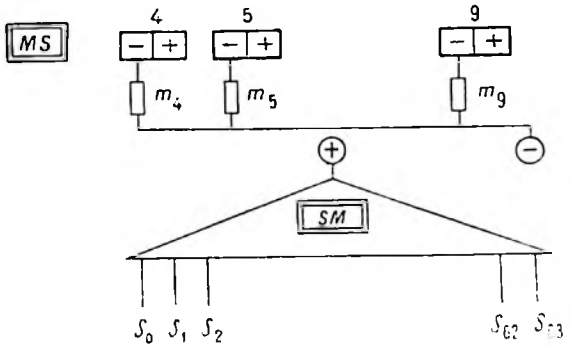


FIG. 23. — Sélecteur à 64 voies et sa mémoire.

le système ci-dessus définissant le fonctionnement de notre sélecteur.

La transposition à un sélecteur à 64 sorties et 6 relais de commande est immédiate. Les 6 relais, numérotés de 4 à 9, pour une raison qui apparaîtra plus loin, sont alimentés par un registre spécial appelé « *mémoire du sélecteur* » *MS*, dans lequel est inscrite au préalable l'adresse de la mémoire à sélecter (*fig. 23*).

Notions de code et de programme.

Nous disposons maintenant d'un opérateur arithmétique simple et d'une mémoire. Nous savons provoquer le transfert dans le registre *A* de l'opérateur du nombre d'adresse inscrite dans le registre *MS* du sélecteur de mémoire par la fermeture des contacts A_L de l'opérateur et des contacts M_K

de la mémoire. Inversement, on peut transférer le contenu du registre de l'opérateur dans la mémoire d'adresse indiquée sur le registre du sélecteur en fermant les contacts A_E et M_L. L'opération d'addition du contenu d'une mémoire sélectée au contenu du registre de l'opérateur nécessite la fermeture des contacts M_L, qui réunit les sorties « lecture » de la mémoire sélectée au jeu de barres, et la fermeture du contact X, qui provoquera en son temps le bouclage de l'opérateur. L'exécution d'une soustraction exige, en outre, le basculement des inverseurs A_C de l'opérateur.

Il nous reste à voir comment et à quels instants ces diverses opérations sont effectuées; en d'autres termes, nous devons étudier maintenant l'organisation des *organes de commande*.

Ce sont les notions de *code* et de *programme* qui vont nous permettre de dégager les conditions que doivent remplir ces organes.

Une machine est capable d'exécuter un nombre limité d'opérations élémentaires fixées à l'avance, dont chacune constitue l'exécution d'une instruction donnée. La liste des instructions que la machine est ainsi capable d'interpréter et d'exécuter constitue son *code*. Chacune des instructions du code est repérée par un numéro d'ordre permettant de l'identifier. En numération binaire, un numéro de 4 chiffres permet par exemple de spécifier une instruction parmi un code de seize instructions.

La liste des instructions correspondant à l'exécution d'un calcul donné constitue un *programme*. Pour la commodité du programmeur, qui prépare ce programme sur une feuille de papier avant de le communiquer à la machine, il est commode d'associer une lettre, appelée souvent *lettre de fonction*, à chaque instruction du code; on peut considérer le numéro d'ordre correspondant comme l'équivalent binaire de cette lettre.

Mais l'indication d'une lettre de fonction ne suffit pas pour définir une opération élémentaire. Cette dernière met, en effet, généralement en jeu un registre de la mémoire, dont il est, par conséquent, nécessaire d'indiquer l'adresse. Une instruction complète se compose donc de la juxtaposition

d'une lettre de fonction et d'une adresse de la mémoire, exprimées toutes deux par leurs équivalents binaires. De même que le programmeur, sur sa feuille de papier, exprime chaque fonction par une lettre, il écrit les adresses en numération décimale. Ce sont les organes d'entrée de la machine qui se chargeront de traduire ces indications dans le système binaire.

Nous voyons donc que, dans notre machine hypothétique, une instruction s'exprime au moyen de 10 chiffres binaires, 4 pour la lettre de fonction (rangs de 0 à 3) et 6 pour l'adresse (rangs de 4 à 9). Nous voyons maintenant pourquoi nous avons baptisé m_4 à m_9 les relais de commande de notre sélecteur de mémoire.

Au début de ce paragraphe, nous avons envisagé quatre opérations élémentaires, qu'il est maintenant utile de définir avec précision en les associant chacune à une lettre de fonction et à son équivalent binaire :

A n (*A* = *addition* : 0001) : ajouter le contenu de la mémoire d'adresse *n* à celui du registre de l'opérateur et conserver la somme ainsi obtenue dans ce même registre.

S n (*S* = *soustraction* : 0010) : soustraire le contenu de la mémoire d'adresse *n* de celui du registre de l'opérateur et conserver la différence dans ce même registre.

R n (*R* = *remplacement* : 0011) : remplacer le contenu du registre de l'opérateur par celui de la mémoire d'adresse *n*.

T n (*T* = *transfert* : 0100) : remplacer le contenu de la mémoire d'adresse *n* par celui du registre de l'opérateur.

C'est là un code très rudimentaire, que nous enrichirons progressivement de nouvelles instructions. Insistons bien sur le fait que les nombres dont on a besoin sont spécifiés, non par leurs valeurs numériques, mais uniquement par leurs adresses dans la mémoire.

Montrons tout de suite un exemple d'emploi de cet embryon de code. Supposons que nous désirions effectuer l'opération suivante :

$$a + b - c = d$$

les nombres a, b et c ayant été préalablement inscrits dans les mémoires 20, 21 et 22 par exemple. Dans la suite, nous représenterons le contenu d'une mémoire par son adresse entre parenthèses. Si, de plus, nous désirons que le résultat de l'opération ci-dessus soit inscrit dans la mémoire 23, nous pourrons donc écrire schématiquement :

$$(20) + (21) - (22) \rightarrow 23.$$

En utilisant les instructions définies plus haut, ce calcul sera effectué par l'exécution du programme suivant, expliqué ligne par ligne :

Instructions	Explications	Explications abrégées
R 20	Inscription de (20) dans le registre de l'opérateur.....	(20) → *A*
A 21	Addition de (21) au contenu du registre de l'opérateur : formation de (20) + (21).....	(*A*) + (21) → *A*
S 22	Soustraction de (22) du contenu du registre de l'opérateur : formation de (20) + (21) — (22).....	(*A*) — (22) → *A*
T 23	Transfert du contenu du registre de l'opérateur dans la mémoire 23	(*A*) → 23

La question se pose maintenant de savoir sous quelle forme les instructions de ce programme sont communiquées à la machine. Pour que son fonctionnement soit automatique, les instructions seront introduites dans la mémoire, par exemple aux adresses 0, 1, 2 et 3 avant le début de l'exécution du programme, et les organes de commande seront

organisés de telle manière qu'ils soient exécutés successivement dans l'ordre convenable. Nous comprenons maintenant pourquoi nous nous sommes arrangés pour que les instructions et les nombres comportent le même nombre de chiffres binaires, 10 dans le cas présent. La mémoire seule ne sait pas distinguer ces deux types de sigles, sans le concours des organes de commande. C'est pourquoi on donne quelquefois la dénomination commune de « *mots* » indifféremment aux nombres et aux instructions.

Notons encore que le type de code envisagé ci-dessus, dans lequel chaque instruction comporte l'indication d'une adresse unique de la mémoire, est appelé pour cette raison « code à simple adresse ». Les quatre opérations considérées jusqu'ici mettant en réalité en jeu deux nombres et un résultat, l'adresse du second terme, ainsi que celle du résultat, est spécifiée implicitement dans le libellé de l'instruction : dans les instructions considérées, c'est celle du registre de l'opérateur. De même, il est entendu que, sauf indication contraire, les instructions d'un programme sont exécutées dans l'ordre de leur inscription en mémoire. Nous verrons n temps utile ce qui se cache derrière notre restriction sauf indication contraire ».

Il existe également des codes à deux adresses, dont les nstructions contiennent les indications des adresses de deux opérateurs, et des codes à trois adresses, qui spécifient en outre l'adresse du résultat de l'opération. D'autre part, certains codes n'impliquent pas l'exécution séquentielle des instructions, et chacune de leurs instructions contient l'indication de l'adresse de l'instruction suivante.

Mais revenons à notre machine hypothétique, dont nous sommes maintenant en mesure d'étudier l'organisation générale des organes de commande.

Organisation générale des organes de commande (*fig. 24*).

Il résulte immédiatement de ce qui précède qu'à chaque instruction du programme doit correspondre deux phases bien distinctes. Au cours de la première, dite « *phase instruc-*

tion », l'instruction à exécuter est extraite de la mémoire et mise en réserve provisoirement dans un registre spécial appelé *mémoire d'instruction MI*. Au cours de la seconde phase, dite « *phase nombre* », l'instruction contenue dans la mémoire d'instruction est interprétée ou, comme l'on dit, « décodée », et exécutée.

L'adresse de l'instruction à exécuter pendant la phase

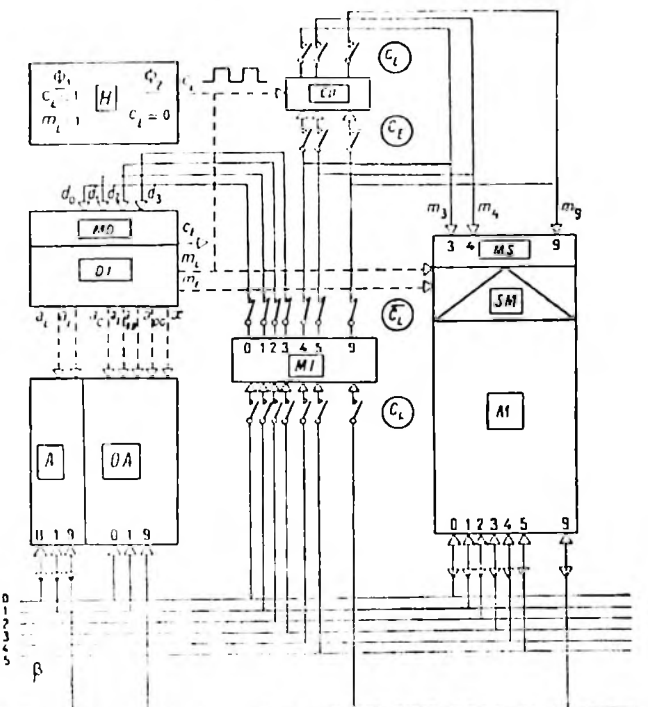


FIG. 24. — Organisation générale d'une calculatrice numérique automatique; pendant la phase « instruction » Φ_1, le relais c_i est excité, ainsi que le relais m_i de lecture de mémoire; pendant la phase « nombre » Φ_2, le relais c_i n'est pas excité; le relais c_i est excité pendant la phase « nombre » en cas de rupture de séquence.

instruction est indiquée par un compteur *CO* appelé « *compteur ordinal* », dont le contenu progresse d'une unité au début de chaque phase nombre, de manière à être prêt à commander la phase instruction suivante. Avant la mise en marche de la machine, on inscrira donc dans ce compteur l'adresse de la première instruction à exécuter, 0 dans le cas du programme considéré plus haut.

Enfin, l'enchaînement des phases sera réglé par un organe de synchronisation, appelé « *horloge* », dont la borne de sortie se trouve reliée alternativement et à cadence régulière au (+) et au (—).

Entrons dans le détail de ce fonctionnement, à la lumière de la *figure 24*.

Phase instruction Φ_1. — Pendant cette phase, le relais c_L, commandé par l'horloge, est excité. Les contacts C_L sont donc fermés et les contacts $\overline{C_L}$ ouverts. Il en résulte que le contenu du compteur ordinal *CO*, qui indique l'adresse de l'instruction à exécuter, est transféré dans le registre du sélecteur de mémoire *MS*. De plus, pendant la phase instruction, le relais m_L de la mémoire est également excité par l'horloge; le contenu de la mémoire sélectée apparaît donc sur les barres β_0 à β_9, d'où il est transféré dans la mémoire d'instruction *MI* par le second jeu de contacts C_L. Ainsi se termine la phase instruction. Le libellé de l'instruction à exécuter se trouve maintenant dans la mémoire d'instruction.

Phase nombre Φ_2. — Pendant cette seconde phase, les contacts C_L sont ouverts et les contacts $\overline{C_L}$ fermés. La partie adresse (chiffres 4 à 9) de l'instruction à exécuter est transférée dans le registre *MS* du sélecteur de mémoire. La partie lettre de fonction ou code d'opération (chiffres 0 à 3) est transférée dans le registre *MD* du décodeur d'instruction. Ce dernier est chargé d'exciter les relais de l'opérateur arithmétique et de la mémoire correspondant à l'opération à exécuter. Son organisation sera étudiée ultérieurement.

Prenons l'exemple du programme construit plus haut, dont l'équivalent binaire de la première instruction R 20 s'écrit :

0 0 0 1 0 1 0 1 0 0

La mémoire 20 se trouve sélectée par le transfert dans le registre du sélecteur des 6 chiffres de droite. Le transfert dans le registre du décodeur des 4 chiffres de gauche provoque l'excitation du relais m_L de la mémoire et du relais a_E de l'opérateur, entraînant le transfert du contenu de la mémoire 20 dans le registre de l'opérateur par l'intermédiaire du jeu de barre.

Pendant ce temps, le contenu du compteur ordinal, qui était 000000 au début du calcul, est devenu 000001, de sorte que, pendant la phase instruction suivante, l'instruction suivante, contenue dans la mémoire 1, est transférée dans la mémoire d'instruction. Cette instruction s'écrit :

0 0 0 1 0 1 0 1 0 1

Pendant la phase nombre suivante, la mémoire 21 est sélectée et le décodeur, ayant reçu dans son registre la configuration 0001, commande l'excitation du relais m_L de la mémoire (lecture) et du relais x de l'opérateur (bouclage), de sorte que la somme des contenus du registre A et de la mémoire 21 se forme d'abord dans le registre A', puis est transférée dans A.

La phase nombre suivante ne diffère de la précédente, outre la substitution de la mémoire 22 à la mémoire 21, que par l'excitation du relais a_C de l'opérateur, qui remplace l'addition par la soustraction.

Enfin, pendant la phase nombre de la dernière instruction du programme, la mémoire 23 est sélectée, et le décodeur doit commander l'excitation du relais m_E de la mémoire (écriture) et du relais a_L de l'opérateur (lecture), de sorte que le résultat du calcul, contenu dans le registre de l'opérateur, se trouve enfin transféré dans la mémoire 23.

Ruptures de séquence.

En précisant plus haut que les instructions étaient en principe exécutées dans l'ordre de leur inscription en mémoire, nous avons toutefois fait une restriction en signalant la possibilité d'une indication contraire. Cette circonstance se présentera lorsque, ayant exécuté une partie d'un certain calcul, on désirera le poursuivre selon une portion de programme inscrite en un endroit différent de la mémoire. Cette possibilité de *rompre la séquence* des instructions permet en outre d'utiliser plusieurs fois un même programme élémentaire au cours d'un calcul. On pourra ainsi, en particulier, préparer à l'avance les programmes des processus de calcul dont on sait que l'on aura fréquemment besoin, les inscrire dans la mémoire et faire appel à eux par une rupture de séquence toutes les fois qu'il le faudra. De tels programmes élémentaires pouvant être incorporés dans un programme principal constituent ce que l'on appelle des « *sous-programmes* ». En fait, un sous-programme peut fort bien faire appel lui-même à un autre sous-programme, de sorte qu'il peut exister toute une hiérarchie de sous-programmes s'imbriquant les uns dans les autres. L'emploi commode d'une calculatrice numérique automatique exige qu'elle soit dotée d'une importante collection de sous-programmes.

Nous devons donc prévoir dans notre code une instruction de rupture de séquence ainsi rédigée :

$N\ n$ ($N = 0101$) : passer à l'exécution de l'instruction d'adresse n.

Autrement dit : substituer n au contenu actuel du compteur ordinal, quel qu'il soit. Cette substitution se fera au cours de la phase nombre Φ_2, pendant laquelle n constitue la partie adresse du contenu de la mémoire d'instruction. Le transfert de cette partie adresse dans le compteur ordinal se fait par fermeture des contacts C_{μ} sous l'effet de la présence de la combinaison 0101 dans le registre du décodeur d'instruction (*fig. 24*).

RUPTURES DE SÉQUENCE CONDITIONNELLES. — Une telle rupture de séquence est dite *inconditionnelle*, puisque son exécution n'est subordonnée à aucune condition préalable. Mais cela n'est pas suffisant. Revenons au cas d'un calcul par approximations successives. En pareil cas, il faut poursuivre le processus tant que la précision cherchée n'est pas atteinte. A cet effet, on calcule, après chaque itération, la différence $\Delta = |x_{n+1}| - |x_n|$ des valeurs absolues du résultat de l'itération précédente et de celui qu'on vient d'obtenir, et on la compare à la précision désirée ε. La comparaison de deux grandeurs peut toujours se ramener à l'examen du signe de leur différence. Ainsi, si la différence $\Delta - \varepsilon$ est positive, la précision désirée n'est pas atteinte et il y a donc lieu de procéder à une nouvelle itération, c'est-à-dire à une nouvelle rupture de séquence. Au contraire, si la différence est négative, le calcul est terminé et on peut donc reprendre le fonctionnement séquentiel.

Le test de signe portera sur le contenu du registre de l'opérateur, où se trouve précisément la différence après sa formation. Notre instruction de *rupture de séquence conditionnelle* sera donc ainsi rédigé :

P n ($P = 0110$) : si le contenu du registre de l'opérateur est positif ou nul (c'est-à-dire si $a_0 = 0$), passer à l'exécution de l'instruction d'adresse *n*; sinon, procéder séquentiellement.

Le décodeur d'instruction devra donc avoir connaissance de l'état de la cellule de rang 0 du registre de l'opérateur.

Pour pouvoir effectuer des multiplications, nous aurons de même besoin de tester le chiffre de plus petit poids du multiplicateur. En effet, dans le système binaire, le multiplicande convenablement décalé s'ajoute ou non à la somme des produits partiels déjà formés selon que le chiffre correspondant du multiplicateur est un 1 ou un 0. De plus, nous nous arrangerons, par des décalages successifs, pour que le chiffre testé soit toujours celui de plus petit poids, et pour que ce test se fasse dans le registre de l'opérateur.

Notre seconde instruction de rupture de séquence conditionnelle s'écrit donc :

Q n (*Q* = 0111) : si le chiffre de plus petit poids a_0 du contenu du registre de l'opérateur est 0, passer à l'exécution de l'instruction d'adresse *n*; sinon, procéder séquentiellement.

Code complet et décodeur d'instruction.

Complétons notre code par quelques instructions supplémentaires avant d'en faire l'inventaire définitif.

Décalages. — Nous venons de faire allusion plus haut à la nécessité de pouvoir faire subir aux nombres des *décalages*, en évoquant l'opération de multiplication. Nous remarquerons à ce propos que le décalage d'un nombre d'un rang binaire vers les forts poids, c'est-à-dire vers la gauche dans l'écriture normale des nombres, correspond à sa multiplication par 2; en effet, par cette opération, les poids de tous les rangs se trouvent augmentés d'une puissance de 2. Inversement, le décalage d'un rang binaire vers les faibles poids, c'est-à-dire vers la droite, équivaut à diviser le nombre par 2, à condition toutefois de ne pas changer le chiffre de plus fort poids, qui définit le signe.

Nous introduirons donc les deux ordres suivants :

G (gauche : 1000) : décaler le contenu du registre de l'opérateur d'un rang vers la gauche.

D (droite : 1001) : décaler le contenu du registre de l'opérateur d'un rang vers la droite.

Circuits de décalage de l'opérateur (*fig. 25*). — Les décalages se feront directement entre les registres *A* et *A'* de l'opérateur, au moyen des contacts A_{DG} pour le décalage

à gauche et des contacts A_{DD} pour le décalage à droite. Cette opération sera évidemment suivie du bouclage de l'opérateur (X). Il sera en outre nécessaire de couper la liaison entre le registre A' et l'additionneur-soustracteur, au moyen de deux contacts $\overline{A_{DG}}$ et $\overline{A_{DD}}$ en série avec les sorties de ce dernier (voir *fig. 14* et *15*).

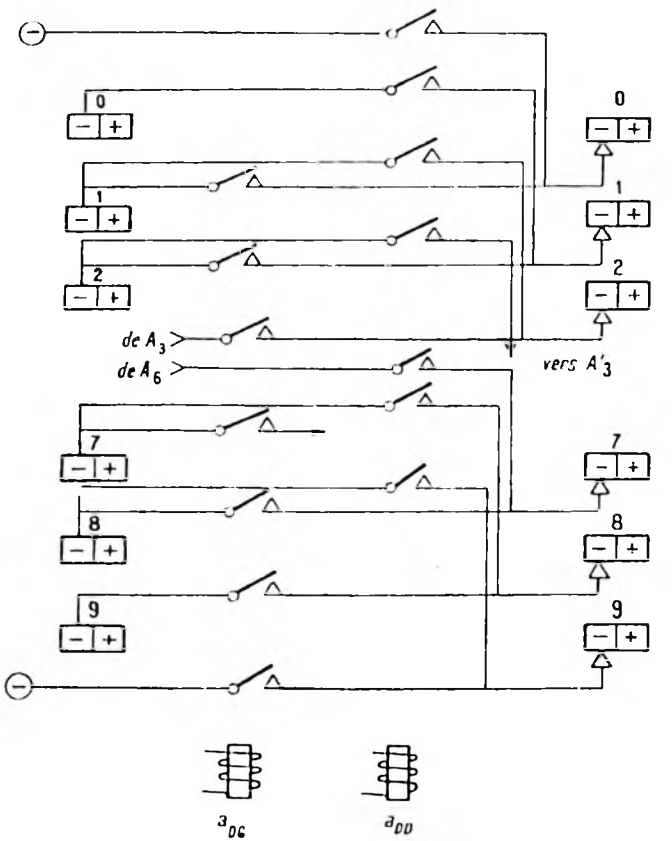


FIG. 25. — Circuits de décalage.

INTERSECTION. — Nous montrerons ultérieurement l'utilité de l'opération d'intersection de deux nombres. Nous introduirons donc dans notre code l'instruction :

I n (*I* = intersection : 1010) : prendre l'intersection du contenu de la mémoire d'adresse *n* avec celui du registre de l'opérateur, et laisser le résultat dans ce dernier.

Cette opération permet, par exemple, d'extraire un ou plusieurs chiffres d'un nombre, à l'exclusion des autres. Supposons, par exemple, que nous désirions extraire la partie adresse de l'instruction suivante :

0 1 1 0 1 0 0 1 1 0

Il nous suffit pour cela de prendre l'intersection de ce mot avec le suivant :

0 0 0 0 1 1 1 1 1 1

Appliquons l'opération d'intersection aux couples de chiffres de chaque rang de ces deux mots :

0 0 0 0 1 0 0 1 1 0

C'est bien le résultat cherché.

CIRCUIT D'INTERSECTION (voir *fig. 14*). — Il est constitué simplement par deux contacts A et B en série et par un inverseur A_I commandé par le relais a_I du décodeur et ayant pour effet de substituer, à la sortie de l'additionneur-soustracteur, le résultat de l'intersection à celui de l'addition ou de la soustraction. Ce circuit, qui utilise l'un des contacts A déjà disponibles, est représenté en trait mixte sur le schéma général de la *figure 14*. Cette opération doit évidemment être suivie du bouclage de l'opérateur (relais x).

INSTRUCTION D'ARRÊT. — L'instruction Z, de code d'opération 1011, commande l'arrêt de la machine.

RÉCAPITULATION DES INSTRUCTIONS; DÉCODEUR D'INSTRUCTION. — Nous avons, jusqu'à présent, 11 instructions distinctes, numérotées de 1 à 11. On évite, en général, d'utiliser les combinaisons 0000 et 1111, utilisées à d'autres fins. Nous avons donc encore trois combinaisons disponibles, qui nous seront utiles pour la commande des organes d'entrée et de sortie.

Le tableau suivant contient la liste des 11 instructions envisagées avec, pour chacune d'elles, l'équivalent binaire de la lettre de fonction, la définition abrégée de l'opération correspondante et l'indication des différents relais à exciter.

TABLEAU RÉCAPITULATIF DES INSTRUCTIONS

Instructions	Codes d'opérations $d_0 d_1 d_2 d_3$	Définitions abrégées	Relais à exciter
$A\ n$	0 0 0 1	$(A) + (n) \rightarrow A$	m_L x
$S\ n$	0 0 1 0	$(A) - (n) \rightarrow A$	m_L x $a_{t'}$
$R\ n$	0 0 1 1	$(n) \rightarrow A$	m_L a_E
$T\ n$	0 1 0 0	$(A) \rightarrow n$	m_E a_L
$N\ n$	0 1 0 1	$(n) \rightarrow CO$	c_E
$P\ n$	0 1 1 0	$(n) \rightarrow CO$ si $a_0 = 1$	c_E
$Q\ n$	0 1 1 1	$(n) \rightarrow CO$ si $a_9 = 1$	c_E
G	1 0 0 0	$2(A) \rightarrow A$	x a_{D0}
D	1 0 0 1	$(A)/2 \rightarrow A$	x a_{D9}
$I\ n$	1 0 1 0	$(A_t) \cdot (n_i) \rightarrow A_t$	m_L x a_I
Z	1 0 1 1	Arrêt	z

La confrontation de la deuxième et de la quatrième colonne du tableau ci-dessus, indiquant respectivement le code d'opération de chaque instruction et les relais à exciter,

nous permet d'écrire, pour chacun de ces relais, l'équation logique à laquelle il doit satisfaire :

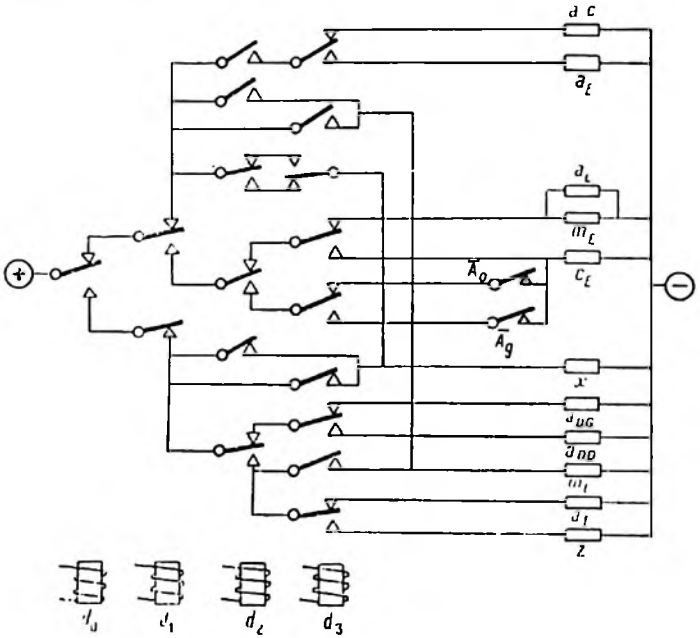


FIG. 26. — Décodeur d'instructions à relais électromagnétiques.

$$m_L = \overline{d_0}\,\overline{d_1}\,(d_2 \vee d_3) \vee d_0\,\overline{d_1}\,d_2\,\overline{d_3};$$

$$m_E = \overline{d_0}\,d_1\,\overline{d_2}\,\overline{d_3};$$

$$x = \overline{d_0}\,\overline{d_1}\,(d_2\,\overline{d_3} \vee \overline{d_2}\,d_3) \vee d_0\,\overline{d_1}\,(\overline{d_2} \vee \overline{d_3});$$

$$a_C = \overline{d_0}\,\overline{d_1}\,d_2\,\overline{d_3};$$

$$a_E = \overline{d_0}\,\overline{d_1}\,d_2\,d_3;$$

$$a_L = \overline{d_0}\,d_1\,\overline{d_2}\,\overline{d_3};$$

$$c_E = \overline{d_0}\, d_1\, (\overline{d_2}\, d_3 \vee d_2\, \overline{d_3}\, a_0 \vee d_2\, d_3\, a_9);$$

$$a_{D0} = d_0\, \overline{d_1}\, \overline{d_2}\, \overline{d_3};$$

$$a_{D1} = d_0\, \overline{d_1}\, \overline{d_2}\, d_3;$$

$$a_I = d_0\, \overline{d_1}\, d_2\, \overline{d_3};$$

$$z = d_0\, \overline{d_1}\, d_2\, d_3.$$

Ces différentes fonctions sont facilement réalisables au moyen de quatre relais : d_0, d_1, d_2 et d_3 (*fig. 26*). On remarque que les deux fonctions m_E et a_L sont identiques, de sorte que les deux relais correspondants peuvent être montés en parallèle, ou même réunis en un seul si le nombre de contacts disponibles le permet. Enfin, les contacts A_0 et A_9 seront commandés par les relais correspondants a_0 et a_9 de l'opérateur arithmétique.

AUTRE FORME DE DÉCODEUR. — Ici encore, comme dans le cas de l'additionneur-soustracteur de la *figure 14 bis*, l'emploi de redresseurs nous permet de donner à notre décodeur la forme plus rationnelle de la *figure 27*. Chacune des seize combinaisons possibles des 4 chiffres d_0, d_1, d_2 et d_3 de la lettre de fonction entraîne, par l'intermédiaire d'un sélecteur en pyramide, l'excitation de l'une des seize lignes verticales numérotées de 0 à 15. Il suffit donc, pour chacune de ces combinaisons, de mettre sous tension les lignes horizontales correspondant aux indications portées dans les colonnes du tableau récapitulatif des instructions. Ici encore, ces connexions doivent être faites au moyen de redresseurs n'autorisant le passage du courant que dans le sens allant des lignes verticales aux lignes horizontales.

On peut dire, dans ce cas, que l'opération se fait en réalité en deux temps :

a) *Décodage* de la lettre de fonction par le sélecteur qui excite la ligne verticale correspondante;

b) *Recodage* par la matrice de redresseurs qui excite la combinaison de relais correspondant à l'exécution de l'opération spécifiée par l'instruction en question.

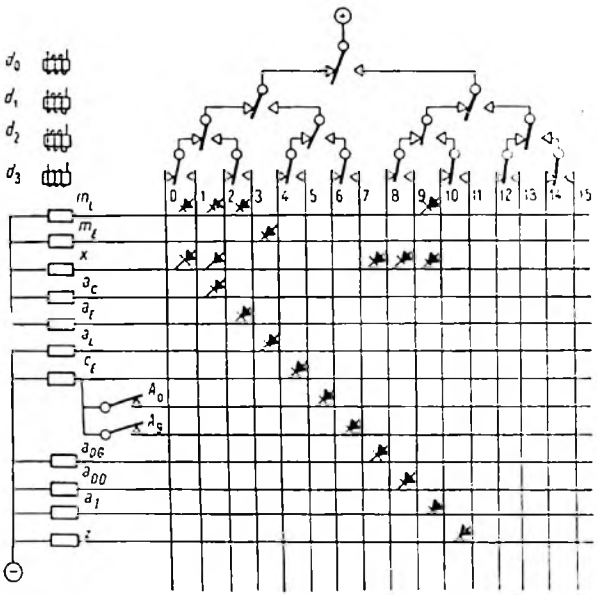


FIG. 27. — Décodeur à relais et redresseurs.

Compteur ordinal.

Le rôle du compteur ordinal est de compter le déroulement des instructions successives, afin d'indiquer, au cours de chaque phase instruction Φ_1, l'adresse de l'instruction qu'il convient d'extraire de la mémoire. Son contenu, exprimé en numération binaire par 6 chiffres (c_4 à c_0) doit donc progresser d'une unité au début de chaque phase nombre Φ_2.

Ce fonctionnement est schématisé sur la *figure 28*, dans laquelle le signal c_L est bien entendu le signal de synchronisation engendré par l'horloge.

Essayons tout d'abord de constituer un compteur binaire au moyen de notre cellule de mémoire élémentaire, qui n'est pas autre chose — nous l'avons signalé — que la version idéalisée du multivibrateur bistable ou bascule électronique. Nous prévoirons donc une cellule de mémoire par étage binaire. L'examen de la *figure 28* nous montre que la cellule c_9,

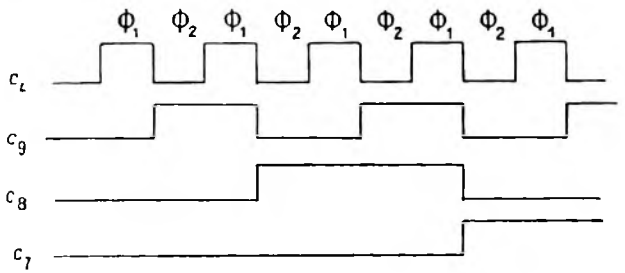


FIG. 28. — Diagramme de fonctionnement du compteur ordinal.

de plus petit poids, doit changer d'état, ou « basculer », lorsque le signal d'horloge c_L passe du niveau (+) au niveau (—), et non lorsqu'il effectue la transition inverse. Il nous faut donc un moyen pour discriminer les échelons négatifs de c_L des échelons positifs, et pour éliminer ces derniers.

La discrimination du signe sera assurée par un quadripôle différentiateur (*fig. 29 a*) constitué par une capacité en série et une résistance en parallèle. Ce circuit est ainsi appelé parce qu'il fournit à ses bornes de sortie un signal qui, au moins pour les variations assez rapides, représente approximativement la dérivée du signal d'entrée par rapport au temps. Fermons brusquement à l'instant $t = 0$ les bornes d'entrée sur une source de tension continue V_0. Pendant les premiers instants du phénomène transitoire qui suit la fermeture de l'interrupteur, la tension V_0 se retrouve entre

les bornes de sortie, car le condensateur n'a pas encore eu le temps de se charger. Puis le condensateur se charge exponentiellement à travers la résistance, et la tension de sortie tend asymptotiquement vers 0 selon une loi exponentielle de constante de temps RC (*fig. 29 b*) :

$$\frac{V}{V_0} = e^{-t/RC}.$$

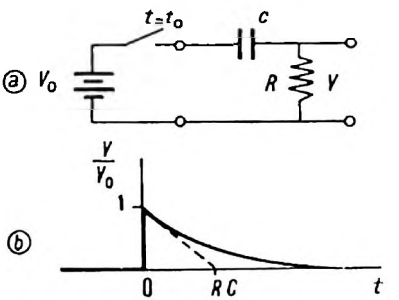


FIG. 29. — Obtention d'une impulsion exponentielle à partir d'un échelon

La constante de temps RC est encore égale à l'abscisse à l'origine de la tangente à la courbe à l'instant : $t = 0$. Précisons que ces résultats ne sont corrects que si aucun courant n'est prélevé sur les bornes de sortie, autrement dit si les organes éventuellement reliés à ces bornes présentent une impédance infinie, c'est-à-dire en pratique suffisamment élevée (grille du tube par exemple) ou si leur impédance interne est inclue dans R.

Appliquons maintenant, à l'entrée d'un tel quadripôle différenciateur, un signal en créneaux tel que c_L (*fig. 30*). Si la constante de temps RC est choisie notablement inférieure à la demi-période du signal d'entrée, le signal de sortie sera constitué par une suite d'impulsions exponentielles alternativement positives et négatives.

La première condition posée plus haut se trouve ainsi satisfaite. Nous devrons maintenant prévoir un dispositif,

schématisé sur la *figure 30* par deux redresseurs, interdisant aux impulsions positives d'atteindre les bornes d'entrée de notre cellule de mémoire. Au contraire, une impulsion

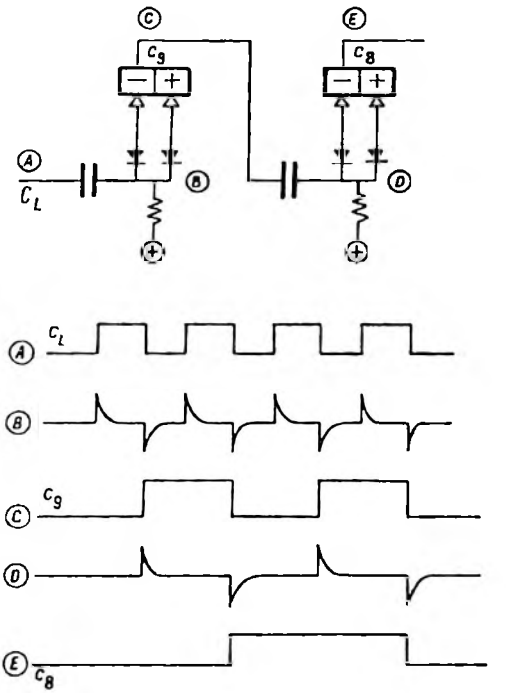


FIG. 30. — Compteur ordinal à « bascules ».

négative sera transmise à la borne de commande de la moitié (+) et entraînera donc le basculement de la cellule. Le signal en créneaux disponible sur la borne de sortie de la moitié gauche de c_9 sera de même différentié avant d'être appliqué à c_8, et ainsi de suite. Sur la *figure 30*, sont également représentés les signaux que l'on peut observer aux points *A*, *B*, *C*, *D* et *E*.

Il est également très instructif d'envisager la possibilité de construire un étage de compteur binaire au moyen d'organes logiques tels que des relais. Le problème se pose comme l'indique la *figure 31*. Il s'agit, étant donné un signal

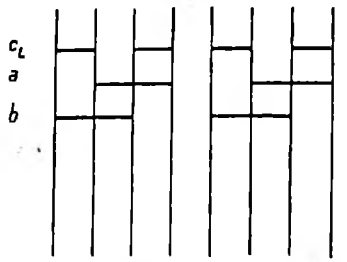


FIG. 31. — Diagramme de fonctionnement du compteur binaire à relais de la *figure* 33.

de commande c_L matérialisé par l'excitation d'un relais, de provoquer l'excitation d'un second relais a à cadence moitié. Le problème ne peut être résolu qu'au moyen d'un relais auxiliaire b dont le fonctionnement est décalé d'une phase par rapport à celui de a. Le cycle de fonctionnement comprend en effet quatre phases, qui ne peuvent être identifiées sans ambiguïté qu'au moyen de deux organes binaires. Notons en passant que le symbolisme de la *figure 31* est usuel pour représenter l'évolution des circuits à fonctionnement séquentiel (*schéma des phases*).

Chacun des relais a et b sera excité par des combinaisons appropriées de contacts C_L, A et B. L'examen de la *figure 31* nous montre que b doit être excité en même temps que c_L pourvu que a ne soit pas excité, mais que cette excitation doit ensuite se prolonger après cessation de celle de c_L jusqu'à ce que celui-ci soit excité à nouveau; on dit que l'excitation de b doit être « entretenue » après cessation de c_L jusqu'à sa réapparition. Ces propositions s'expriment

sous forme condensée au moyen de l'équation logique :

$$b = C_L \cdot \overline{A} \vee B \cdot \overline{C_L}.$$

La *figure 32 a* représente la transcription de cette équation au moyen d'un inverseur C_L, d'un contact « repos » A et d'un contact « travail » B. Ce circuit présente toutefois un grave inconvénient : l'alimentation de b se trouve interrompue pendant la durée du basculement de C_L, que notre

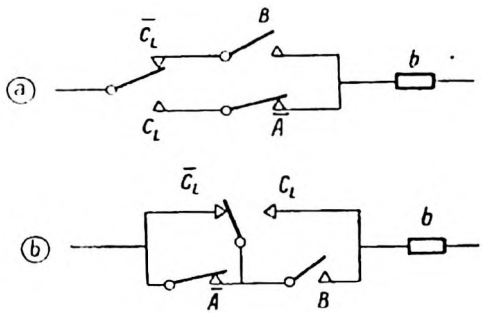


FIG. 32. — Circuit de commande du relais b (a) et circuit modifié (b) dans le but de supprimer la discontinuité qui se produit dans le circuit (a) au moment du basculement de l'inverseur C_L.

diagramme de fonctionnement suppose instantané. Il est heureusement facile de lever cette difficulté au moyen du circuit de la *figure 32 b*, dont le fonctionnement est identique à celui de la *figure 32 a*, comme il est facile de s'en rendre compte. Dans ce circuit, l'excitation de b et l'absence simultanée d'excitation de a assure la continuité de l'alimentation de b pendant le basculement de l'inverseur.

L'équivalence des deux circuits peut facilement être établie au moyen de l'algèbre de branchement. Le second circuit (*fig. 32 b*) est régi par l'équation :

$$b = (C_L \vee B) \cdot (\overline{C_L} \vee \overline{A}).$$

... le second membre sous la forme d'une réunion d'intersections :

$$b = C_L \cdot \overline{A} \vee B \cdot \overline{C_L} \vee \overline{A} \cdot B \vee C_L \cdot \overline{C_L}.$$

On remarque tout d'abord que : $C_L \cdot \overline{C_L} = 0$; ce terme peut donc être supprimé. En second lieu, l'égalité : $b = \overline{A} \cdot B$ est déjà contenue dans la réunion (*ou*) des deux premiers termes du second membre; en effet, il est évident que b est excité, quel que soit l'état de C_L, si l'on a simultanément : $B = 1$ et $\overline{A} = 1$, puisque, dans ces conditions, l'un de ces deux termes est toujours égal à 1. Cela se voit très clairement sur la *figure 32 b*. Le produit $\overline{A} \cdot B$ peut donc lui aussi être supprimé, et l'on retrouve ainsi la première expression.

Cette transformation est fréquemment utilisée dans les circuits à relais, afin d'assurer la *continuité* des circuits pendant le basculement des inverseurs.

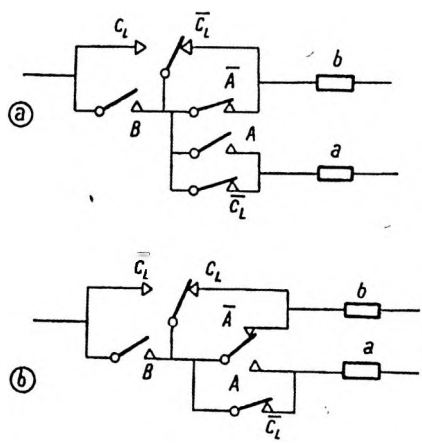


FIG. 33. — Étage binaire à relais.

Écrivons maintenant l'équation de fonctionnement relais a et appliquons-lui la même transformation :

$$a = B \cdot \overline{C_L} \vee A \cdot C_L = (C_L \vee B) \cdot (\overline{C_L} \vee A).$$

Nous obtenons ainsi le circuit complet de la *figure 33 a* qui se simplifie en celui de la *figure 33 b* par la réunion des deux contacts A et $\overline{A}$, qui ont un point commun, en un inverseur.

Schéma complet et organisation des phases.

Le schéma complet de la *planche I* (hors-texte sur dépliant) résulte de la représentation explicite de certains des organes schématisés sur la *figure 24*.

Le sélecteur de mémoire *SM*, le décodeur d'instructions *DI* et l'opérateur arithmétique *OA*, non explicités, sont constitués par des réseaux de contacts commandés par les divers

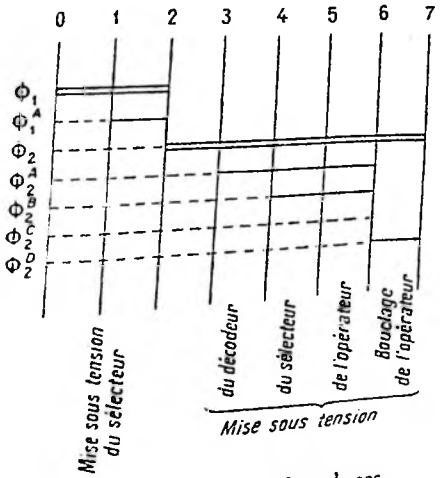


FIG. 34. — Organisation des phases.

relais figurant sur la *planche I* et représentés en détail sur les *figures 22, 26* et *14*. Afin d'éviter l'usure des contacts, il convient, autant que possible, de positionner ces contacts par excitation des relais correspondants avant de mettre sous tension les réseaux qu'ils constituent. Les pôles (+) des divers réseaux de contacts seront donc reliés à la source (+) dans un ordre déterminé et gouverné par l'horloge. Ces phases secondaires sont indiquées dans des triangles sur la *planche I* et leur enchaînement est défini sur la *figure 34*.

Justifions cette organisation en suivant le déroulement des opérations au cours d'un cycle de calcul constitué par une phase « instruction » Φ_1 et une phase « nombre » Φ_2.

Phase instruction Φ_1. — Au début de cette phase, les contacts C_L se ferment et les contacts $\overline{C_L}$ s'ouvrent. Le contenu du compteur ordinal *CO* est transféré par les contacts C_L dans la mémoire du sélecteur *MS*, qui excite ceux des relais m_i correspondant à l'adresse de l'ordre à exécuter. En même temps, le relais m_L est excité par l'horloge, de sorte que le contact M_L du sélecteur de mémoire *SM* se ferme, préparant la lecture de la mémoire sélectée. Lorsque tous ces contacts sont en place, le sélecteur est mis sous tension (phase Φ_1^4). A ce moment, le relais l_i de la mémoire sélectée se trouve excité à travers le contact M_L fermé; le contenu de la mémoire sélectée apparaît donc sur le jeu de barres, d'où il est transféré dans la mémoire d'instruction *MI* par les contacts C_L.

Phase nombre Φ_2. — Au début de cette phase, les contacts C_L s'ouvrent et les contacts $\overline{C_L}$ se ferment. De plus, le contenu du compteur ordinal est augmenté d'une unité en vue de la phase instruction suivante. La partie adresse de l'instruction courante est transcrite dans la mémoire du sélecteur *MS*, et le code d'opération dans la mémoire du décodeur *MD*. A ce moment, le décodeur d'instruction *DI* est mis sous tension (phase Φ_2^4), afin de provoquer l'excitation des relais de l'opérateur et de la mémoire correspondant à l'opération à exécuter. Si l'instruction commande une rupture de séquence, le transfert de la partie adresse de *MI* dans

le compteur ordinal *CO* se produit dès le début de cette phase. Le sélecteur de mémoire *SM* est alors à son tour mis sous tension (phase $\Phi\frac{6}{2}$), ce qui, pour certaines instructions, a pour effet de relier au jeu de barres les bornes d'écriture ou de lecture de l'une des mémoires; si les bornes de lectures sont reliées aux barres, les cellules dans l'état 1 de la mémoire sélectée entraînent l'excitation des relais *b* correspondants de l'opérateur arithmétique. Ce dernier peut maintenant être mis sous tension (phase $\Phi\frac{7}{2}$), de sorte que le résultat de l'opération effectuée vient s'inscrire dans le registre A'. A ce moment, l'alimentation du décodeur, du sélecteur et de l'opérateur peut être coupée. Il reste, dans une dernière phase $\Phi\frac{0}{2}$, à commander le transfert du contenu de A' dans A par bouclage de l'opérateur à travers les contacts X', dans la mesure toutefois où le relais x a été préalablement excité au cours du décodage de l'instruction. On voit que, pour pouvoir couper l'alimentation du décodeur avant le début de cette phase et d'éviter ainsi des manœuvres inutiles des contacts de l'opérateur après le bouclage, on a muni le relais x d'un enroulement d'entretien branché en parallèle avec l'enroulement de x'.

Le cycle de calcul complet comprend donc sept temps, dont deux pour la phase « instruction » et cinq pour la phase « nombre ». Ce type de fonctionnement séquentiel est caractéristique de toutes les calculatrices numériques automatiques, dont le fonctionnement ne peut donc être compris sans une étude approfondie de l'*enchaînement des phases.*

Préparation des programmes, ordinogrammes.

Il est temps maintenant d'entrer un peu dans le détail du travail de préparation des programmes. Nous verrons, en particulier, tout le parti que l'on peut tirer du calcul sur les instructions elles-mêmes, ainsi que de l'usage de sous-programmes se raccordant automatiquement au programme principal.

PREMIER EXEMPLE : CALCUL DU MODULE D'UN NOMBRE. — Soit à remplacer le contenu de la mémoire d'adresse 100

par sa valeur absolue. Il suffit, pour cela, de tester le signe de ce contenu au moyen d'une instruction *P* et de le changer par complémentation, s'il est reconnu négatif. Les cinq instructions suivantes réalisent cette opération. Leur numérotation est choisie arbitrairement. L'effet de chaque instruction est expliqué à droite :

11	*R* 100		(100) est transféré dans l'opérateur.
12	*P* 16	⟵	test du signe : si (100) $\geqslant$ 0, ne rien modifier et passer à la suite du calcul en sautant à l'instruction 16; si (100) $<$ 0, changer son signe en le soustrayant du contenu du registre de l'opérateur préalablement vidé (instructions 13, 14 et 15).
13	*R* 0		mise à zéro du registre de l'opérateur, la mémoire 0 étant supposée vide.
14	*S* 100		soustraction de (100) du registre de l'opérateur.
15	*T* 100		transfert du module ainsi calculé dans 100.
16	...	⟵	suite du calcul.

Pour l'exécution de ce programme, nous avons été obligés de vider le registre de l'opérateur en y transférant le contenu de la mémoire d'adresse 0 supposée contenir le nombre 0000000000 :

$$(0) = 0000000000.$$

Nous réserverons de même les quelques adresses suivantes pour inscrire d'une manière permanente quelques autres constantes dont nous aurons fréquemment besoin.

Il est commode d'indiquer par une flèche que l'instruction de rupture de séquence conditionnelle *P* 16 est susceptible de renvoyer à l'instruction 16.

Le déroulement de ce programme peut être schématisé sous la forme du diagramme de la *figure 35*, que l'on appelle un « ordinogramme ».

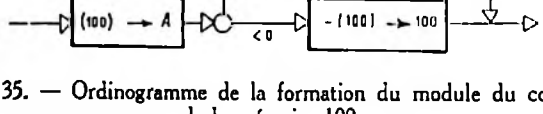


FIG. 35. — Ordinogramme de la formation du module du contenu de la mémoire 100.

DEUXIÈME EXEMPLE : TOTALISATION; PROGRESSION D'ADRESSE ET COMPTAGE. — Soit à calculer la somme des contenus de toutes les adresses comprises entre 100 et 200 inclusivement, le résultat devant être inscrit dans la mémoire d'adresse 100 :

$$(100) + (101) + (102) + \ldots + (200) \rightarrow 100.$$

Nous nous servirons également de la mémoire 100 pour y inscrire les sommes partielles.

Nous commencerons donc par les trois instructions :

R 100
A 101
T 100

Puis nous devrons exécuter les trois instructions :

R 100
A 102
T 100

Au lieu d'inscrire 100 fois ce groupe d'instructions, nous profiterons du fait que les instructions se présentent dans la mémoire sous la même forme que les nombres pour augmenter d'une unité la partie adresse de l'instruction *A*, supposée initialement égale à 100, à chaque pas du calcul de la somme.

A cet effet, nous inscrirons dans la mémoire d'adresse 1 une unité de plus petit poids :

$$(1) = 0000000001.$$

Nous saurons enfin que la sommation est terminée lorsque cette instruction sera devenue *A* 201, ce dont nous nous apercevrons en testant le signe de la différence *A* 200 — *A i*, *i* désignant l'adresse variable. Nous devrons donc également disposer dans une mémoire de l'équivalent binaire de (*A* 200). Un tel mot, qui se présente comme une instruction, et qui n'est pas destiné à être exécuté comme tel, mais à être utilisé comme une quantité numérique, est appelé une « pseudo-instruction ».

Dans le programme ci-dessous, expliqué en détail, l'instruction variable $A\ i$ est inscrite entre crochets, et la pseudo-instruction A 200 est précédée d'une double barre. Ces conventions sont empruntées à M. V. WILKES [1].

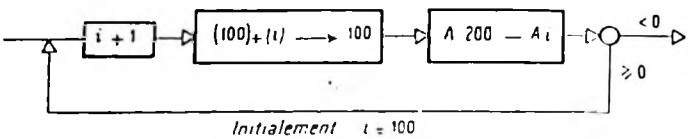


FIG. 36. — Ordinogramme de la formation de la somme des nombres d'adresses comprises entre 100 et 200.

21	R 25	progression d'une unité de l'adresse de l'instruction variable 25.
22	A 1	
23	T 25	
24	R 100	totalisation dans 100 : (100) + (i) → 100.
25	[A 100]	
26	T 100	
27	R 31	test de fin de sommation : — si : A 200 − $A\ i \geqslant 0$, ajouter le terme suivant; — si : A 200 − $A\ i < 0$, sauter à la suite du calcul par l'instruction N suivante.
28	S 25	
29	P 21	
30	N ...	sauter à la suite du calcul.
31	‖ A 200	pseudo-instruction servant au test de fin d'opération.

Le déroulement de ce programme est schématisé par l'ordinogramme de la *figure 36*.

(1) M. V. WILKES : *Automatic digital computers*, Methuen & Co. Ltd, London, 1956.

ADRESSES FLOTTANTES. — Dans le programme ci-dessus, trois des adresses figurant dans les instructions sont des adresses du programme lui-même. Il en résulte que si, dans l'établissement du programme, on a par exemple oublié une instruction, la numérotation des instructions se trouve modifiée, ainsi par conséquent que les instructions contenant des adresses du programme. Cet inconvénient se trouve éliminé si, dans l'établissement du programme, on se contente de désigner les adresses des instructions dont on a effectivement besoin par des lettres indexées, telles que n_1, n_2, n_3, etc... Après avoir vérifié que le programme ainsi établi ne comporte pas d'erreurs, on pourra numéroter les instructions successives et donner aux adresses flottantes leurs valeurs véritables :

Adresses flottantes				Adresses vraies	
n_3	$R\ n_1$	progression.		21	R 25
	A 1			22	A 1
	$T\ n_1$			23	T 25
	R 100	totalisation.	$n_1 = 25$	24	R 100
n_1	[A 100]		$n_2 = 31$	25	[A 100]
	T 100		$n_3 = 21$	26	T 100
	$R\ n_2$	test de fin.		27	R 31
	$S\ n_1$			28	S 25
	$P\ n_3$			29	P 21
	N ...			30	N ...
n_2	‖ A 200			31	‖ A 200

PARAMÈTRES. — La valeur initiale de l'instruction variable 25, ainsi que celle de la pseudo-instruction de référence 31, peuvent être mises en place juste avant l'utilisation de ce programme; elles constituent alors des *paramètres* et donnent au programme une valeur tout à fait générale en permettant de sommer un nombre quelconque de termes dont le premier se trouve placé à une adresse quelconque de la mémoire.

TROISIÈME EXEMPLE : SOMMATION DE MODULES; APPEL D'UN SOUS-PROGRAMME ET RACCORDEMENT. — Proposons-nous de calculer la somme des modules des nombres d'adresses comprises entre 101 et 200 inclusivement, en faisant la totalisation dans la mémoire 100, supposée initialement vide.

Nous utiliserons à cet effet un sous-programme de valeur absolue préparé à l'avance et organisé de telle manière qu'il renvoie automatiquement au programme principal après avoir été exécuté, et ceci quelle que soit l'adresse à laquelle on a quitté le programme principal en appelant le sous-programme en question.

Soit n l'adresse de la première instruction du sous-programme. Supposons que nous ayons besoin de faire appel à ce sous-programme — c'est-à-dire de calculer une valeur absolue — après l'instruction d'adresse $(m - 1)$ du programme principal. Les deux instructions suivantes seront ainsi libellées :

$$\begin{array}{ll} m & R\,m \\ m+1 & N\,n \end{array}$$

Nous voyons que la première de ces deux instructions a pour effet de se recopier elle-même dans le registre de l'opérateur, ce qui nous permet de garder trace de l'endroit où nous avons quitté le programme principal. La seconde provoque une rupture de séquence inconditionnelle avec transfert à la première instruction du sous-programme.

Pour pouvoir, après l'exécution du sous-programme, revenir à l'instruction $(m + 2)$ du programme principal, il faut que figure, à la fin du sous-programme, une instruction de rupture de séquence inconditionnelle, dite « *instruction de raccordement* », ainsi rédigée :

$$n + p \qquad N\,m + 2.$$

Cette instruction peut être fabriquée à partir de $R\,m$, qui se trouve actuellement dans le registre de l'opérateur. En effet, les équivalents numériques de R et de N sont respecti-

vement 3 et 5; leur différence 2 est l'équivalent numérique de S. On peut donc écrire

$$N = R + S.$$

Pour pouvoir transformer $(R\ m)$ en $(N\ m + 2)$, il faut donc lui ajouter l'équivalent numérique de $(S\ 2)$, que nous supposerons conservé dans la mémoire d'adresse 2 :

$$(2) = S\ 2 = 0010000010.$$

Il faudra donc que les toutes premières instructions du sous-programme effectuent cette opération et en transfèrent le résultat à la fin du sous-programme :

n	$A\ 2$	calcul et mise en place de l'instruction de
$n+1$	$T\ n+p$	raccordement.
$n+2$	.....	instructions du sous-programme.
...	.....	
...	.....	

$n + p$ $(N\ m + 2)$ devient l'instruction de raccordement.

Nous mettrons, comme ci-dessus, entre parenthèses le contenu d'une mémoire ainsi mise en place par l'exécution du programme lui-même.

Un sous-programme ainsi organisé est appelé « sous-programme fermé ».

Appliquons ces considérations à la construction d'un sous-programme fermé de valeur absolue. Toutefois, cette fois, le résultat sera conservé dans le registre de l'opérateur, ce qui économisera une instruction à la fois dans le sous-programme lui-même et dans le programme de sommation des modules qui en fera usage. Le sous-programme est écrit ci-dessous, à gauche en adresse flottante, à droite en adresse absolue; le nombre dont on cherche la valeur absolue est supposé placé dans la mémoire d'adresse n_2; de plus, la

mémoire 2 contient $S\ 2$, comme nous l'avons déjà dit plus haut :

	Adresses flottantes		Adresses absolues	
	$A\ 2$ $T\ n_1$	formation et placement de l'instruction de raccordement.	11 12	$A\ 2$ $T\ 17$
	$R\ n_2$ $P\ n_1$	test du signe de (n_2).	13 14	$R\ 18$ $P\ 17$
	$R\ 0$ $S\ n_2$	changement de signe éventuel.	15 16	$R\ 0$ $S\ 18$
n_1	( )	devient l'instruction de raccordement.	17	( )
n_2	( )	contient le nombre dont on cherche le module.	18	( )

Appliquons ce sous-programme au calcul de la somme :

$$|(101)| + (|102|) + \ldots + |(200)| \rightarrow 100.$$

	dresses flottantes		Adresses absolues	
	$R\ 0$ $T\ 100$	mise à zéro de 100.	21 22	$R\ 0$ $T\ 100$
n_4	$R\ n_1$ $A\ 1$ $T\ n_1$	progression de l'adresse de l'instruction n_1.	23 24 25	$R\ 26$ $A\ 1$ $T\ 26$
n_1	$[R\ 100]$ $T\ 18$	transfert du terme suivant dans la mémoire 18 du sous-programme de module.	26 27	$[R\ 100]$ $T\ 18$
n_3	$R\ n_3$ $N\ 11$	appel du sous-programme de module (ce dernier est placé dans le registre de l'opérateur).	28 29	$R\ 28$ $N\ 11$
	$A\ 100$ $T\ 100$	totalisation du module ainsi calculé avec la somme partielle (100).	30 31	$A\ 100$ $T\ 100$
	$R\ n_3$ $S\ n_1$ $P\ n_4$	test de fin d'opération par différence $R\ 200 - R\ i$; terminé quand elle devient négative.	32 33 34	$R\ 36$ $S\ 26$ $P\ 23$

$\frac{N \ldots}{\| \ R\ 200}$	sauter à la suite du programme. pseudo-instruction utilisée pour le test de fin de sommation.	35 36	$\frac{N \ldots}{\| \ R\ 200}$

Nous voyons qu'une fois le sous-programme établi et mis en mémoire, tout se passe pour le programmeur comme si le code avait été enrichi d'une instruction supplémentaire déclenchant le calcul de la valeur absolue.

Examinons à nouveau sur ce cas particulier comment se forme l'instruction de raccordement. L'instruction R 28, d'adresse 28, se recopie elle-même dans le registre de l'opérateur. L'instruction suivante N 11 appelle l'instruction 11, première instruction du sous-programme. Celle-ci, A 2, ajoute au contenu du registre de l'opérateur, R 28, le contenu de la mémoire 2, soit S 2 :

$$R\,28 + S\,2 = N\,30.$$

La seconde instruction du sous-programme, T 17, transfert N 30 dans la mémoire 17. A la fin du sous-programme, l'instruction N 30 nous ramène donc à l'instruction 30 du programme principal. A ce moment, le module du terme considéré se trouve donc dans le registre de l'opérateur, et il suffit donc de lui ajouter le contenu du registre 100, dans lequel se fait la totalisation.

QUATRIÈME EXEMPLE : TROUVER LE PLUS GRAND D'UNE SUITE DE NOMBRE. — Soit à trouver le plus grand des nombres d'adresses comprises entre 100 et 199 inclusivement.

On peut opérer systématiquement de la manière suivante : on commence par comparer le second nombre au premier; si le second est inférieur au premier, on répète l'opération avec le troisième et le premier, et ainsi de suite; supposons que le n^e nombre soit supérieur au premier. On recommence alors à partir du n^e nombre, c'est-à-dire que l'on commence par le comparer avec le $(n + 1)^e$, puis avec le $(n + 2)^e$, et ainsi de suite. Si, par exemple, on ne trouve pas dans la suite de nombre supérieur au n^e, c'est que ce dernier est le plus grand de tous les nombres de la suite.

Le programme sera donc bâti autour des deux instructions variables :

$$\begin{matrix} n_1 & [R\,i] \\ n_2 & [S\,j] \end{matrix} \qquad j \geqslant i \qquad \text{initialement : } i = j = 100,$$

l'indice j progressant d'une unité tant que (i) est supérieur ou égal à (j) et l'indice i sautant à la valeur j lorsque (j) devient supérieur à (i) (1). On s'apercevra que l'opération est terminée lorsque Sj sera devenu S 200.

Le processus est schématisé sur l'ordinogramme de la *figure 37*.

Le programme est écrit ci-dessous, en adresses flottantes et en adresses absolues, et commenté dans la suite. La mémoire n_3 contient la référence S 200 et la mémoire n_6 est réservée pour le résultat :

Adresses flottantes				Adresses absolues		
n_4	R	n_2	progression de l'indice j de n_2.	11	R	17
	A	1		12	A	1
	T	n_2		13	T	17
	S	n_3	test de fin : terminé quand $S\,j$ est devenu S 200.	14	S	28
	P	n_5		15	P	23
n_1	[R	100]	comparaison de (i) et (j); faire progresser j si $(i) \geqslant (j)$; repartir de (j) si $(j) < (i)$.	16	[R	100]
n_2	[S	100]		17	[S	100]
	P	n_4		18	P	11
	R	n_2	substitution de $R\,j$ à $R\,i$: $S\,j + A\,0 = R\,j$ [(5) = A 0].	19	R	17
	A	5		20	A	5
	T	n_1		21	T	16
	N	n_4		22	N	11
n_5	R	n_1	(i) est le plus grand nombre; transférer l'instruction $R\,i$ en n_7, afin de pouvoir transférer (i) dans n_6.	23	R	16
	T	n_7		24	T	25
n_7	(R	i)		25	(R	i)
	T	n_6		26	T	29

(1) (i) et (j) désignent comme d'ordinaire les contenus des mémoires d'adresses i et j.

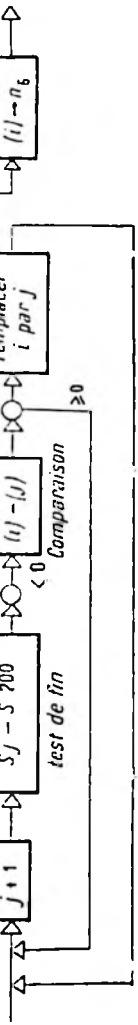


FIG. 37. — Ordinogramme de l'extraction du plus grand des nombres d'adresses comprises entre 100 et 200.

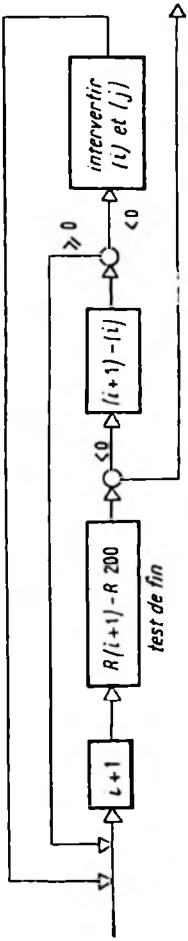


FIG. 38. — Ordinogramme du classement par valeurs croissantes des nombres d'adresses comprises entre 100 et 200.

	$N \ldots$	sauter à la suite du programme.	27	$N \ldots$
	$\parallel S$ 200	pseudo-instruction pour test de fin.	28	$\parallel S$ 200
n_6	( )	résultat.	29	( )

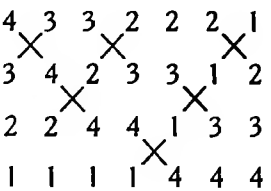
Le début de ce programme se comprend aisément. On fait progresser l'indice j d'unité en unité tant que $(i) \geqslant (j)$. Lorsque l'on rencontre un nombre (j) supérieur à (i), on repart de ce nombre, c'est-à-dire que l'on substitue Rj à Ri. Les équivalents numériques de S et R étant respectivement 2 et 3, il suffit pour cela d'ajouter à Si, contenu dans la mémoire n_2, la quantité A 0. Cette dernière, qui nous servira à nouveau dans la suite, est supposée inscrite dans la mémoire 5 :

$$(5) = A\ 0 = 0001000000.$$

L'adresse i de l'instruction n_1 telle qu'elle se présente lorsque tous les nombres ont été ainsi passés en revue est donc celle du plus grand nombre de la suite. Il convient maintenant de transférer ce nombre dans n_6. A cet effet, l'instruction Ri d'adresse n_1 est transférée à l'adresse n_7, afin d'être exécutée, c'est-à-dire d'amener le plus grand nombre (i) dans le registre de l'opérateur, d'où il est transféré dans la mémoire de résultat n_6.

CINQUIÈME EXEMPLE : CLASSEMENT D'UNE SUITE DE NOMBRES. — Soit à classer par ordre croissant les nombres d'adresses comprises entre 100 et 199 inclusivement. On peut y parvenir systématiquement de la manière suivante : on compare successivement les couples formés par deux nombres consécutifs de la suite; dès que l'on parvient à un nombre de rang $(i + 1)$ inférieur au nombre de rang i, on intervertit ces deux derniers nombres et on recommence en repartant du début de la suite. On parvient ainsi nécessairement à classer tous les nombres dans l'ordre

croissant. Montrons-le sur l'exemple de la suite 4, 3, 2, 1 :



Le programme sera donc bâti autour de la paire d'instructions variables :

n_1 $[R\,i + 1]$
n_2 $[S\,i]$ initialement : $i = 100$.

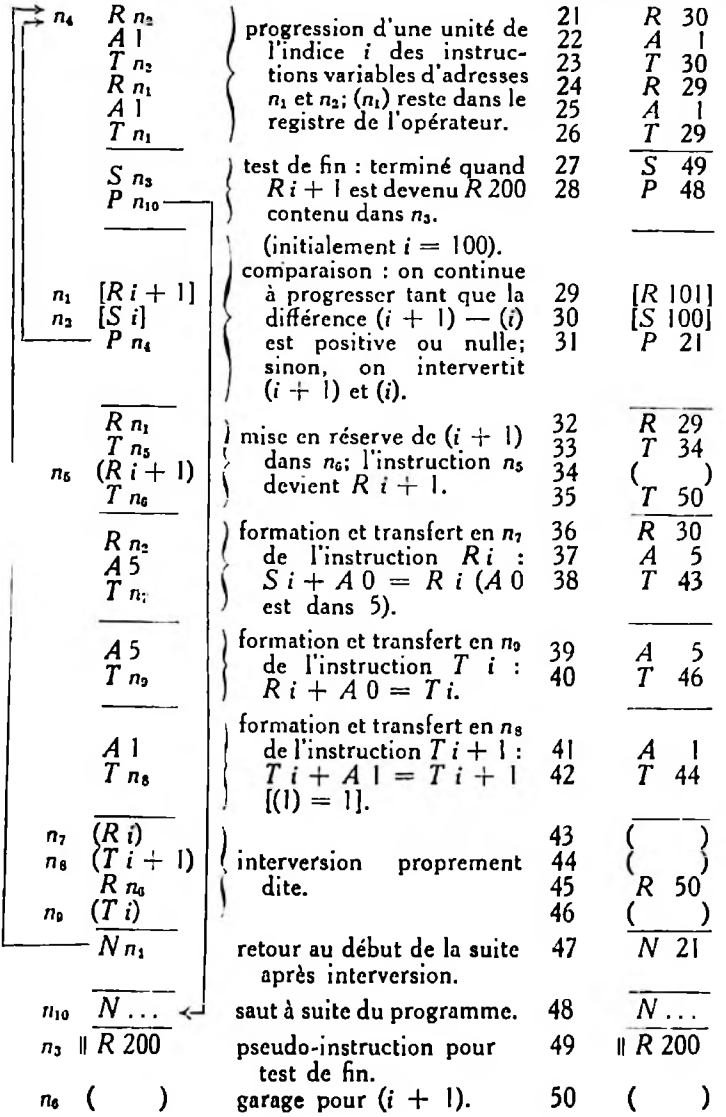
On aura terminé lorsque $R\,i + 1$ sera devenu $R\,200$, emmagasiné dans la mémoire d'adresse n_3.

Comme précédemment, nous expliquerons les points les plus délicats du programme écrit page suivante; l'ordinogramme est conforme au schéma de la *figure 38*.

On voit que l'interversion, nécessaire si $(i + 1)$ est inférieur à (i), est assurée par les quatre instructions 43 à 46, qui sont successivement fabriquées et mises en place par les instructions 36 à 42. Il est nécessaire, au préalable, de rendre disponible l'une des deux adresse i ou $i + s$ en mettant son contenu en réserve dans la mémoire 50; étant donné que l'instruction $R\,i + 1$ est déjà disponible en 29, c'est l'adresse $i + 1$ qui est ainsi libérée dans le programme écrit ci-dessus : il suffit pour cela de recopier cette instruction en 34.

SIXIÈME EXEMPLE : PRODUIT DE DEUX NOMBRES. — Les calculatrices numériques automatiques comportent toujours dans leur code une ou plusieurs instructions susceptibles de déclencher le calcul du produit de deux nombres par un organe spécial appelé multiplieur. Nous donnerons ultérieurement quelques explications à ce sujet.

Tenons-nous-en pour le moment à notre machine simplifiée et construisons, à titre d'exercice, un sous-programme


Adresses flottantes
Adresses absolues
n_4 $R\ n_2$
$A\ 1$
$T\ n_2$
$R\ n_1$
$A\ 1$
$T\ n_1$
progression d'une unité de l'indice i des instructions variables d'adresses n_1 et n_2; (n_1) reste dans le registre de l'opérateur.
21 R 30
22 A 1
23 T 30
24 R 29
25 A 1
26 T 29
$S\ n_3$
$P\ n_{10}$
test de fin : terminé quand $R\,i+1$ est devenu $R\,200$ contenu dans n_3.
27 S 49
28 P 48
(initialement $i = 100$).
n_1 $[R\,i+1]$
n_2 $[S\ i]$
$P\ n_4$
comparaison : on continue à progresser tant que la différence $(i+1) - (i)$ est positive ou nulle; sinon, on intervertit $(i+1)$ et (i).
29 $[R\ 101]$
30 $[S\ 100]$
31 P 21
$R\ n_1$
$T\ n_5$
n_5 $(R\,i+1)$
$T\ n_6$
mise en réserve de $(i+1)$ dans n_6; l'instruction n_5 devient $R\,i+1$.
32 R 29
33 T 34
34 ( )
35 T 50
$R\ n_2$
$A\ 5$
$T\ n_7$
formation et transfert en n_7 de l'instruction $R\,i$: $S\,i + A\,0 = R\,i$ ($A\,0$ est dans 5).
36 R 30
37 A 5
38 T 43
$A\ 5$
$T\ n_9$
formation et transfert en n_9 de l'instruction $T\ i$: $R\,i + A\,0 = T\,i$.
39 A 5
40 T 46
$A\ 1$
$T\ n_8$
formation et transfert en n_8 de l'instruction $T\,i+1$: $T\,i + A\,1 = T\,i+1$ $[(1) = 1]$.
41 A 1
42 T 44
n_7 $(R\,i)$
n_8 $(T\,i+1)$
$R\ n_6$
n_9 $(T\,i)$
interversion proprement dite.
43 ( )
44 ( )
45 R 50
46 ( )
$N\ n_1$
retour au début de la suite après interversion.
47 N 21
n_{10} $N\ \ldots$
saut à suite du programme.
48 $N\ \ldots$
n_3 $\|\ R\ 200$
pseudo-instruction pour test de fin.
49 $\|\ R\ 200$
n_6 ( )
garage pour $(i+1)$.
50 ( )

de multiplication. Nous commencerons par le produit de deux nombres positifs, puis nous indiquerons deux méthodes pour la multiplication de deux nombres de signes quelconques.

Produit de deux nombres positifs. — Le produit de deux nombres positifs se forme à la manière habituelle par addition des produits partiels. Mais, ici, la chose est particulièrement simple, puisque le produit de chaque chiffre du multiplicateur par le multiplicande ne peut être que 0 ou le multiplicande lui-même.

Ainsi :

$$\begin{array}{r} 0{,}1\,1\,0\,1 \\ \times\ 0{,}1\,0\,1\,1 \\ \hline 1\,1\,0\,1 \\ 1\,1\,0\,1 \\ 0\,0\,0\,0 \\ 1\,1\,0\,1 \\ \hline 0{,}1\,0\,0\,0\,1\,1\,1\,1 \end{array}$$

On testera donc les chiffres successifs du multiplicateur, en commençant par le chiffre de plus faible poids; si le chiffre de rang n est un « 1 », le produit partiel correspondant est constitué par le multiplicande décalé de n rangs vers la gauche.

Pratiquement, la totalisation des produits partiels se fera dans le registre A de l'opérateur arithmétique. On décalera donc les sommes partielles d'un rang vers la droite avant chaque addition d'un nouveau produit partiel, en abandonnant d'ailleurs chaque fois le chiffre de plus petit poids, qui sort de la capacité de l'opérateur :

Premier produit partiel	1 1 0 1
Décalage à droite	1 1 0
Deuxième produit partiel	+ 1 1 0 1
Première somme partielle	1 0 0 1 1
Décalage à droite	1 0 0 1
Troisième produit partiel	+ 0 0 0 0
Deuxième somme partielle	1 0 0 1
Décalage à droite	1 0 0
Quatrième produit partiel	+ 1 1 0 1
Troisième somme partielle	1 0 0 0 1
Décalage à droite	1 0 0 0

Nous retrouvons évidemment le produit arrondi aux quatre premiers rangs binaires :

0,1 0 0 0

Enfin, pour tester les chiffres successifs du multiplicateur m, on décalera ce dernier d'un rang vers la droite après chaque addition de produit partiel, et on testera toujours le chiffre de plus petit poids m_9, au moyen de l'instruction Q du code, qui a été introduite précisément à cet effet.

Le cycle élémentaire de calcul décrit ci-dessus (test du multiplicateur, addition du produit partiel et décalage de la somme partielle) doit être répété neuf fois. Ce comptage peut ici se faire d'une manière particulièrement simple en introduisant initialement dans une mémoire réservée à cet effet le mot 0111111111 représenté par la pseudo-instruction Q 63; si ce mot est décalé d'un rang vers la droite après chaque cycle, son chiffre de plus petit poids deviendra 0 après le neuvième pas, ce dont on s'apercevra facilement au moyen d'une instruction Q.

Un sous-programme de multiplication de deux nombres positifs, résumé sur l'ordinogramme de la *figure 39*, est écrit et commenté ci-après. Le multiplicande M et le multiplicateur m ont été introduits préalablement dans les mémoires n_1 et n_2. Les produits partiels sont totalisés dans la mémoire n_3. La mémoire n_4 sert de compteur; son contenu initial Q 63 est inscrit en permanence dans n_9.

Examinons l'effet de l'instruction de test de m_9 : Q n_7, d'adresse n_8. Si $m_9 = 0$, elle déclenche la suite des instructions :

$$\left.\begin{array}{l} R\,0 \\ A\,n_3 \\ D \\ T\,n_3 \end{array}\right\} 0 + (n_3) \rightarrow A$$

Il a pour effet de vider le registre de l'opérateur, d'y introduire la somme partielle précédente et de la décaler d'un rang vers la droite.

Adresses flottantes			Adresses absolues		
	A 2 T n_5	formation et placement de l'instruction de raccordement.	21 22	A 2 T 41
	R n_8 T n_1	réglage initial du compteur n_4 à la valeur 0111111111 contenue dans n_9.	23 24	R 44 T 45
	R n_2	test de m_9 : si 1, ajouter M au produit partiel et décaler; si 0,	25	R 43
n_6	Q n_7	décaler seulement produit partiel.	26	Q 30
	R n_1		27	R 42
	N n_9		28	N 30
n_7	R 0	totalisation du produit partiel.	29	R 0
n_6	A n_3		30	A 44
	D		31	D
	T n_3	décalage de la somme partielle.	32	T 44
	R n_1		33	R 45
	D	décalage du compteur n_4 et test	34	D
	T n_1	de fin.	35	T 45
	Q n_5		36	Q 41
	R n_2		37	R 43
	D		38	D
	T n_2	décalage de m.	39	T 43
	N n_6		40	N 26
n_5	( )	devient l'instruction de raccordement.	41	( )
n_1	(M)	multiplicande.	42	(M)
n_2	(m)	multiplicateur.	43	(m)
n_3	(sp)	sommes partielles.	44	(sp)
n_4	(CTR)	compteur.	45	(CTR)
n_9	$\|Q$ 63	pseudo-instruction : valeur initiale du compteur.	46	$\|Q$ 63

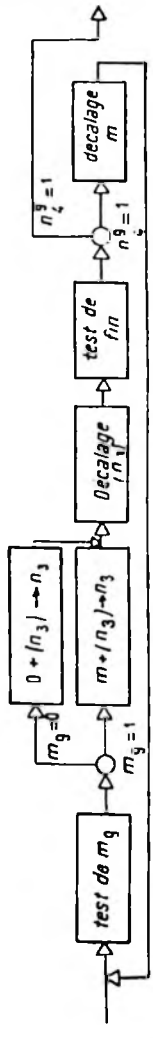


FIG. 39. — Ordinogramme du produit de deux nombres positifs.

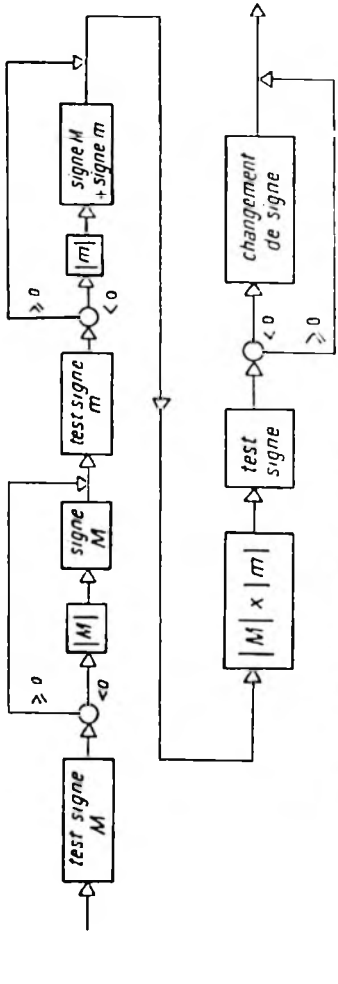


FIG. 40. — Ordinogramme du produit de deux nombres de signes quelconques par produit des valeurs absolues et combinaison des signes.

Si $m_9 = 1$, les instructions suivantes sont exécutées :

$$\left.\begin{matrix} R\ n_1 \\ A\ n_3 \\ D \\ T\ n_3 \end{matrix}\right\} (n_1) + (n_3) \rightarrow A.$$

M est alors ajouté à la somme partielle précédente, puis nouvelle somme est décalée.

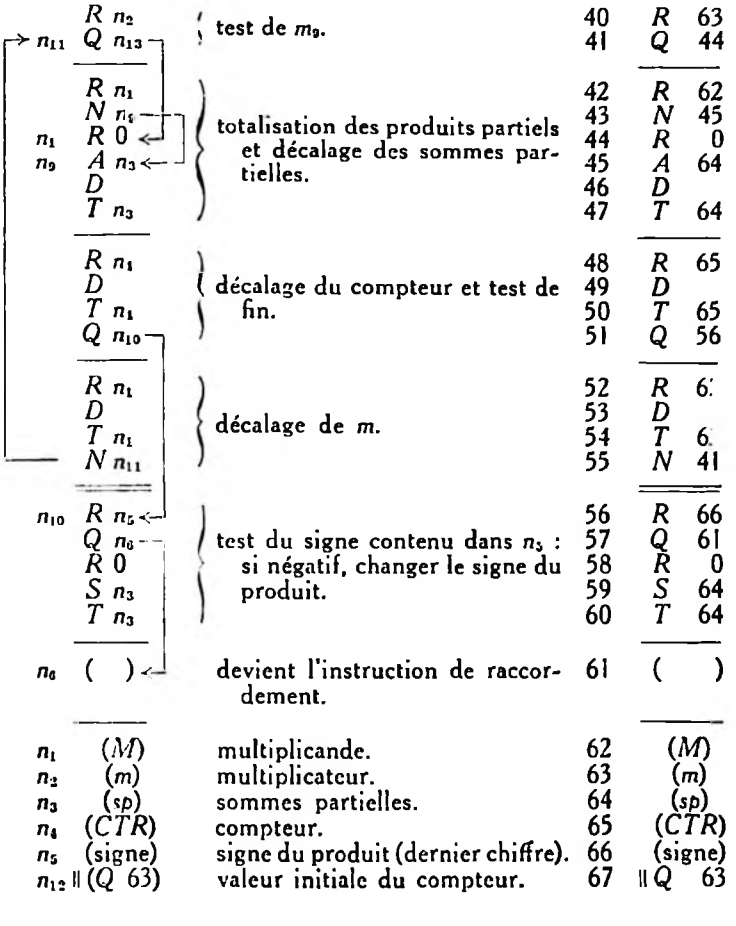
Première méthode pour le produit de deux nombres de signes quelconques. — Il suffit de faire le produit des valeurs absolues comme ci-dessus et de combiner les signes selon la règle habituelle. Si le produit est négatif, on changera le signe du produit des modules en les soustrayant après avoir vidé le registre de l'opérateur.

Or, la règle de combinaison des signes s'exprime par l'addition modulo 2 ou dilemme. Nous opérerons donc de la manière suivante : si M est négatif, nous inscrirons une unité de plus petit poids (contenu de la mémoire 1) dans une mémoire n_5 réservée à cet effet; si m est également négatif, nous ajouterons une unité de plus petit poids au contenu de n_5. Le chiffre de plus petit poids du contenu final de n_5 indiquera alors le signe du produit. Il suffira de le tester par une instruction Q pour savoir si on doit ou non changer le signe du produit des modules. Le rôle des mémoires n_1, n_2, n_3, n_4 est le même que précédemment; le contenu initial du compteur n_4 est inscrit dans n_{12}.

Le programme, écrit pages suivantes, est résumé par l'ordinogramme de la *figure 40*.

Deuxième méthode pour le produit de deux nombres de signes quelconques. — On peut aussi effectuer le produit de deux nombres de signes quelconques en faisant le produit brut de leurs représentations dans le code des compléments et en le corrigeant convenablement pour obtenir le résultat correct. Commençons par établir la théorie de cette méthode; après quoi, nous écrirons le programme correspondant.

Adresses flottantes			Adresses absolues		
	A 2	formation et placement de l'instruction de raccordement.	21	*A*	2
	T n_6		22	*T*	60
	R n_1		23	*R*	62
	P n_7		24	*P*	30
	R 0		25	*R*	0
	S n_1	test du signe de *M* : si négatif, changer de signe et inscrire (1) dans n_5.	26	*S*	62
	T n_1		27	*T*	62
	R 1		28	*R*	1
	T n_5		29	*T*	65
n_7	*R* n_2 ←		30	*R*	63
	P n_8		31	*P*	38
	R 0		32	*R*	0
	S n_2	test du signe de *m* : si négatif, changer de signe et ajouter (1) à (n_3).	33	*S*	63
	T n_2		34	*T*	63
	R n_5		35	*R*	65
	A 1		36	*A*	1
	T n_5		37	*T*	65
n_8	*R* n_{12} ←	réglage initial du compteur n_4.	38	*R*	66
	T n_4		39	*T*	64


R n2
Q n13
n11
test de m9.
40 R 63
41 Q 44
R n1
N n5
n1 R 0
n9 A n3
D
T n3
totalisation des produits partiels et décalage des sommes partielles.
42 R 62
43 N 45
44 R 0
45 A 64
46 D
47 T 64
R n1
D
T n1
Q n10
décalage du compteur et test de fin.
48 R 65
49 D
50 T 65
51 Q 56
R n1
D
T n1
N n11
décalage de m.
52 R 6
53 D
54 T 6
55 N 41
n10 R n5
Q n6
R 0
S n3
T n3
test du signe contenu dans n5 : si négatif, changer le signe du produit.
56 R 66
57 Q 61
58 R 0
59 S 64
60 T 64
n6 ( )
devient l'instruction de raccordement.
61 ( )
n1 (M) multiplicande. 62 (M)
n2 (m) multiplicateur. 63 (m)
n3 (sp) sommes partielles. 64 (sp)
n4 (CTR) compteur. 65 (CTR)
n5 (signe) signe du produit (dernier chiffre). 66 (signe)
n12 ‖ (Q 63) valeur initiale du compteur. 67 ‖ Q 63

Produit de deux nombres de signes contraires : considérons le produit d'un nombre positif a et d'un nombre négatif b, qui sera donc représenté par son complément vrai $(2 - b)$:

$$a \cdot (2 - b) = - a \cdot b + 2\,a$$

soit :
$$- a \cdot b = a \cdot (2 - b) - 2\,a.$$

Il faut donc corriger la valeur du produit brut obtenu comme ci-dessus en lui retranchant le double du facteur positif.

Exemple : Soit à effectuer le produit des équivalents binaires de 5/8 et — 7/8 :

$$- a \cdot b = 0{,}101 \times (- 0{,}111).$$

Le facteur négatif est représenté par son complément vrai 1,001, et le produit se présente donc sous la forme :

$$a\,(2 - b) = 0{,}101 \times 1{,}001$$

soit :

$$\begin{array}{r} 0,1\,0\,1 \\ \times\ 1,0\,0\,1 \\ \hline 1\,0\,1 \\ 1\,0\,1\,0\,0 \\ \hline 0,1\,0\,1\,1\,0\,1 \end{array}$$

Le double du facteur positif a s'obtient par décalage d'un rang vers la gauche, soit 1,01. Pour le retrancher du produit brut, nous lui ajouterons son complément, soit 0,11 :

$$\begin{array}{r} 0,1\,0\,1\,1\,0\,1 \\ +\ 0,1\,1 \\ \hline 1,0\,1\,1\,1\,0\,1 \end{array}$$

Ce dernier résultat représente un nombre négatif, dont la valeur absolue s'obtient par complémentation : 0,1 0 0 0 1 1, équivalent binaire de 35/64, ce qui est bien le résultat correct.

Produit de deux nombres négatifs : prenons maintenant le produit de deux nombres négatifs représentés tous deux par leurs compléments $(2 - a)$ et $(2 - b)$:

$$(2 - a) \cdot (2 - b) = a \cdot b + 2\,(2 - a) + 2\,(2 - b)$$

d'où :

$$a \cdot b = (2 - a)(2 - b) - 2(2 - a) - 2(2 - b).$$

Il faut donc, cette fois, corriger le produit brut en lui retranchant le double des compléments de chacun des deux nombres, c'est-à-dire le double des représentations de chacun des deux facteurs dans la machine.

Exemple : Considérons le produit des équivalents binaires de (— 7/8) et (— 5/8) :

$$a \cdot b = (-0{,}111) \times (-0{,}101).$$

Ce produit se présente dans la machine sous la forme :

$$(2 - a) \cdot (2 - b) = 1{,}011 \times 1{,}001$$

soit :

$$\begin{array}{r} 1,0\,1\,1 \\ \times\ 1,0\,0\,1 \\ \hline 1\,0\,1\,1 \\ 1\,0\,1\,1\,0\,0 \\ \hline 1,1\,0\,0\,0\,1\,1 \end{array}$$

Faisons la somme de nos deux nombres tels qu'ils se présentent dans la machine :

$$\begin{array}{r} 1,0\,0\,1 \\ +\ 1,0\,1\,1 \\ \hline \textcircled{1}\ 0,1\,0\,0 \end{array}$$

soit, modulo 2 : 0,1 ; le double de ce nombre est 1 ; pour le retrancher, nous ajouterons son complément, qui est également 1 :

$$\begin{array}{r} 1,1\,0\,0\,0\,1\,1 \\ +\ 1 \\ \hline \textcircled{1}\ 0,1\,0\,0\,0\,1\,1 \end{array}$$

Le second rang à gauche de la virgule étant ignoré de la

machine, ce nombre est bien l'équivalent binaire du produit cherché 35/64.

Nous aurions évidemment obtenu le même résultat en faisant directement la somme des compléments des représentations de nos deux nombres :

$$\begin{array}{r} 0{,}1\,1\,1 \\ +\ 0{,}1\,0\,1 \\ \hline 1{,}1\,0\,0 \end{array}$$

dont le double est 11, soit 1 modulo 2.

Règle générale pour le produit de deux nombres de signes quelconques : Soient x et y les représentations dans la machine de deux nombres dont on désire effectuer le produit. On commencera par effectuer le produit brut. Puis on testera le signe de x : s'il est négatif, on retranchera du produit brut le double modulo 2 de y. On testera ensuite le signe de y : s'il est négatif, on retranchera encore le double modulo 2 de x du résultat obtenu précédemment.

Programme correspondant à la règle ci-dessus : Il est facile d'écrire le programme correspondant à la règle énoncée ci-dessus conforme à l'ordinogramme de la *figure 41*. Il faut seulement remarquer que l'on aura besoin du multiplicateur m pour effectuer les corrections; comme il est décalé d'un rang vers la droite à chaque pas du produit, il faut donc commencer par le recopier dans la mémoire n_5 réservée à cet effet. On notera également qu'il faut, cette fois, répéter le cycle de calcul 10 fois au lieu de 9 comme précédemment. Le contenu initial du compteur n_1 devra donc être 1111111111; nous introduirons ultérieurement quatre lettres de fonction supplémentaires, dont E d'équivalent binaire 1111; le mot précédent peut donc être représenté par la pseudo-instruction E 63, que nous supposerons écrite dans la mémoire n_{11}. Enfin, le doublage de M et m, nécessaire dans les corrections, sera assuré par une instruction G de décalage d'un rang binaire à gauche.

On voit que ce programme est un peu plus court que le précédent par produit des modules et combinaison des signes (43 instructions au lieu de 46).

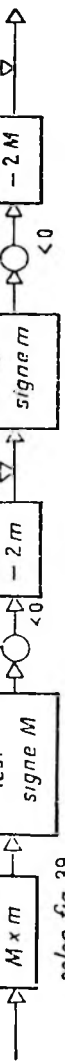


FIG. 41. — Ordinogramme du produit de deux nombres de signes quelconques par produit brut et corrections.

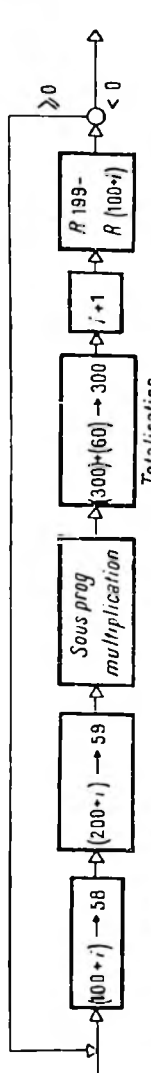


FIG. 42. — Ordinogramme du produit scalaire des deux vecteurs à 100 dimensions dont les coordonnées sont les nombres d'adresses comprises entre 100 et 199, d'une part, et entre 200 et 299, d'autre part.

Adresses flottantes			Adresses absolues		
	$A\ 2$	formation et placement de l'instruction de raccordement.	21	A	2
	$T\ n_6$		22	T	56
	$R\ n_2$	recopie de m dans n_5.	23	R	58
	$T\ n_5$		24	T	61
	$R\ n_{11}$	réglage initial du compteur n_4.	25	R	62
	$T\ n_4$		26	T	60
	$R\ n_2$	test de m_9.	27	R	58
$\rightarrow n_9$	$Q\ n_{12}$		28	Q	32
	$R\ n_1$	totalisation des produits partiels et décalage des sommes partielles.	29	R	58
	$N\ n_7$		30	N	60
n_{12}	$R\ 0$		31	R	0
n_7	$A\ n_3$		32	A	60
	D		33	D	
	$T\ n_3$		34	T	60
	$R\ n_4$	décalage du compteur et test de fin.	35	R	61
	D		36	D	
	$T\ n_4$		37	T	61
	$Q\ n_8$		38	Q	43
	$R\ n_2$	décalage de m.	39	R	59
	D		40	D	0
	$T\ n_2$		41	T	59
	$N\ n_9$		42	N	29
n_8	$R\ n_1$	test du signe de M.	43	R	58
	$P\ n_{10}$		44	P	50
	$R\ 0$	si M négatif, retrancher 2 m du produit brut.	45	R	
	$S\ n_2$		46	S	59
	G		47	G	
	$A\ n_3$		48	A	60
	$T\ n_3$		49	T	60
n_{10}	$R\ n_2$	test du signe de m.	50	R	59
	$P\ n_6$		51	P	57
	$R\ 0$	si m négatif, retrancher 2 M de (n_3).	52	R	0
	$S\ n_1$		53	S	58
	G		54	G	
	$A\ n_3$		55	A	60
	$T\ n_3$		56	T	60
n_6	$(\ )$	devient l'instruction de raccordement.	57	$(\ )$	

n_1	(M)	multiplicande.	58	(M)
n_2	(m)	multiplicateur.	59	(m)
n_3	(sp)	sommes partielles et résultat.	60	(sp)
n_4	(CTR)	compteur.	61	(CTR)
n_5	(m)	multiplicateur recopié ici.	62	(m)
n_{11}	$\parallel E$ 63	valeur initiale du compteur n_4.	63	$\parallel E$ 63

SEPTIÈME EXEMPLE : PRODUIT SCALAIRE. — Comme dernier exemple, appliquons le sous-programme de multiplication que nous venons d'écrire au calcul de la somme des produits

Adresses flottantes			Adresses absolues	
	R 0 T 300	mise à 0 de 300 (totalisation).	71 72	R 0 T 300
→ n_1	[R 100]		73	[R 100]
	T 58		74	T 58
n_2	[R 200]	mise en place des deux facteurs.	75	[R 200]
	T 59		76	T 59
n_3	R n_3 N 21	appel du sous-programme de multiplication.	77 78	R 77 N 21
	R 300		79	R 300
	A 60	totalisation des produits.	80	A 60
	T 300		81	T 300
	R n_1		82	R 73
	A 1	progression de l'adresse de n_1.	83	A 1
	T n_1		84	T 73
	R n_2		85	R 75]
	A 1	progression de l'adresse de n_2.	86	A 1
	T n_2		87	T 75
	R n_4		88	R 92
	S n_1	test de fin : terminé quand (n_1)	89	S 73
	P n_1	est devenu R 200.	90	P 73
	N ...	saut à la suite du calcul.	91	N ...
n_4	$\parallel R$ 199	pseudo-instruction de référence pour test de fin.	92	$\parallel R$ 199

deux à deux des nombres d'adresses comprises entre 100 et 199 inclusivement par les nombres d'adresses comprises entre 200 et 299 inclusivement

$$\sum_{i=0}^{i=99} (100 + i)(200 + i).$$

Cette somme constitue le produit scalaire, dans un espace à 100 dimensions, du vecteur de coordonnées $(100 + i)$ par le vecteur de coordonnées $(200 + i)$.

Le programme, écrit ci-dessus conformément à l'ordinogramme de la *figure 42*, se passe de commentaire. La totalisation se fait dans la mémoire 300.

Organisation d'un multiplieur.

Nous avons déjà signalé que les calculatrices numériques automatiques comportent toujours dans leur code les instructions nécessaires au déclenchement automatique de l'opération de multiplication. Notre code étant un code à simple adresse, l'un des facteurs du produit doit donc être indiqué implicitement. A cet effet, il sera introduit préalablement dans un registre spécial dit registre du multiplieur, que nous désignerons par L; ce transfert sera commandé par une instruction de la forme :

$L\ n$: inscrire dans le registre du multiplieur le contenu de la mémoire d'adresse n.

Le contenu de ce registre pourra alors être multiplié par celui d'une mémoire d'adresse quelconque n', au moyen d'une instruction telle que :

$M\ n'$: multiplier le contenu du registre du multiplieur par celui de la mémoire d'adresse n', et laisser le produit dans le registre A de l'opérateur.

Dans le cas de notre machine, nous pourrons donner à L et M les équivalents binaires 1100 et 1101, qui sont encore disponibles.

La *figure 43 a* représente l'organisation générale d'un multiplieur conçu selon ces principes. La totalisation des produits partiels se fait dans le registre de l'opérateur. Le registre L peut être décalé d'un rang vers la droite par l'excitation du relais d. De plus, la moitié gauche de sa cellule de plus petit poids excite ou non le relais t selon que le chiffre qu'elle contient est un 1 ou un 0. Les contacts T du relais t commandent, lorsque ce dernier est excité, le transfert dans l'opérateur du contenu de la mémoire n' spécifiée dans l'instruction $M\ n'$ et sélectée en conséquence par le sélecteur. Afin de ne pas troubler le fonctionnement normal de l'opérateur, ces contacts sont normalement court-

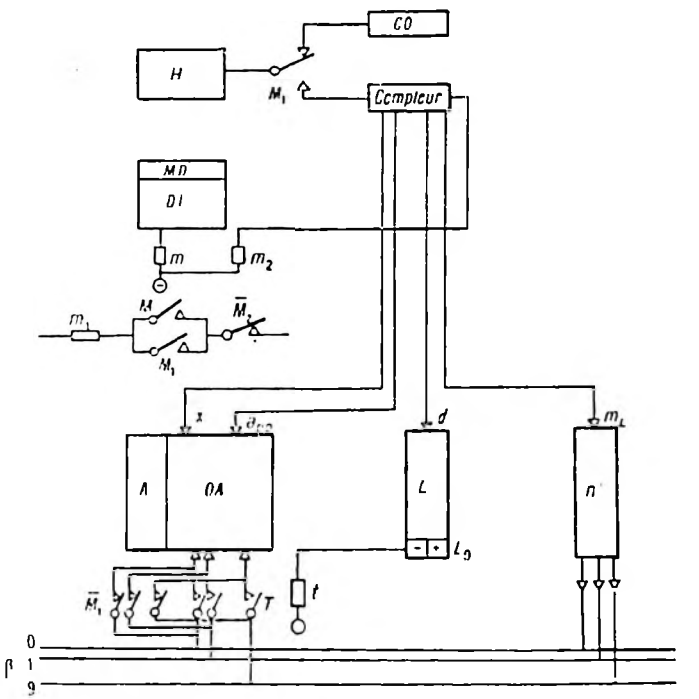


FIG. 43 *a*. — Organisation schématique d'un multiplieur.

circuités par les contacts M_1 ouverts seulement par l'excitation du relais m_1 sous l'effet du code 1101 (*voir détail plus loin*).

L'exécution de l'instruction $M\,n'$ comprend donc la répétition 10 fois de suite du cycle élémentaire suivant :

a) Addition du contenu de n' à celui de A, supposé vide au début du premier cycle; à cet effet, les relais m_L (lecture de la mémoire) et x (bouclage de l'opérateur) doivent être excités; cependant cette opération n'a lieu que si le chiffre de plus petit poids du second facteur contenu dans L est un 1; sinon, le contenu de A demeure inchangé;

b) Décalage d'un rang vers la droite des contenus de A et de L, par excitation des relais a_{DD} (décalage à droite), x (bouclage de l'opérateur) et d (décalage de L).

De plus, pendant toute la durée de l'opération, la liaison entre l'horloge et le compteur ordinal devra être interrompue. Un compteur actionné par l'horloge contrôlera la succession des pas et rétablira le fonctionnement normal de la machine après 10 répétitions. Ces fonctions sont schématisées sur la *figure 43 a*. L'excitation du relais m du décodeur par la lettre de fonction M provoque celle du relais m_1, qui possède un contact d'entretien M_1', ainsi que le basculement de l'inverseur M_1, dont le contact « repos » est relié au compteur ordinal et le contact « travail » au compteur qui commande et contrôle le déroulement du cycle décrit plus haut. L'excitation de m_1 provoque aussi l'ouverture des contacts $\overline{M_1}$ qui court-circuitent les contacts T. En fin d'opération, le compteur excite le relais m_2, qui ouvre le circuit de m_1 et remet les choses en état normal.

Du point de vue technologique, la fonction de comptage et de commande de la séquence d'opérations peut être assurée par un ou plusieurs pas à pas téléphoniques.

Remarquons qu'il est possible de ne pas perdre la moitié de plus petit poids du produit, dont un chiffre sort de la capacité de l'opérateur à chaque décalage à droite de la somme partielle. On peut utiliser à cet effet le registre L du multiplicateur, dont les cellules sont successivement

rendues disponibles, en partant de la cellule de plus fort poids, à raison d'une position nouvelle à chaque décalage à droite du multiplicateur. On peut donc faire rentrer successivement les chiffres de plus petits poids du produit dans le registre L, par son extrémité de plus fort poids, au fur et

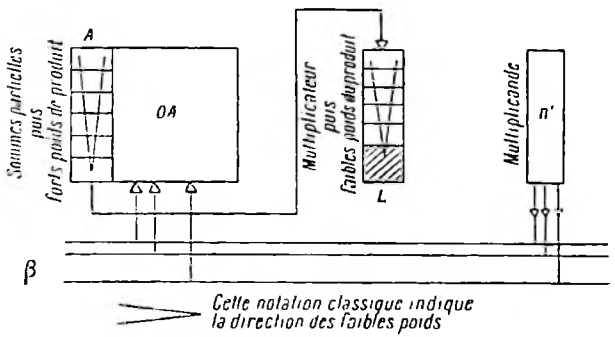


FIG. 43 *b*. — Multiplieur permettant de conserver tous les chiffres du produit.

à mesure que les chiffres du multiplicateur en sortent par son extrémité de plus faible poids (*fig. 43 b*).

Au demeurant, de nombreuses machines possèdent un totalisateur de capacité double, dans le but de réduire les erreurs qui ne manquent pas de s'accumuler lors de la formation d'une somme de produit.

Division.

Si toutes les calculatrices numériques automatiques sont dotées d'un opérateur arithmétique capable d'effectuer les opérations d'addition, de soustraction et de multiplication, un grand nombre d'entre elles ne possèdent pas, dans leur code, d'instructions relatives à la division. Celle-ci doit alors être programmée.

On peut tout d'abord utiliser une variante de la division par soustractions successives, mais sans retour en arrière, selon le procédé ci-dessous, valable quels que soient les

signes du dividende et du diviseur, à condition de représenter les nombres négatifs par leurs compléments vrais :

Si le dividende est nul ou du même signe que le diviseur, lui soustraire le diviseur; sinon, lui ajouter le diviseur. Si le résultat de cette opération d'addition ou de soustraction est nul ou du même signe que x, prendre 1 comme chiffre correspondant du quotient; dans le cas contraire, noter 0. Décaler la somme ou la différence obtenue d'un rang vers la gauche, et répéter le même cycle, et ainsi de suite. La virgule suit le premier chiffre du quotient obtenu. Si le quotient est négatif, il est obtenu sous la forme de son complément vrai.

Dans l'exemple ci-dessous, nous désignons par D le dividende, par d le diviseur, par r_i les restes successifs et par q_i les chiffres successifs du quotient. Le résultat sera arrondi à 4 chiffres binaires après la virgule en égalant systématiquement à 1 le chiffre de plus petit poids (*voir ci-dessous : arrondi*).

$$D = 1{,}0111\ (-9/16); \qquad d = 0{,}1011\ (11/16).$$

D d r_1	1, 0 1 1 1 0, 1 0 1 1 0, 0 0 1 0	$q_7 = 1$
$2\ r_1$ $(2 - d)$ r_2	0, 0 1 0 0 1, 0 1 0 1 1, 1 0 0 1	$q_1 = 0$
$2\ r_2$ d r_3	1, 0 0 1 0 0, 1 0 1 1 1, 1 1 0 1	$q_2 = 0$
$2\ r_3$ d r_4	1, 1 0 1 0 0, 1 0 1 1 0, 0 1 0 1	$q_3 = 1$
		$q_4 = 1$ (arrondi)

Le quotient est donc : 1,0011 — soit le nombre négatif — 0,1101, équivalent binaire de — 13/16, ce qui est bien le quotient des deux nombres proposés arrondi au quatrième rang binaire après la virgule.

L'établissement du programme correspondant à ce processus serait, pour le lecteur, un excellent exercice.

La méthode suivante présente l'avantage de mobiliser une faible capacité de mémoire, mais est assez lente. Le même principe est toutefois applicable au calcul d'autres fonctions. Soit à calculer le quotient a/b, les deux nombres a et b étant tous deux inférieurs à l'unité et, de plus, a plus petit que b. On part de la valeur 1/2, que l'on multiplie par b. Si le résultat est supérieur à a, on soustrait 1/4 de la valeur de départ; sinon, on ajoute 1/4. On multiplie la nouvelle valeur ainsi obtenue par b, et on lui soustrait ou on lui ajoute 1/8 selon que le produit est supérieur ou inférieur à a. On continue ainsi à ajouter ou à soustraire 1/16, 1/32, etc..., jusqu'à ce que la précision cherchée soit atteinte, c'est-à-dire jusqu'à ce que la différence entre deux résultats successifs soit inférieure à une unité de plus petit poids des nombres traités.

Mais les méthodes les plus couramment utilisées pour le calcul du quotient de deux nombres sont des méthodes d'itération (approximations successives). Une méthode fréquemment utilisée pour le calcul du quotient de deux nombres positifs a et b consiste à calculer la suite de nombres définie par les formules :

$$a_{n+1} = a_n (2 - b_n)$$
$$b_{n+1} = b_n (2 - b_n)$$

en prenant pour valeurs de départ les nombres a et b eux-mêmes. On voit que :

$$\frac{a_{n+1}}{b_{n+1}} = \frac{a_n}{b_n} = \ldots = \frac{a}{b}$$

et que b_{n+1} tend vers l'unité quand n croît; il s'ensuit que a_{n+1} tend vers a/b.

Arrondi.

Dans tout calcul numérique, il est nécessaire, à certains moments, d'arrondir correctement certains résultats. En effet, pour arrondir un nombre à N chiffres après la virgule, il ne suffit pas d'abandonner purement et simplement les décimales suivantes, car on commet ainsi une erreur moyenne non nulle, mais égale à $\eta/2$, en désignant par η une unité de plus petit poids. Les deux méthodes suivantes donnent, par contre, une erreur moyenne nulle :

a) Le dernier chiffre conservé est remplacé, s'il est pair, par le chiffre impair immédiatement supérieur;

b) Le dernier chiffre conservé est augmenté d'une unité, i la partie abandonnée est supérieure ou égale à $\eta/2$; ceci evient à ajouter $\eta/2$ et à conserver les N premiers chiffres lu résultat.

Dans le système binaire, la première méthode consiste simplement à prendre systématiquement 1 comme dernier chiffre conservé, et la seconde à ajouter une unité du $(N + 1)^e$ rang.

La seconde méthode est plus précise, car elle donne une erreur quadratique moyenne égale à $\eta^2/12$, contre $\eta^2/3$ pour la première. Elle est employée pour la multiplication lorsque l'on dispose d'un totalisateur de capacité 2 N. La première méthode est couramment utilisée pour la division, comme nous l'avons fait dans l'exemple ci-dessus.

Exemples :

Nombres binaires à arrondir à 4 chiffres après la virgule	Nombres arrondis	
	Méthode (*a*)	Méthode (*b*)
0,1101 1001	0,1101	0,1110
0,1100 1001	0,1101	0,1101

Aperçu sur le contenu d'une collection de sous-programmes.

La commodité d'emploi d'une calculatrice numérique automatique est étroitement liée à la richesse de la collection de sous-programmes dont elle est munie, collection qui s'enrichit d'ailleurs sans cesse au cours de l'exploitation de la machine. On doit toutefois, dès le départ, disposer d'une collection minimum, dont la préparation incombe au fournisseur de la machine.

Parmi les fonctions élémentaires d'usage très fréquent, la *division*, la *racine carrée* et la *racine cubique* sont avantageusement effectuées par des méthodes d'itération, tandis que les *fonctions trigonométriques* directes et inverses, les *exponentielles* et les *logarithmes* sont commodément calculées par leurs développements en série. Cette dernière méthode est généralement plus avantageuse, dans le cas du calcul automatique, que l'interpolation entre les valeurs d'une table mise en mémoire. On peut toutefois considérer les coefficients d'un développement en série comme la quintessence d'une table numérique, et le développement lui-même comme une formule d'interpolation.

Un ou plusieurs sous-programmes seront consacrés à l'opération de *quadrature*. On a fréquemment recours, à cet effet, à des formules d'intégration assez élaborées, telles que la formule d'intégration de GAUSS, dans laquelle les intervalles sont inégaux. Il en résulte à la fois un accroissement de la vitesse de calcul et une économie de mémoire.

La résolution des *équations différentielles* ordinaires est également l'objet d'un ou plusieurs sous-programmes. Comme toute équation différentielle ordinaire peut toujours être mise sous la forme d'un système d'équations différentielles du premier ordre simultanées, un sous-programme de résolution d'un tel système permettra la résolution de toutes les équations différentielles ordinaires.

Le *calcul matriciel* étant à la base de nombreux procédés de calcul, des sous-programmes seront consacrés à la multiplication des matrices, à la résolution des systèmes d'équations algébriques linéaires simultanées, à l'inversion des

matrices, au calcul des valeurs et des vecteurs propres, etc... La résolution des systèmes d'équations linéaires est effectuée, soit par des méthodes d'élimination successive et systématique des variables, soit par des méthodes d'itération ou de répétition. Les méthodes par élimination successive sont souvent d'emploi commode en calcul automatique.

Nous parlerons ultérieurement des sous-programmes associés aux organes d'entrée et de sortie.

Disons quelques mots pour terminer des sous-programmes d'interprétation.

Sous-programmes d'interprétation.

Nous avons déjà montré qu'un sous-programme peut être considéré par le programmeur comme une instruction supplémentaire du code dont il dispose. *On peut encore aller plus loin et permettre au programmeur de travailler, pour certains calculs, avec un code totalement différent de celui dont est dotée la machine.* Chaque instruction écrite dans ce nouveau code doit alors être interprétée par un sous-programme construit à cet effet dans le but d'élaborer la suite d'instructions ordinaires équivalente à l'instruction à interpréter.

Supposons, par exemple, que nous désirions effectuer des additions et des soustractions de *quantités complexes*, sans avoir à nous préoccuper que nous travaillons sur des nombres complexes. Nous conviendrons de placer la partie réelle et la partie imaginaire d'un nombre complexe donné dans deux cellules de mémoire consécutives d'adresses n et $(n + 1)$, et nous appellerons cette double adresse l' « adresse » de notre nombre complexe. Nous devrons également disposer d'un « registre d'opérateur complexe » constitué par deux positions de mémoire consécutives, 10 et 11 par exemple.

Pour effectuer un calcul sur des nombres complexes, nous commencerons donc par appeler le sous-programme d'interprétation correspondant, préalablement mis en mémoire. A partir de ce moment, les instructions d'addition $A\ n$, de soustraction $S\ n$, de remplacement $R\ n$ et de transfert $T\ n$ devront être comprises comme s'appliquant au nombre

complexe dont la partie réelle est contenue dans la mémoire d'adresse n, et, par conséquent, la partie imaginaire à l'adresse $(n + 1)$. Le sous-programme d'interprétation extraira alors une à une les instructions à interpréter du programme principal, et provoquera leur exécution en élaborant les groupes d'instructions ordinaires convenables. Par exemple, pour l'exécution de l'instruction $A\,n$, il faudra additionner la partie réelle (n) avec celle, (10), du registre complexe, et transférer le résultat dans 10, puis additionner de même les parties imaginaires $(n + 1)$ et (11) et transférer leur somme dans 11.

Le mécanisme général, au moyen duquel ces opérations sont effectuées, est le suivant. A chaque lettre de fonction du code interprétatif est associé un groupe d'instructions ordinaires provoquant l'exécution de l'opération correspondante. Ces différents groupes peuvent être appelés par autant d'instructions de ruptures de séquence constituant en somme un *répertoire*. Après avoir extrait l'instruction à interpréter du programme principal, le sous-programme d'interprétation examine d'abord sa lettre de fonction, qu'il isole par intersection. D'après sa valeur numérique, il fabrique alors une instruction de rupture de séquence qui renvoie à l'instruction convenable du répertoire et, de là, au groupe correspondant d'instructions ordinaires. L'adresse de l'instruction à interpréter, qui est indispensable pour l'exécution correcte de ce groupe d'instructions, aura donc été mise en réserve au moment de l'examen de la lettre de fonction.

Des sous-programmes d'interprétation sont également utiles pour les calculs suivants :

a) *Calculs en double précision* : pour traiter, sur une machine à N chiffres, des nombres de $2\,N$ chiffres, il suffit de loger les deux moitiés des nombres dans deux cellules de mémoire consécutives, et de programmer en conséquence; ici encore, un sous-programme d'interprétation sera utile, car il permettra, une fois appelé, d'utiliser les lettres de code normales en liaison avec des nombres de longueur double. La même méthode permet évidemment de traiter des nombres de longueur quelconque.

b) *Calculs en virgule flottante :* il est souvent impossible de travailler en virgule fixe, par suite de l'ignorance dans laquelle on se trouve des ordres de grandeurs de certaines quantités. On adopte alors la notation dite en « virgule flottante », dans laquelle les nombres sont représentés sous la forme semi-logarithmique $p \cdot B^q$, B étant la base de numération, p et q des nombres positifs ou négatifs; de plus, la mantisse p doit être comprise dans l'intervalle $1 \leqslant p < B$. Avant de procéder à l'addition ou à la soustraction de deux nombres, on commence donc par « cadrer » le plus petit par rapport au plus grand, en lui donnant le même exposant; après l'opération, il faut éventuellement recadrer le résultat, de manière que sa mantisse satisfasse à la condition énoncée ci-dessus. La multiplication et la division sont effectuées respectivement en faisant le produit ou le quotient des mantisses, la somme ou la différence des exposants, et en recadrant le résultat.

La mantisse et l'exposant seront logés, soit dans la même position de mémoire, soit dans deux ou plusieurs positions consécutives.

c) *Calculs en nombres décimaux :* dans certains cas, il peut être avantageux de traiter directement des nombres exprimés en numération décimale. Il suffit, pour cela, d'exprimer chaque chiffre décimal par un groupe de 4 chiffres binaires consécutifs, et de traiter ces groupes séparément au moyen d'un sous-programme convenable. Ce genre de fonctionnement est assuré automatiquement dans les machines travaillant en numération décimale.

Enfin, il n'est pas interdit de travailler à la fois en numération décimale et en virgule flottante, comme le font automatiquement plusieurs machines récentes.

Instructions symbolisées.

Les codes interprétatifs envisagés ci-dessus sont formés d'instructions analogues à celles du code normal, mais s'appliquant à des nombres de caractère spécial. On peut

encore aller plus loin et essayer de libeller les instructions sous une forme qui se rapproche le plus possible de l'écriture mathématique habituelle, par exemple :

$(n'') = (n') \times (n)$: inscrire dans la mémoire n'' le produit des contenus des deux mémoires n' et n;

$(n') = \sqrt{(n)}$: inscrire dans la mémoire n' la racine carrée du contenu de la mémoire d'adresse n.

Dans une telle convention, les signes d'opération, tels que « × » ou « $\sqrt{\ }$ », ont des équivalents numériques, tout comme les lettres de fonction ordinaires. Il est donc bien évident qu'un sous-programme d'interprétation convenable est susceptible de gouverner l'exécution de telles opérations écrites sous cette forme. Toutefois, il est clair que l'encombrement dans la mémoire d'une telle instruction est supérieu à celui d'une instruction ordinaire, tout au moins dans l cas d'un code à simple adresse. Il faut donc, soit affecter deux ou plusieurs cellules de mémoire à chaque instruction symbolisée, soit convenir que les opérations symbolisées ne s'appliquent qu'à une partie de la mémoire, afin de limiter la valeur maximum des adresses n; c'est cette dernière solution qui sera en général retenue, les machines possédant fréquemment, comme nous le verrons, une mémoire de grande capacité à accès relativement lent et une mémoire de capacité plus réduite, mais d'accès rapide; les adresses figurant dans les opérations symbolisées seront donc exclusivement celles de tout ou partie de la mémoire rapide.

Organes d'entrée et de sortie.

Ces organes permettent en somme à la machine de communiquer avec son utilisateur.

ORGANES D'ENTRÉE. — Les informations à communiquer à une calculatrice automatique — données numériques et instructions du programme — doivent être inscrites sur un support matériel susceptible d'être lu par l'organe d'entrée.

On utilise encore fréquemment à cet effet le *ruban de télétype*, auquel nous nous limiterons pour le moment.

Une *perforatrice*, munie d'un clavier à touches, permet de percer des trous dans le ruban, un trou étant représentatif du chiffre binaire 1 et l'absence de trou du chiffre 0. Les configurations de trous ou d'absences de trous sont groupées par lignes de 5 positions chacune. Une ligne entière est perforée par la manœuvre de chaque touche. Chaque ligne de trous et de blancs constitue donc un groupe de 5 chiffres binaires. Le lecteur de ruban perforé comporte 5 balais réalisant ou non un contact électrique selon qu'ils rencontrent un trou ou un blanc.

Les touches du clavier portent les chiffres de 0 à 9 et les diverses lettres de fonction. Il est avantageux d'adopter un code numérique différent de celui utilisé dans les liaisons par télétype.

Le plus commode est de représenter les 10 chiffres décimaux par leurs équivalents binaires. Une partie de la conversion décimale-binaire des nombres est ainsi effectuée automatiquement lors de la préparation du ruban. Les combinaisons binaires comprises entre 10 et 31 inclusivement sont donc disponibles pour les lettres de fonction.

Lors de la lecture d'une ligne du ruban, les 5 chiffres binaires correspondants sont transférés dans les 5 positions de plus faibles poids d'un registre de la mémoire, au moyen de l'instruction :

E n : lire la ligne suivante du ruban, et placer le nombre binaire lu, multiplié par 2^{-N}, dans la mémoire d'adresse *n*.

Lors de la lecture d'un groupe de lignes contenant les chiffres successifs d'un nombre décimal, la suite de la conversion décimale-binaire est effectuée par la machine elle-même, au moyen d'un sous-programme approprié. La conversion consiste simplement à multiplier les rangs successifs par les puissances correspondantes de 10, supposées mises en mémoire, et à faire la somme de ces produits. Toutefois, dans le cas de nombres fractionnaires, ceci n'est pas tout à

fait suffisant. Soit par exemple à convertir 0,84 en numération binaire. Commençons par calculer l'équivalent binaire de 84 :

$$84 = 4 + (8 \times 10)$$

soit, en numération binaire :

$$100 + (1000 \times 1010) = 1010100.$$

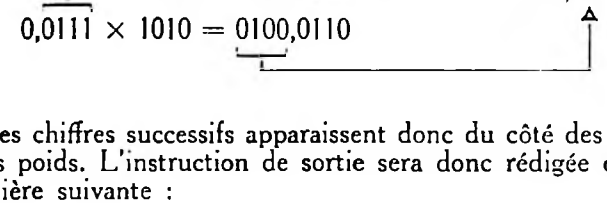
Mais, mis sous la forme 0,01010100, ce nombre est l'équivalent binaire de $84/(16)^2$, et non celui de $84/(10)^2$, comme nous le désirons. Il faut donc le multiplier par $(16/10)^2$, ou encore le diviser par $(10/16)^2$, dont l'équivalent binaire est 0,011001, ce qui donne 0,110101 qui, arrondi à 5 chiffres après la virgule, donne bien $27/32 = 0{,}84...$

Les différents ordres d'informations inscrites sur le ruban perforé sont en général préparées indépendamment. Dans une organisation typique, les données numériques, le programme principal et les sous-programmes étudiés spécialement pour le problème en cours sont perforés en double exemplaire, puis vérifiés au moyen d'une machine spéciale appelée « vérificatrice »; cette dernière s'arrête dès qu'il y a désaccord entre les deux rubans qu'elle lit en synchronisme. On perfore ensuite, également en double exemplaire pour vérification ultérieure, le ruban définitif, par recopie, au moyen d'une « copieuse », des rubans partiels ainsi préparés et de rubans extraits de la collection de sous-programmes préparés à l'avance.

ORGANES DE SORTIE. — Une forme d'organe de sortie tout à fait symétrique de l'organe d'entrée qui vient d'être décrit est constituée par un *téléimprimeur*, dont le mécanisme de frappe des lettres et des chiffres est commandé par 5 solénoïdes excités en parallèle selon un code à 5 moments. Certaines valeurs du nombre binaire à 5 chiffres constituent le code commandant la sélection des lettres ou des chiffres, l'avance du papier et le saut de ligne. La mise en page peut ainsi être programmée, divers sous-programmes correspondant à divers mode de présentation des résultats.

Avant leur impression, le nombres doivent être convertis en numération décimale au moyen d'un sous-programme. Considérons tout d'abord un nombre déjà exprimé sous la forme d'une fraction décimale. On peut faire apparaître le chiffre de plus fort poids à gauche de la virgule en multipliant le nombre considéré par 10. En supprimant cette partie entière et en multipliant à nouveau la partie fractionnaire restante par 10, on fait apparaître le second chiffre à gauche de la virgule, et ainsi de suite.

Le même procédé s'applique à un nombre exprimé dans le système binaire, en multipliant chaque fois la partie fractionnaire par l'équivalent binaire de 10, soit 1010. Soit, par exemple, à convertir en numération décimale le nombre binaire 0,11011 (27/32 = 0,84 . . .). Les deux premiers produits successifs par 1010 s'écrivent comme suit :



Les chiffres successifs apparaissent donc du côté des plus forts poids. L'instruction de sortie sera donc rédigée de la manière suivante :

S n : transférer les 5 chiffres de plus forts poids du contenu de l'adresse *n* de la mémoire au téléimprimeur et imprimer le caractère correspondant.

Toutefois, il est souvent plus avantageux de ne pas commander le téléimprimeur directement, mais par l'intermédiaire d'une perforatrice électrique de ruban elle-même commandée par la machine au moyen d'une instruction analogue à la précédente. La cadence maximum d'une perforatrice étant supérieure à celle d'un téléimprimeur (15 caractères par seconde contre 5 ou 6), un tel système à l'avantage de permettre une cadence moyenne de sortie

supérieure, le téléimprimeur rattrapant son retard pendant les périodes de calcul sans sortie de résultats.

Il peut également être avantageux de choisir un code de sortie spécial différent du code d'entrée, afin de contrôler le fonctionnement du téléimprimeur. Par exemple, les chiffres décimaux, et eux seuls, seront représentés par des combinaisons contenant toutes deux 1 et trois 0. Une erreur de transmission unique entraînera donc l'application au téléimprimeur d'une combinaison contenant, soit trois 1, soit un seul 1, et ne représentant donc pas un chiffre. Une telle erreur sautera aux yeux lors de l'examen d'un tableau de résultats numériques. Ce n'est que dans le cas — hautement improbable — de deux erreurs de transmission qui se compensent, qu'un chiffre faux serait imprimé (1).

Programme d'instructions initiales.

Le travail d'une calculatrice automatique, y compris les opérations d'entrée et de sortie décrites ci-dessus, se déroule sous l'action d'un programme d'instructions préalablement mis en mémoire. Une fois démarré, ce processus peut continuer indéfiniment, le programme pouvant, le cas échéant, commander à certains moments la lecture d'informations supplémentaires inscrites sur le ruban d'entrée. Mais il est clair que le « démarrage » doit être assuré par une méthode auxiliaire indépendante de la séquence de commande normale et permettant d'inscrire des instructions dans la mémoire. Cette opération est assurée par un programme dit d' « *instructions initiales* » mis en mémoire par un moyen extérieur avant toute autre chose. Si la première instruction de ce programme est inscrite à l'adresse 0 de la mémoire et si le compteur ordinal est mis à zéro avant la mise en marche de la machine, les instructions initiales provoqueront la lecture

(1) L'accélération du fonctionnement des organes d'entrée et de sortie est l'un des soucis actuels des constructeurs; le ruban magnétique joue un rôle important dans ces développements.

et la mise en mémoire, sous forme convenable, des instructions et des données numériques portées par le ruban d'entrée, jusqu'à ce que l'on rencontre une indication codée provoquant le début du calcul par renvoi à la première instruction à exécuter.

Pour chaque ligne du ruban de programme, les opérations à exécuter sont essentiellement les suivantes :

— lire la lettre de fonction et la mettre en réserve;

— lire et convertir en numération binaire les chiffres successifs de l'adresse;

— réunir la lettre de fonction et l'adresse pour constituer l'instruction;

— transférer l'instruction ainsi construite à l'adresse convenable de la mémoire;

— augmenter d'une unité l'adresse de l'instruction de transfert ci-dessus;

— recommencer en lisant la lettre de fonction de l'instruction suivante, et ainsi de suite.

Des groupes de symboles spéciaux, appelés « combinaisons de commande » et non destinés à être mis en mémoire, permettent d'indiquer la position de mémoire à partir de laquelle les instructions qui vont être lues doivent être inscrites, de repérer la dernière instruction à transférer et de renvoyer au début du programme, etc...

Mais le programme d'instructions initiales doit encore assurer une fonction très importante, sans laquelle l'emploi des sous-programmes serait malaisé. L'examen des sous-programmes établis précédemment montre que certaines de leurs instructions se réfèrent à des adresses du sous-programme lui-même. Il en résulte que l'écriture de ces instructions dépend de la position occupée dans la mémoire par la première instruction du programme. Ceci est évidemment mal commode et il y a le plus grand intérêt à ce que les sous-programmes puissent être préparés sous une forme telle que leur fonctionnement correct ne dépende pas de leur

emplacement dans la mémoire, à moins évidemment qu'ils ne soient inscrits à demeure dans la mémoire, ce qui n'est pas le cas en règle générale.

En d'autres termes, il doit être possible de spécifier les instructions visées ci-dessus en valeur relative, par rapport au début du sous-programme, et non en valeur absolue. Certaines machines possèdent un code permettant d'obtenir ce résultat automatiquement, sans aucune modification du sous-programme, au moyen d'instructions du type de la suivante :

$A' n$: ajouter au contenu du registre de l'opérateur celui de l'adresse $(m + n)$ de la mémoire, l'adresse m étant celle du registre contenant cette instruction elle-même.

En l'absence d'une telle facilité, la modification des instructions contenant des *adresses relatives* est assurée par les instructions initiales. A cet effet, les instructions du sous-programme préparé à l'avance sont numérotées arbitrairement à partir de 0, les adresses relatives étant suivies d'une lettre-code spéciale. Au moment de la lecture et de la mise en mémoire du sous-programme sous l'effet du programme d'instructions initiales, les adresses ainsi repérées sont majorées de l'adresse à laquelle est placée en mémoire la première instruction du sous-programme. On obtient ainsi, du point de vue du programmeur, la même souplesse qu'avec un système de numérotation relative des adresses.

DEUXIÈME PARTIE

NOTIONS SUR LA TECHNOLOGIE DES CALCULATRICES NUMÉRIQUES AUTOMATIQUES

Nous espérons que la première partie de cet ouvrage aura permis à notre lecteur de se familiariser progressivement avec les principes fondamentaux du calcul numérique automatique. Nous nous proposons maintenant de passer rapidement en revue les principales techniques électroniques mises en œuvre dans les calculatrices automatiques actuelles.

Nous examinerons successivement deux sortes de technologies : l'une à base de *diodes* et de tubes électroniques, ces derniers pouvant d'ailleurs tout aussi bien être remplacés par des transistors; l'autre à base de *tores magnétiques*. Pour chacune d'elles, nous commencerons par étudier les circuits fondamentaux, avant d'en donner quelques applications typiques. Nous terminerons par quelques indications sur les mémoires magnétiques dynamiques (tambours et rubans) et les mémoires électrostatiques, les machines décimales, les méthodes de contrôle et, enfin, l'application de l'algèbre binaire à la résolution des problèmes de logique.

Représentation et transmission des informations.

Dans les registres des organes de calcul, de commande et de mémoire, les chiffres binaires sont représentés par des organes doués de deux états stables tels que les deux états d'une bascule électronique ou les deux états de saturation d'un matériau magnétique à cycle d'hystérésis rectangulaire.

Les chiffres des mots — nombres ou instructions — sont transmis d'un organe à un autre par des lignes électriques

sous la forme d'impulsions brèves. Notons immédiatement que les signaux représentant les chiffres d'un même mot peuvent être transmis simultanément sur autant de lignes séparées (*fonctionnement parallèle*), ou successivement sur une même ligne, sous la forme d'un train d'impulsions (*fonctionnement série*). Ainsi, la *figure 44* montre la transmission du nombre binaire 1011 dans ces deux modes de fonctionnement, en supposant simplement pour le moment

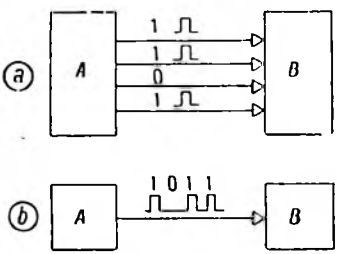


FIG. 44. — Transfert parallèle (*a*) et série (*b*).

qu'un « 1 » est représenté par une impulsion positive, et un « 0 » par l'absence d'impulsion (1). Dans une machine décimale, dans laquelle les chiffres décimaux sont codés dans un système binaire, on a le choix entre trois possibilités : transmettre tous les chiffres binaires en parallèle, les transmettre tous en série, les différents groupes correspondant aux chiffres décimaux successifs se succédant dans le temps, et, enfin, la solution mixte qui consiste à transmettre en parallèle les chiffres binaires d'un même chiffre décimal et en série les impulsions des groupes successifs.

Mais il convient d'examiner de plus près les conditions dans lesquelles s'effectue la transmission de ces signaux. Ils ne sauraient avoir pratiquement la forme théorique indiquée sur la *figure 44* : leurs fronts ne peuvent être

(1) Dans le fonctionnement série, les chiffres de faible poids sont transmis en tête, pour le bon fonctionnement des circuits d'addition.

infiniment raides, car le passage d'une tension électrique d'un niveau à un autre en un temps infiniment court mettrait en jeu une énergie infinie; ils subissent toujours une certaine distorsion en traversant les organes auxquels ils sont appliqués; enfin, il s'y superpose toujours des fluctuations de

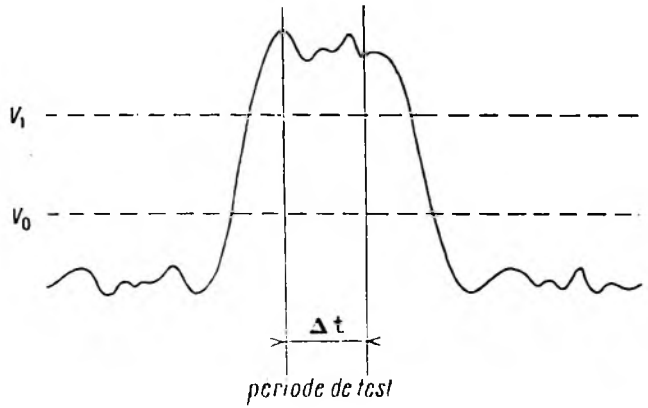


FIG. 45. — Interprétation de l'information binaire contenue dans un signal électrique : valeur 0 pour un signal inférieur à V_0, valeur 1 pour un signal supérieur à V_1, période de test Δt.

bruit de fond qui, dans certains cas, peuvent ne pas être négligeables.

Cette imperfection de la forme des signaux conduit à préciser la définition de l'information qu'ils transportent en fixant deux niveaux de référence V_0 et V_1 (*fig. 45*), tels qu'une tension inférieure à V_0 représente le chiffre binaire 0 et une tension supérieure à V_1 le chiffre binaire 1. En outre, l'examen de ce niveau ne peut évidemment se faire pendant les instants où il est susceptible de changer, mais bien au contraire pendant des intervalles de temps bien définis Δt pendant lesquels on est certain qu'il ne subira pas de variation significative. Ces intervalles de temps sont fréquemment

appelés *périodes de test*. On fera donc alterner, dans le fonctionnement de la machine, les périodes de test et les périodes transitoires. On voit ici apparaître la notion fondamentale de *découpage temporel*. La fréquence et la durée des périodes

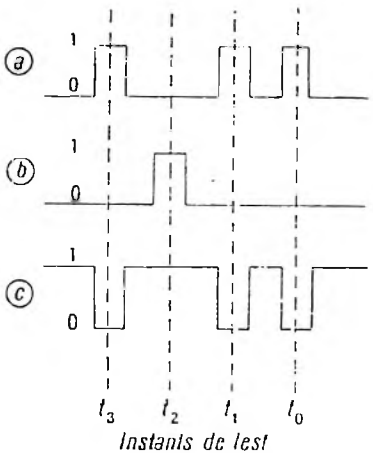


FIG. 46. — Testés aux instants indiqués, les deux signaux (*b*) et (c) contiennent la même information binaire, complémentaire de celle du signal (*a*).

de test sera définie par un générateur central appelé *horloge* ou *générateur de rythme* engendrant des *impulsions normales* de forme et de durée bien déterminées. Les fréquences de rythme fondamental des calculatrices actuelles se situe entre quelques kilohertz et quelques mégahertz (1). Dans la suite, nous désignerons par θ la période de rythme fondamental.

(1) Rappelons à ce propos que l'unité internationale de fréquence a reçu le nom de hertz (Hz), et non « cycle par seconde », et encore moins « période par seconde »; elle est systématiquement ignorée des anglo-saxons.

Dans une machine à fonctionnement série, la durée de transmission d'un mot de N chiffres, égale à $N\,0$, est appelée *cycle mineur*. Elle constitue en quelque sorte l'unité de temps du fonctionnement de la machine, puisque aucune opération ne peut se dérouler en un temps inférieur à un cycle mineur.

Montrons tout de suite, par un exemple, l'importance de cette notion de *test* ou de *prélèvement*. Considérons à nouveau (*fig. 46 a*) la suite temporelle d'impulsions représentant le nombre binaire 1011 en numération série, et demandons-

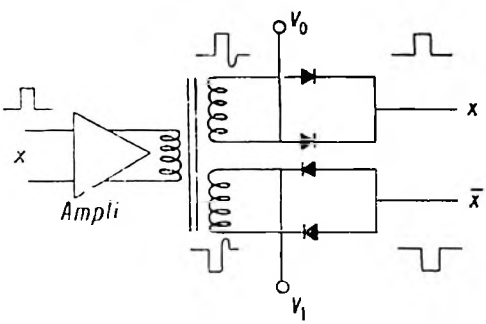


FIG. 47. — Transmission d'un signal et de son complément par une liaison par transformateur.

nous comment représenter son complément restreint 0100. Si l'on s'en tient à la définition simpliste donnée plus haut de la représentation de 1 par une impulsion et de 0 par l'absence d'une impulsion, on obtient la représentation de la *figure 46 b*. Considérons maintenant le signal (*c*), déduit du signal (*a*) par l'interversion du niveau bas V_0 et du niveau haut V_1. Si l'on teste les signaux (*b*) et (*c*) aux instants repérés par les traits interrompus, on voit qu'ils contiennent très exactement la même information. Or, du point de vue technique, l'obtention du signal (*c*) à partir du signal (*a*) est beaucoup plus facile que celle du signal (*b*). Considérons en particulier un transformateur (*fig. 47*) dont le primaire, alimenté par un amplificateur par exemple, reçoit un signal

tel que (*a*) de la *figure 46*. Deux secondaires en opposition de phase fourniront simultanément le signal (*a*) lui-même et son complément (*c*); une de leurs extrémités est polarisée, pour l'un à la tension V_0, pour l'autre à la tension V_1; enfin, des diodes éliminent les portions des signaux secondaires de niveaux supérieur à V_1 ou inférieur à V_0. De plus, la liaison par transformateur permet d'abaisser l'impédance au secondaire à une valeur telle que le signal puisse être transmis à une certaine distance dans de bonnes conditions, et en particulier sans risque de diaphonie par induction mutuelle avec les lignes voisines.

Symbolique des opérateurs logiques.

Nous avons vu que toutes les opérations de calcul et de commande mises en jeu dans le fonctionnement des calculatrices automatiques se ramènent aux trois opérations logiques fondamentales : intersection (*et*), réunion (*ou* inclusif, c'est-à-dire l'un ou l'autre ou les deux) et négation ou complément. Il est donc fort utile de disposer de symboles pour représenter les opérateurs correspondant à ces trois opérations. Cela permet en particulier de représenter la structure d'une calculatrice automatique sous la forme d'un *schéma logique* faisant partiellement abstraction de la technologie; partiellement seulement, car une fonction complexe

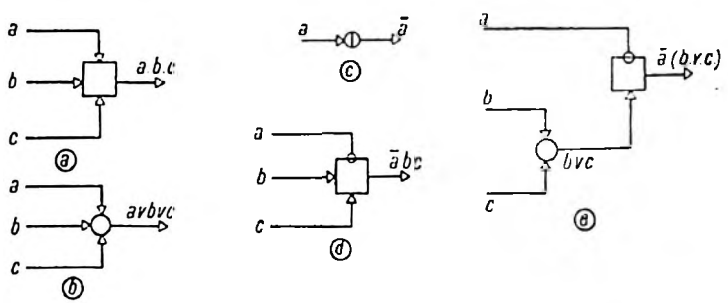


FIG. 48. — Symbolique des opérateurs logiques.

peut toujours être effectuée selon plusieurs schémas logiques, entre lesquels le choix sera dicté par des considérations d'ordre technologique.

Diverses conventions ont été proposées pour la représentation des opérateurs logiques. Une symbolique commode, illustrée par quelques exemples, est résumée sur la *figure 48.*

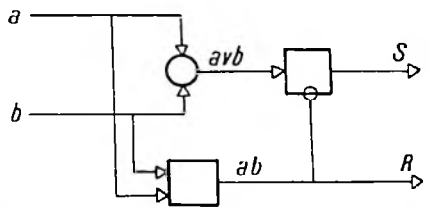


FIG. 49. — Représentation symbolique d'un demi-additionneur binaire.

Le symbole de négation est en général reporté à l'extrémité de la ligne transportant la variable considérée, à cheval sur le contour de l'organe auquel elle aboutit, comme on le voit en (*d*) et (*e*). Les organes d'intersection et de réunion sont souvent appelés *intersecteurs* et *mélangeurs* (1).

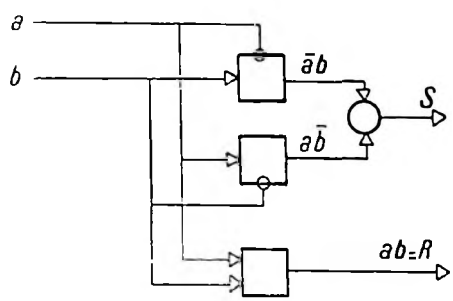


FIG. 50. — Autre forme de demi-additionneur binaire.

(1) En anglais, *gate* et *buffer*.

Rappelons les équations logiques du demi-additionneur, qui permet d'obtenir la somme S et la retenue R de 2 chiffres binaires a et b :

$$R = a \cdot b.$$

$$S = a \cdot \bar{b} \vee \bar{a} \cdot b = (a \vee b)\,\overline{a\,b}.$$

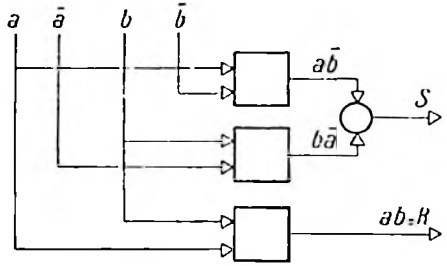


FIG. 51. — Modification du demi-additionneur de la *figure 50* au cas où l'on dispose simultanément des deux chiffres à additionner et de leurs compléments.

Les deux formes possibles de S correspondent aux deux schémas logiques des *figures 49* et *50*. Si, comme c'est fréquemment le cas, les variables et leurs compléments sont simultanément disponibles, ce dernier schéma se met sous la forme de celui de la figure *51*.

Comme nous l'avons déjà vu, deux demi-additionneurs $A/2$ peuvent être combinés pour donner un additionneur à trois entrées, permettant d'obtenir la somme S et la retenue R

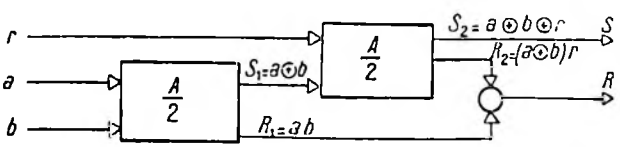


FIG. 52. — Additionneur binaire à deux demi-additionneurs.

de 2 chiffres binaires a et b et de la retenue r en provenance de l'étage précédent (*fig. 52*). Nous avons déjà fait remarquer que S_1 et R_1 ne peuvent simultanément être égaux à 1, de sorte que les retenues R_1 et R_2 des deux demi-additionneurs ne peuvent exister simultanément et peuvent donc être mélangées sans précaution. L'application de l'algèbre logique permet d'ailleurs de vérifier aisément que les variables S et R sont bien la somme et la retenue cherchées :

$$S_2 = r\,\bar{S}_1 \vee \bar{r}\,S_1$$

mais : $S_1 = a\bar{b} \vee \bar{a}b$

d'où : $\bar{S}_1 = \overline{a\bar{b}}\cdot\overline{\bar{a}b}$ (car $\overline{x \vee y} = \bar{x}\,\bar{y}$),

$= (\bar{a} \vee b)(a \vee \bar{b})$

$= a\,b \vee \bar{a}\,\bar{b}$ (car $a\,\bar{a} = 0$ et $b\,\bar{b} = 0$),

et $S_2 = a\,b\,r \vee a\,\bar{b}\,\bar{r} \vee \bar{a}\,b\,\bar{r} \vee \bar{a}\,\bar{b}\,r$

$= S(a, b, r)$.

S_2, égal à 1 quand une ou trois des trois variables a, b et r sont égales à 1, est bien la somme modulo 2 de ces 3 chiffres.

De même :

$$R_1 \vee R_2 = a\,b \vee r(a\,\bar{b} \vee \bar{a}\,b)$$

$$= a\,b(r \vee \bar{r}) \vee r(a\,\bar{b} \vee \bar{a}\,b) \quad (\text{car } r \vee \bar{r} = 1)$$

$$= a\,b\,r \vee a\,b\,\bar{r} \vee a\,\bar{b}\,r \vee \bar{a}\,b\,r$$

$$= R(a, b, r).$$

R est bien la retenue cherchée, égale à 1 quand deux ou trois des 3 chiffres a, b et r sont égaux à 1; on peut encore l'écrire :

$$R = a\,b \vee a\,r \vee b\,r.$$

On peut aussi construire un additionneur à trois entrées directement à partir des variables a, b et r, et de leurs compléments, en s'appuyant sur les expressions de S et de R obtenues ci-dessus. Ces équations, qui donnent le schéma logique de la *figure 53*, constituent d'ailleurs la traduction

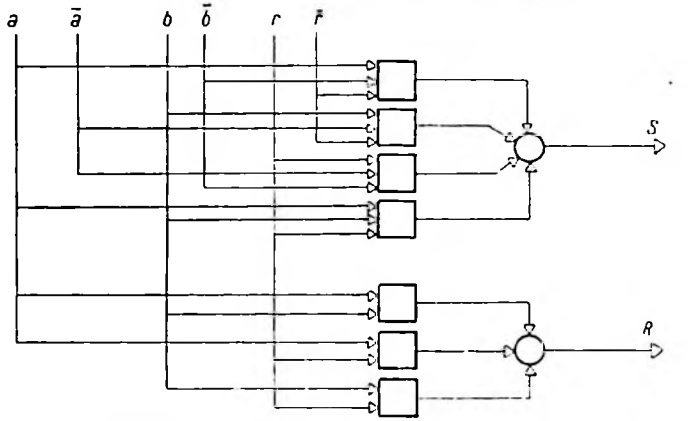


FIG. 53. — Additionneur binaire à trois entrées.

immédiate des résultats du tableau des huit cas possibles, reproduit ci-dessous :

a	b	r	S	R
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
0	0	1	1	0
1	1	0	0	1
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

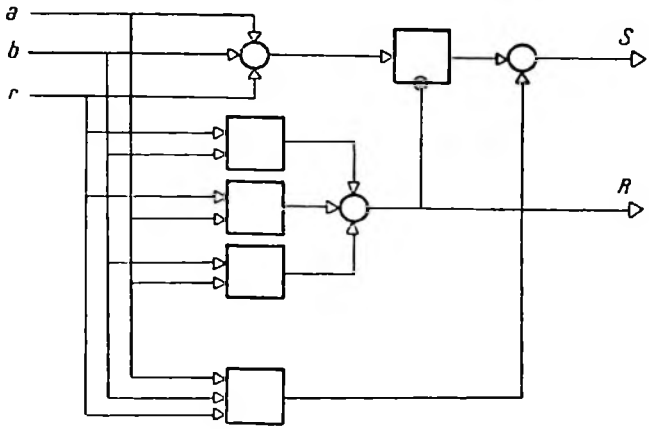


FIG. 54. — Autre forme d'additionneur à trois entrées.

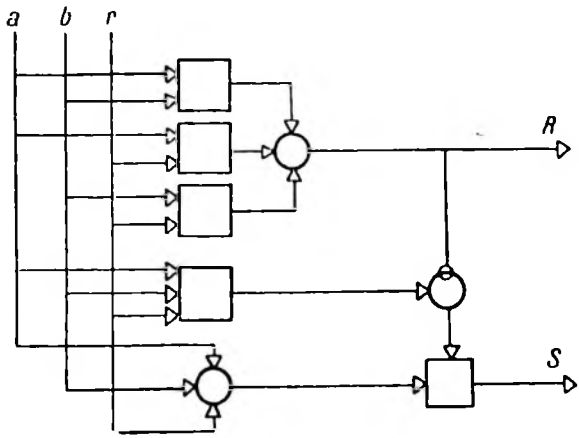


FIG. 55. — Autre forme d'additionneur à trois entrées.

Ce type d'additionneur, dont nous avons antérieurement construit la version à relais, possède la propriété remarquable de se correspondre à lui-même par complémentation : la somme et la retenue des compléments des variables sont respectivement égaux aux compléments de la somme et de la retenue de ces variables.

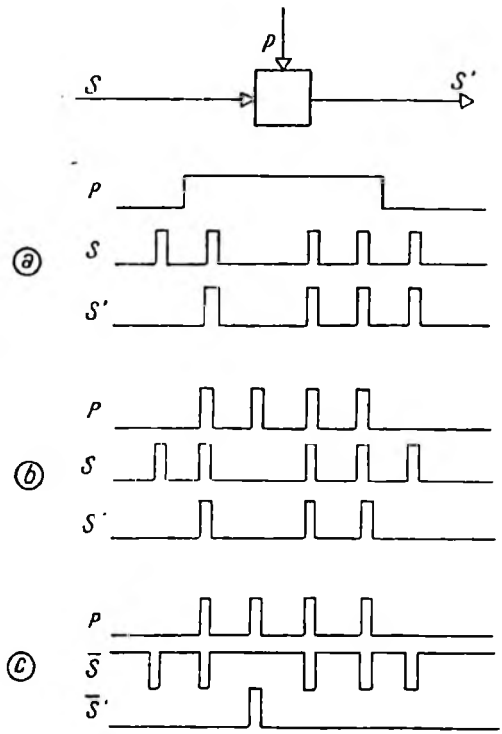


FIG. 56. — Porte : le signal d'ouverture de la porte peut se présenter, soit sous la forme d'un signal rectangulaire d'une durée égale à la durée d'ouverture (*a*), soit sous la forme d'un train d'impulsions de test (*b*) et (*c*).

L'examen du tableau ci-dessus ou la transformation des équations au moyen des règles de l'algèbre de BOOLE permet de mettre la somme S sous diverses formes, telles que les deux suivantes :

$$S = a\,b\,r \vee (a \vee b \vee r)\,\bar{R},$$

$$S = (a \vee b \vee r)\,(\bar{R} \vee a\,b\,r),$$

qui correspondent aux schémas logiques des *figures 54* et *55*.

PORTES. — Lorsqu'un circuit d'intersection est organisé de telle sorte que le passage d'un signal d'information soit subordonné à la présence simultanée d'un ou de plusieurs signaux de commande, on lui donne le nom de « porte », équivalent du « gate » des anglo-saxons. Considérons, par exemple, une porte gouvernant la transmission d'un nombre de N chiffres binaires (*fig. 56*). Selon la technologie adoptée, le signal de commande pourra être constitué, soit par N impulsions consécutives (*a*), soit par un signal rectangulaire de durée égale à un cycle mineur (*b*); dans le premier cas, si le signal incident se présente sous la forme d'un complément par inversion des polarités, il se trouve à la sortie remis sous la forme habituelle (*c*).

Circuits logiques à diodes.

Les diodes ont la propriété de présenter au passage du courant une résistance différente selon la polarité de la

FIG. 57. — Redresseur : le courant passe si V_1 est supérieur à V_2

tension qui leur est appliquée, ce qui permet en particulier de redresser le courant alternatif. Une diode idéale aurait

une résistance nulle pour un sens du courant et une résistance infinie pour l'autre sens. Dans la technique des machines à calculer électronique, on utilise beaucoup à l'heure actuelle les diodes à *cristaux de germanium*, qui présentent les avan-

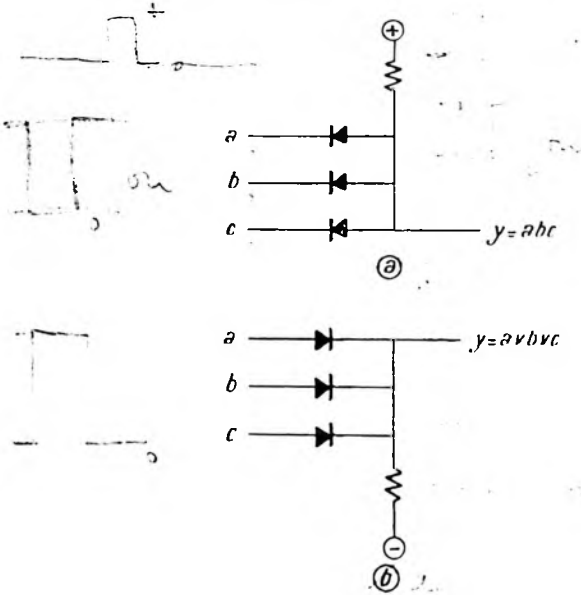


FIG. 58. — Circuits d'intersection et de réunion à diodes.

tages d'un faible encombrement, d'une longue vie, d'une résistance inverse élevée et d'une tension inverse relativement élevée. La *figure 57* montre la représentation schématique d'une diode, qui conduit si $V_1 > V_2$.

Les deux circuits logiques fondamentaux à diodes sont représentés sur la *figure 58*. Supposons que les chiffres binaires 0 et 1 soient représentés respectivement par les

niveaux (—) et (+). Dans le circuit (*a*), il suffit que l'une des entrées soit au niveau (+) pour que la diode correspondante conduise; la résistance directe des diodes étant supposée très faible devant la résistance de charge, la borne de sortie se trouve donc elle aussi au niveau (+). Ce circuit réalise donc la fonction de réunion (*ou*). Au contraire, dans le cas du circuit de la *figure 58 b*, il suffit que l'une des entrées soit au niveau (—) pour qu'un courant circule dans la résistance de charge et que, par suite, la borne de sortie soit au niveau (—); pour que la borne de sortie soit portée au niveau (+), il faut et il suffit que les trois entrées soient portées simultanément à une tension légèrement supérieure au niveau (+) : c'est la fonction d'intersection (*et*).

Remarquons que les fonctions de ces deux circuits s'inversent si l'on décide de représenter le chiffre 1 par le niveau (—) et le chiffre 0 par le niveau (+). Le circuit (*a*) devient alors un circuit d'intersection, puisque les trois entrées doivent être simultanément au niveau (—) pour que la borne de sortie y soit également. Inversement le circuit (*b*) réalise la fonction de réunion, puisqu'il suffit que l'une des entrées soit au niveau (—) pour qu'il en soit de même de la sortie. Cette dualité traduit les formules de négation de l'algèbre logique :

$$\bar{y} = \bar{a}\,\bar{b}\,\bar{c} = \overline{a \vee b \vee c} \qquad \text{pour le circuit } (a);$$

$$y = \bar{a} \vee \bar{b} \vee \bar{c} = \overline{a\,b\,c} \qquad \text{pour le circuit } (b).$$

Le fonctionnement correct de ces circuits suppose qu'ils sont alimentés par une source d'impédance faible et débitent dans une charge d'impédance élevée, telle que le circuit de grille d'un tube électronique. Cependant, moyennant certaines précautions, il est possible de monter plusieurs circuits logiques élémentaires en cascade, à la manière de la *figure 59*. Il est commode de mettre ces circuits sous la forme de matrices de commutation à diodes (*fig. 60*).

On peut réaliser ainsi n'importe quelle expression logique

se présentant sous la forme, soit d'une réunion d'intersections, soit d'une intersection de réunions.

Considérons par exemple la fonction de dilemme, ou addition modulo 2 :

$$y = a \cdot \bar{b} \vee b \cdot \bar{a} = a \oplus b$$

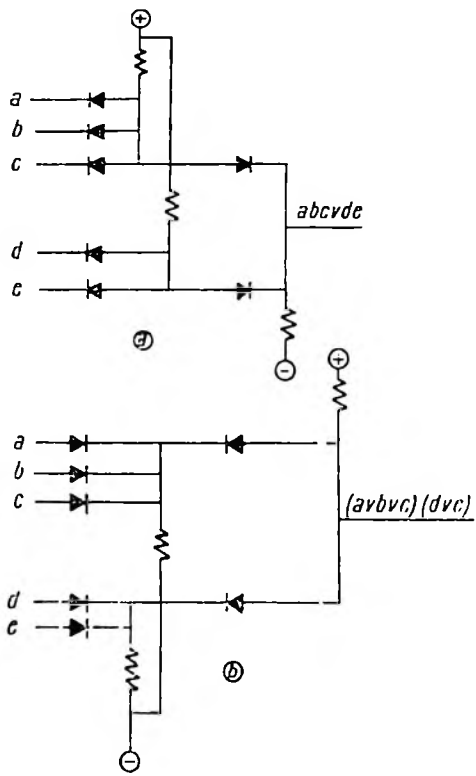


FIG. 59. — Exemples de circuits logiques à diodes.

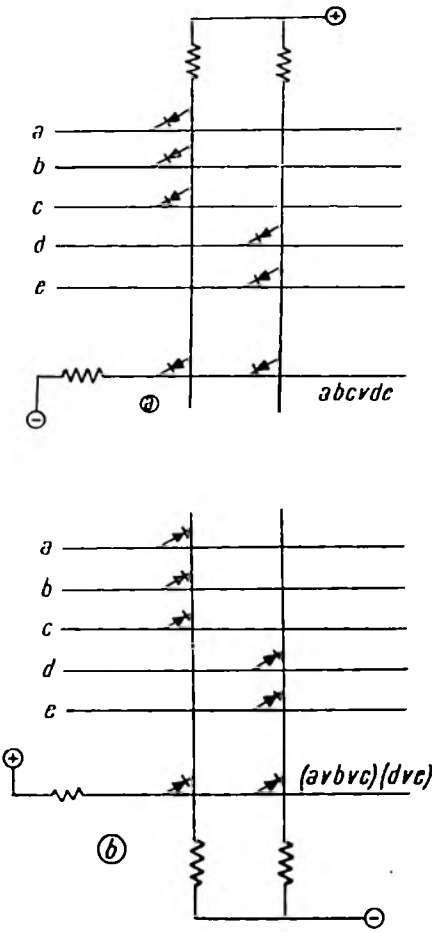


FIG. 60. — Forme matricielle des circuits logiques de la *figure 59*.

dont les quatre cas possibles sont rassemblés dans le tableau ci-dessous :

a	0	1	0	1
$\bar{a}$	1	0	1	0
b	0	0	1	1
$\bar{b}$	1	1	0	0
$a \oplus b$	0	1	1	0

Cette fonction est engendrée par la matrice de la *figure 61*, dont on remarquera que les diodes occupent les mêmes positions que les « 1 » du tableau ci-dessus.

De même, considérons à nouveau les équations de définition d'un additionneur à trois entrées, mises sous la forme suivante :

$$S = a\,\bar{b}\,\bar{r} \vee \bar{a}\,b\,\bar{r} \vee \bar{a}\,\bar{b}\,r \vee a\,b\,r;$$

$$R = a\,b \vee b\,r \vee r\,a.$$

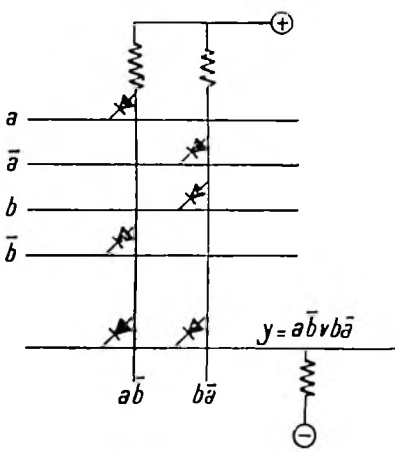


FIG. 61. — Circuit de dilemme à diodes.

Si l'on dispose simultanément des variables et de leurs compléments, ces équations sont satisfaites par la matrice à diodes de la *figure 62.*

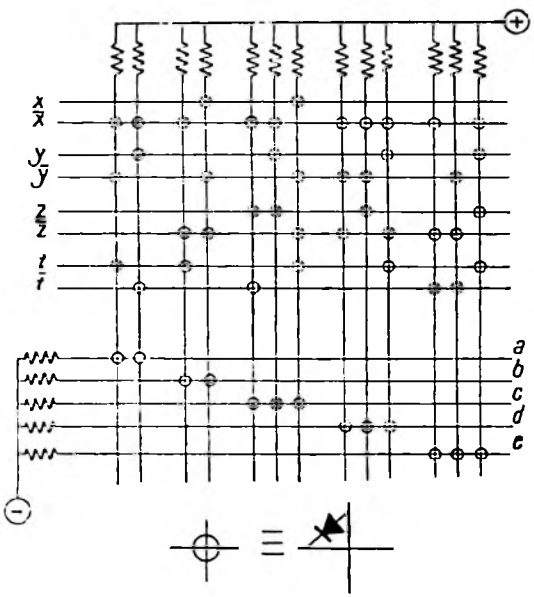


FIG. 62. — Additionneur binaire à diodes.

Ces circuits de commutation peuvent encore être considérés comme des circuits de *traduction* ou de *codage*, permettant de passer d'un code à un autre. Considérons, par exemple, la traduction de la forme binaire des chiffres décimaux dans un code à cinq moments tel que chaque chiffre décimal soit représenté par deux « 1 » et trois « 0 »; nous avons déjà indiqué les avantages d'un tel code en ce qui

concerne le contrôle. La traduction à effectuer est conforme au tableau ci-dessous :

Entrée	Sortie
$x\ y\ z\ t$	$a\ b\ c\ d\ e$
0 0 0 0	0 0 0 1 1
0 0 0 1	1 1 0 0 0
0 0 1 0	0 0 1 1 0
0 0 1 1	1 0 0 1 0
0 1 0 0	1 0 0 0 1
0 1 0 1	0 1 0 1 0
0 1 1 0	1 0 1 0 0
0 1 1 1	0 0 1 0 1
1 0 0 0	0 1 0 0 1
1 0 0 1	0 1 1 0 0

Comme c'est fréquemment le cas, les expressions logiques que traduisent ce tableau peuvent être simplifiées par application de l'algèbre de BOOLE [1] :

$$a = \bar{x}\,\bar{y}\,\bar{z}\,t \vee \bar{x}\,\bar{y}\,z\,t \vee \bar{x}\,y\,\bar{z}\,t \vee \bar{x}\,y\,z\,\bar{t}$$
$$= \bar{x}\,\bar{y}\,t\,(\bar{z} \vee z) \vee \bar{x}\,y\,\bar{t}\,(\bar{z} \vee z)$$
$$= \bar{x}\,\bar{y}\,t \vee \bar{x}\,y\,\bar{t}$$

$$b = \bar{x}\,\bar{y}\,\bar{z}\,t \vee \bar{x}\,y\,\bar{z}\,t \vee x\,\bar{y}\,\bar{z}\,\bar{t} \vee x\,\bar{y}\,\bar{z}\,t$$
$$= \bar{x}\,\bar{z}\,t\,(\bar{y} \vee y) \vee x\,\bar{y}\,\bar{z}\,(\bar{t} \vee t)$$
$$= \bar{x}\,\bar{z}\,t \vee x\,\bar{y}\,\bar{z}$$

(1) WILKES : *Automatic digital computers*, Methuen and Co. Ltd, 1956, p. 229. Édition française sous presse (Dunod, éd.).

$$
\begin{aligned}
c &= \bar{x}\,\bar{y}\,z\,\bar{t} \vee \bar{x}\,y\,z\,\bar{t} \vee \bar{x}\,y\,z\,t \vee x\,\bar{y}\,\bar{z}\,t \\
&= (\bar{x}\,\bar{y}\,z\,\bar{t} \vee \bar{x}\,y\,z\,\bar{t}) \vee (\bar{x}\,y\,z\,t \vee \bar{x}\,y\,z\,t) \vee x\,\bar{y}\,\bar{z}\,t \\
&= \bar{x}\,z\,\bar{t} \vee \bar{x}\,y\,z \vee x\,\bar{y}\,\bar{z}\,t \\
d &= \bar{x}\,\bar{y}\,\bar{z}\,\bar{t} \vee \bar{x}\,\bar{y}\,z\,\bar{t} \vee \bar{x}\,\bar{y}\,z\,t \vee \bar{x}\,y\,\bar{z}\,t \\
&= (\bar{x}\,\bar{y}\,\bar{z}\,\bar{t} \vee \bar{x}\,\bar{y}\,z\,\bar{t}) \vee (\bar{x}\,\bar{y}\,z\,t \vee \bar{x}\,\bar{y}\,z\,t) \vee \bar{x}\,y\,\bar{z}\,t \\
&= \bar{x}\,\bar{y}\,\bar{t} \vee \bar{x}\,y\,z \vee \bar{x}\,y\,\bar{z}\,t \\
e &= \bar{x}\,\bar{y}\,\bar{z}\,\bar{t} \vee \bar{x}\,y\,\bar{z}\,\bar{t} \vee \bar{x}\,y\,z\,t \vee x\,\bar{y}\,\bar{z}\,\bar{t} \\
&= (\bar{x}\,\bar{y}\,\bar{z}\,\bar{t} \vee \bar{x}\,y\,\bar{z}\,\bar{t}) \vee (\bar{x}\,\bar{y}\,\bar{z}\,\bar{t} \vee x\,\bar{y}\,\bar{z}\,\bar{t}) \vee \bar{x}\,y\,z\,t \\
&= \bar{x}\,\bar{z}\,\bar{t} \vee \bar{y}\,\bar{z}\,\bar{t} \vee \bar{x}\,y\,z\,t
\end{aligned}
$$

De ces expressions simplifiées résulte la matrice de traduction de la *figure 63*, dans laquelle les diodes sont figurées par des jonctions entre lignes et colonnes (1).

On remarque que la matrice ci-dessus donne des signaux de sortie tous nuls si la configuration $x\,y\,z\,t$ ne représente pas un chiffre décimal. Un circuit supplémentaire réalisant la fonction $\bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} \cdot \bar{e}$, ou encore $\overline{a \vee b \vee c \vee d \vee e}$ constitue donc un circuit de détection d'erreur repérant la formation d'un caractère non décimal.

Si l'on abandonne la représentation matricielle, les expressions ci-dessus peuvent encore se simplifier et se mettre sous la forme suivante :

$$
\begin{aligned}
a &= \bar{x}\,(\bar{y}\,t \vee y\,\bar{t}) \\
b &= \bar{z}\,(\bar{x}\,t \vee x\,\bar{y}) \\
c &= \bar{x}\,y\,(\bar{t} \vee y) \vee x\,\bar{y}\,z\,t \\
d &= \bar{x}\,\bar{y}\,(\bar{t} \vee z) \vee \bar{x}\,y\,\bar{z}\,t \\
e &= \bar{z}\,\bar{t}\,(\bar{x} \vee \bar{y}) \vee \bar{x}\,y\,z\,t
\end{aligned}
$$

(1) Une méthode systématique de simplification des expressions logiques a été indiquée par McCluskey, dans son article : Minimization of Boolean functions, *Bell System Technical Journal*, novembre 1956; elle fait l'objet d'un des chapitres de notre ouvrage, en préparation, sur les automatismes à séquences.

Les *sélecteurs* constituent des circuits de traduction particulièrement simples. Le problème consiste à sélecter une ligne parmi 2^n lignes au moyen de son numéro inscrit en numération binaire dans un registre à n cellules binaires.

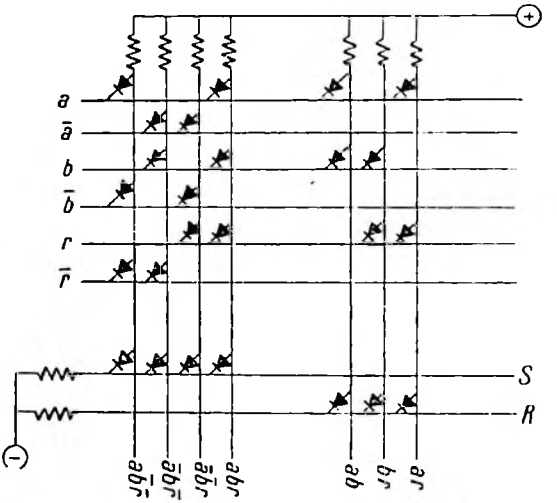


FIG. 63. — Circuit traducteur transformant le code binaire *xyzt* en un code à cinq moments *abcde*, dont deux toujours égaux à 1.

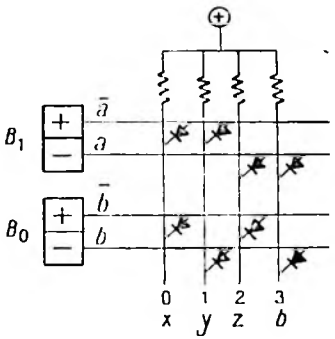
Ainsi, pour quatre lignes, le tableau ci-dessus traduit la correspondance à réaliser entre les quatre états possibles des deux cellules du registre et l'état des lignes :

a	$\bar{a}$	b	$\bar{b}$	x	y	z	t
0	1	0	1	1	0	0	0
0	1	1	0	0	1	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	0	0	1

D'où les quatre équations logiques :

$$x = \bar{a}\,\bar{b} \qquad y = \bar{a}\,b \qquad z = a\,\bar{b} \qquad t = a\,b$$

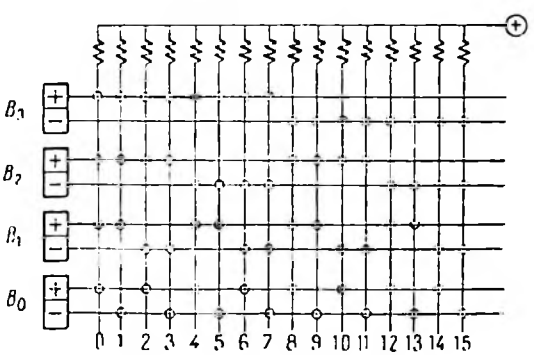
que l'on traduit facilement au moyen des quatre circuits d'intersection de la matrice à diodes de la *figure 64.*



B_1	B_0	Ligne
0	0	1
0	1	2
1	0	3
1	1	4

n=2 (4 *lignes*)
8 *diodes*

FIG. 64. — Sélecteur à 4 voies.



B_3	B_2	B_1	B_0	Ligne
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

n=4 (16 *lignes*), 64 *diodes*

FIG. 65. — Sélecteur à 16 voies.

La *figure 65* représente de même un sélecteur à 16 voies ($n = 4$); il comporte 64 diodes, schématisées par des petits cercles. On peut ramener ce nombre à 48 diodes en effectuant le décodage en deux temps (*fig. 66*). On commence par choisir une ligne dans un premier groupe de quatre au moyen des bascules *a* et *b* et, de même, une ligne dans un second groupe de quatre au moyen des bascules *c* et *d*.

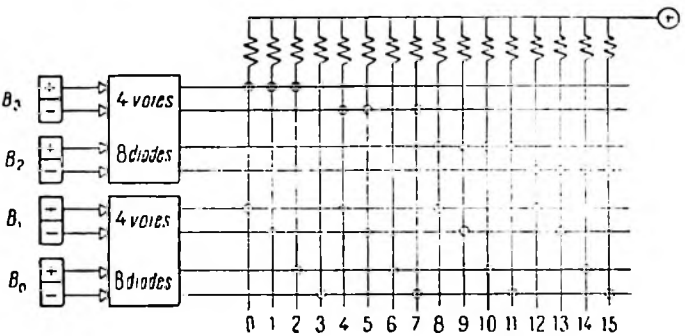


FIG. 66. — Sélecteur à 16 voies en deux étages de sélection.

On utilise ensuite les deux groupes de quatre lignes, dont deux seulement sont au niveau (+), pour sélecter l'une des 16 voies de sortie, au moyen du décodeur à 32 diodes de la *figure 66*. On effectue ainsi des coïncidences simples successives au lieu de la coïncidence quadruple de la *figure 65*. L'économie est encore plus grande pour $n = 8$ (256 voies): un décodeur unique nécessite 2048 diodes, et un décodeur à trois étages seulement 608 diodes.

Circuits à tubes électroniques.

CIRCUIT DE COMPLÉMENT. — Une triode, avec résistance de plaque, réalise très simplement la fonction de complément, puisque sa tension de plaque diminue lorsque croît

sa tension de grille, et inversement. Une impulsion positive appliquée à la grille donnera donc sur la plaque une impulsion négative (*fig. 67*). Les niveaux haut et bas du signal de sortie seront ramenés aux valeurs normales au moyen d'un pont de résistances établi entre la plaque et une source de polarisation négative. L'application sur la grille d'un train d'impul-

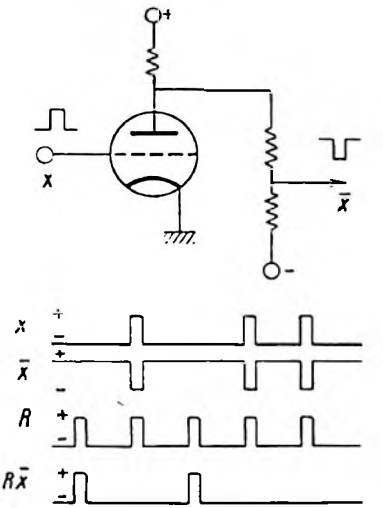


FIG. 67. — Circuit de complémentation à triode.

sions représentant un nombre binaire donnera donc à la sortie le complément restreint de ce nombre, si l'on convient de tester les signaux à des instants récurrents définis par le générateur de rythme, comme il a déjà été expliqué (*fig. 46*). Ainsi, le signal résultant de l'intersection de $\bar{x}$ avec les signaux de rythme R dérive de x par interversion de la présence et de l'absence des impulsions positives.

Les deux fonctions de complémentation et d'intersection peuvent être remplies par un tube unique, dont la cathode

reçoit le signal x à complémenter et la grille de commande le complément $\bar{R}$ du signal de rythme (*fig. 68*).

Nous avons déjà signalé (*fig. 56*) que la forme du signal de commande d'une porte destinée à gouverner la transmission d'un train d'impulsions dépendait de la technique utilisée pour la formation des compléments.

Remarquons encore que le couplage direct des *figures 67*

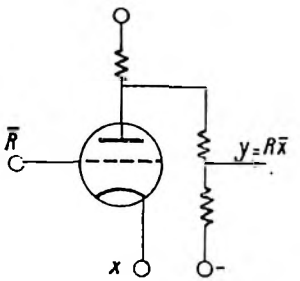


FIG. 68. — Autre forme de circuit de complémentation.

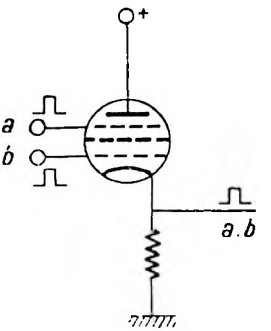


FIG. 69. — Circuit d'intersection à pentode.

et 68 peut encore être remplacé par un couplage par transformateur. Comme nous l'avons déjà signalé antérieurement (*fig. 47*), le signal de sortie est alors mis en forme au moyen de diodes. En outre, nous verrons dans un instant (*fig. 73*) que la fonction de complémentation peut avantageusement être combinée avec la régénération des impulsions.

Les autres fonctions logiques peuvent également être réalisées au moyen de tubes électroniques. Ainsi, la pentode de la *figure 69* constitue un intersecteur; au repos, la grille de commande et la grille suppresseuse sont polarisées négativement; le courant anodique ne peut prendre naissance que si elles sont simultanément portées à un potentiel

suffisamment positif. De même, les trois triodes de la *figure 70* sont montées en mélangeur, puisqu'il suffit que l'une d'elles reçoive une impulsion positive sur sa grille de commande pour que la résistance cathodique commune soit parcourue par un courant et qu'une impulsion positive apparaisse à la sortie. Nous n'insisterons pas davantage sur ces circuits, dont l'intérêt devient rapidement rétrospectif.

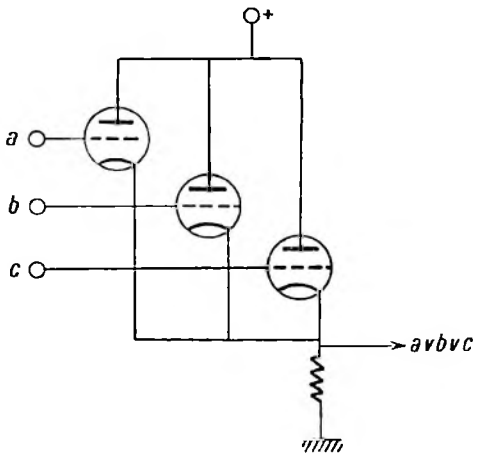


FIG. 70. — Circuit de réunion à triodes.

RÉGÉNÉRATEURS D'IMPULSIONS. — En parcourant les réseaux de diodes étudiés ci-dessus, ou les lignes de retard dont il sera question plus loin, les impulsions se détériorent par *affaiblissement* et *distorsion*. Lorsque leur forme devient incompatible avec le bon fonctionnement des circuits, il est donc nécessaire de les « *régénérer* », c'est-à-dire en réalité de leur substituer des impulsions « neuves » en provenance directe de l'horloge.

L'opération de régénération est schématisée sur la *figure 71 : X* représente le signal à régénérer, et *R* les signaux de rythme

engendrés par l'horloge; S est le signal régénéré. Ce fonctionnement est conforme à l'équation logique :

$$S = X \cdot R \vee S \cdot R.$$

Le premier terme traduit le déclenchement de la régénération par coïncidence du signal à régénérer avec une impul-

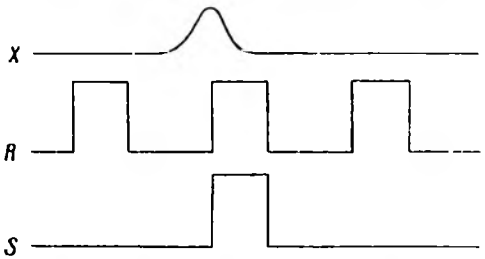


FIG. 71. — Régénération d'une impulsion.

sion de rythme; le second traduit l'*entretien* du signal régénéré jusqu'à la fin de l'impulsion de rythme en question. Cette équation se simplifie sous la forme suivante :

$$S = R \cdot (X \vee S).$$

Le circuit correspondant est représenté sur la *figure 72*. L'amplificateur A isole le réseau de régénération des circuits

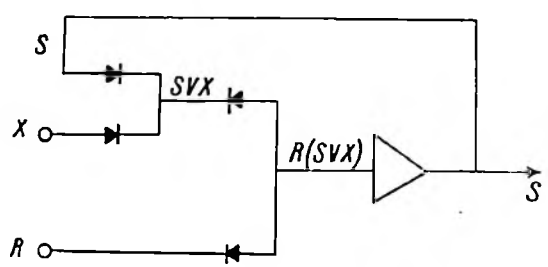


FIG. 72. — Schéma de principe d'un régénérateur à diodes.

alimentés par le signal régénéré. Il permet en outre facilement, s'il est muni de deux sorties symétriques, d'engendrer simultanément le signal S et son complément.

La *figure 73* représente, à titre d'exemple, un circuit combiné de régénération et de complémentation à couplage en courant alternatif. On économise deux diodes en confiant la fonction d'intersection au tube amplificateur lui-même,

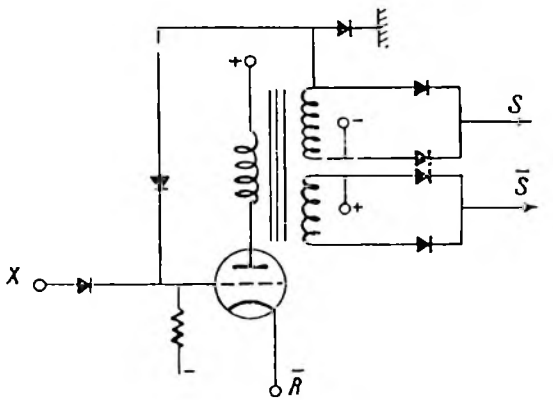


FIG. 73. — Exemple de régénérateur à liaison par transformateur.

qui reçoit sur sa grille de commande le signal $(X \vee S)$ et sur sa cathode le complément $\bar{R}$ du signal d'horloge. Enfin, si le signal régénéré est destiné à être transporté à une certaine distance, on sortira sur des enroulements à basse impédance et on reportera les diodes de mise en forme à l'autre extrémité de la liaison, dans le secondaire d'un transformateur élévateur de tension.

L'examen de la *figure 71* montre que le signal à régénérer doit être légèrement en avance sur l'impulsion de rythme devant servir à le régénérer. En raison du retard qu'il a nécessairement subi pendant son parcours à travers les circuits précédant le régénérateur, il ne saurait donc être

régénéré avec sa phase initiale. En règle générale, il doit même être retardé artificiellement de manière que son retard total soit un peu inférieur à une période du rythme fondamental. Les impulsions régénérées sont donc retardées exactement d'une période de rythme par rapport aux impulsions fraîches appliquées au circuit précédant le régénérateur. C'est pour réduire ce retard que certaines machines sont dotées de plusieurs rythmes décalés l'un par rapport à l'autre.

Cellules de mémoires à multivibrateur bistable.

Comme nous l'avons déjà signalé, une cellule de mémoire ou mémoire unitaire est un organe doué de deux états stables auxquels on attribue arbitrairement les valeurs

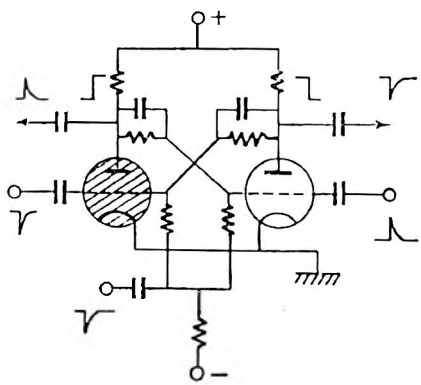


FIG. 74. — Multivibrateur bistable (bascule électronique).

binaires 0 et 1 respectivement, et susceptible de changer d'état sous l'effet d'un signal de commande convenable.

Nous verrons ultérieurement que des mémoires unitaires peuvent être réalisées au moyen de lignes à retard et de tores magnétiques. Contentons-nous pour, le moment, d'étudier le fonctionnement et quelques applications du *multi-*

vibrateur bistable ou bloqué, appelé communément « *basculeur* » ou « *bascule* » électronique.

C'est un montage à deux tubes (ou un tube double) (*fig. 74*), doué de deux états stables, dans chacun desquels l'un des tubes est bloqué, cependant que l'autre conduit un certain courant anodique fonction des éléments du

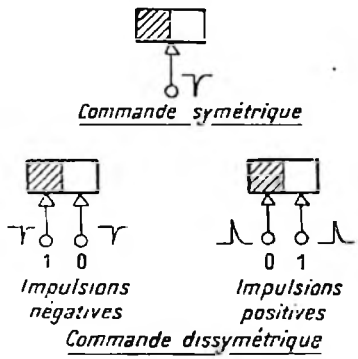


FIG. 75. — Représentation schématique d'un multivibrateur bistable et de ses divers modes de commande.

montage. Dans nos schémas, nous indiquons l'état conducteur d'un tube par des hachures. On peut changer l'état du système ou, comme on dit, le faire « basculer », en appliquant une impulsion négative à la grille du tube conducteur ou, au contraire, une impulsion positive à la grille du tube bloqué (*commande dissymétrique*), soit encore en appliquant une impulsion négative aux deux grilles ou aux deux plaques (*commande symétrique*). Ces deux modes de commande sont représentés schématiquement sur la *figure 75 ;* dans le mode d'attaque dissymétrique, l'impulsion 1 provoque le passage de l'état 0, représenté sur la figure, dans l'état 1, tandis que l'impulsion 0 entraîne la remise à zéro. Quel que soit le mode de commande, on recueille, au moment du basculement, un échelon négatif de tension sur la plaque du tube

préalablement bloqué, et un échelon positif sur celle du tube préalablement conducteur; ces échelons peuvent être transformés en impulsions exponentielles au moyen d'un réseau différentiateur résistance-capacité, comme nous avons déjà eu l'occasion de l'expliquer (*fig.* 29).

Dans le cas de la commande dissymétrique, les condensateurs de couplage ne sont pas strictement indispensable : leur rôle est seulement d'accélérer le processus de basculement en fournissant un chemin de basse impédance aux composantes à haute fréquence des signaux transmis des plaques vers les grilles. Par contre, dans le mode de commande symétrique, leur présence est essentielle. En effet, dans les instants suivant immédiatement l'application d'une impulsion négative aux deux grilles de commande, les deux tubes se bloquent : le système se trouve alors dans un état symétrique — instable il est vrai — mais il n'aurait aucune raison de basculer dans un sens plutôt que dans l'autre si les condensateurs de couplage n'avaient en quelque sorte conservé la mémoire de l'état précédent. Considérons, en effet, une bascule dans l'état où elle est représentée sur la *figure 74*, le tube de gauche étant conducteur. Les tensions de grilles ne différant l'une de l'autre que de quelques volts, on voit que le condensateur de droite est chargé à une tension plus élevée que celui de gauche d'une quantité approximativement égale à la chute ohmique dans la résistance de charge du tube de gauche. Considérons donc l'état dans lequel, juste après l'application de l'impulsion de commande, les deux tubes se trouvent bloqués et, par suite, les deux plaques se trouvent portées à la haute tension. La charge des condensateurs n'a pas encore eu le temps de s'écouler et la grille du tube de droite se trouve donc portée à une tension positive par rapport à celle du tube de gauche. C'est cette dissymétrie, introduite par le jeu des condensateurs de couplage, qui oriente le basculement vers l'état symétrique de l'état précédent.

Dans le cas d'une commande symétrique, il est possible de faire en sorte que la bascule réponde à des impulsions négatives appliquées symétriquement aux deux grilles, mais soit complètement insensible à des impulsions positives.

Cette faculté de discrimination peut être rendue plus sûre en injectant les impulsions de commande par l'intermédiaire de diodes (*fig. 76*) qui remplissent un double rôle : d'une part, elles ne laissent pas passer les impulsions positives; d'autre part, elles aiguillent les impulsions négatives vers la grille du tube conducteur.

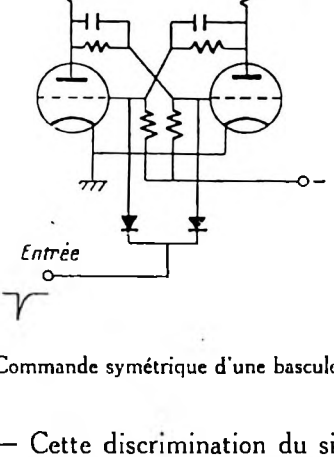


FIG. 76. — Commande symétrique d'une bascule par diodes.

COMPTEURS. — Cette discrimination du signe des impulsions de commande permet de construire un *compteur binaire* par le montage en cascade d'un certain nombre de bascules (*fig. 77*), dont chacune est attaquée symétriquement par les impulsions en provenance de l'une des plaques de la bascule précédente. Par exemple, la tension de la plaque de gauche de la bascule B_0 est un signal en créneaux de fréquence moitié de celle des impulsions de comptage. La différenciation de ce signal par le réseau RC de couplage fournit un train d'impulsions alternativement positives et négatives. Les impulsions positives étant sans effet sur la bascule B_1, cette dernière basculera donc une fois pour deux basculements de B_0, et ainsi de suite.

On peut de même constituer un système à n états stables,

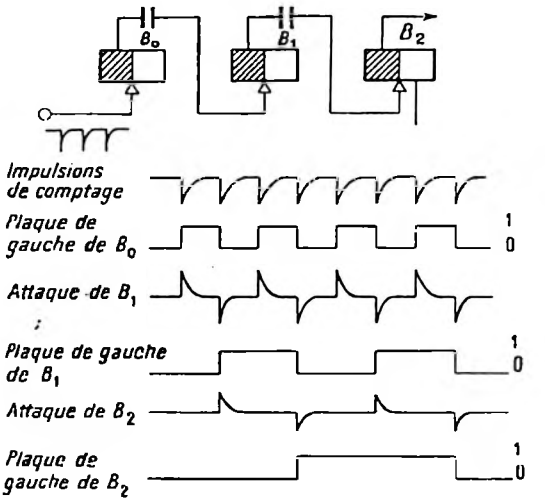


FIG. 77. — Compteur binaire.

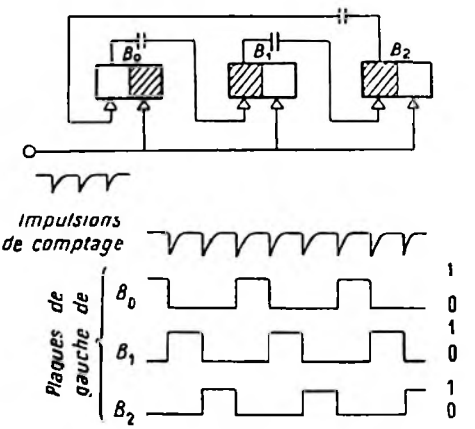


FIG. 78. — Anneau de 3.

ou *échelle de n*, en montant en anneau n bascules à commande dissymétrique (*fig. 78*). Dans l'état représenté sur la figure, seule la bascule B_0 est sensible aux impulsions négatives de comptage. Elle bascule donc sous l'effet de la première impulsion et, ce faisant, entraîne le basculement de B_1 en envoyant une impulsion négative à la grille de son tube de gauche, initialement conducteur. A ce moment, l'état de l'anneau a donc progressé d'un tiers de tour vers la droite, B_1 jouant maintenant le rôle que jouait initialement B_0, et ainsi de suite. Il est donc facile de constituer une échelle de 10, ou *décade*, à dix états stables, en commandant une quinte ou échelle de 5 par l'intermédiaire d'un étage binaire à commande symétrique.

REGISTRES. — Montrons maintenant comment les bascules peuvent être utilisées pour constituer des *registres*. Un registre à n chiffres binaires sera constitué par n bascules

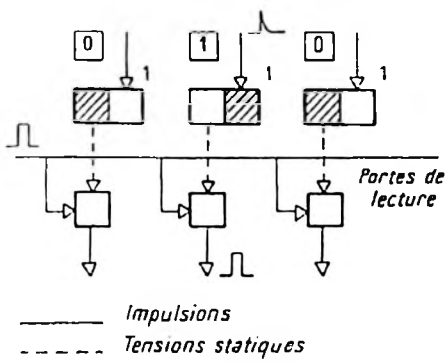


FIG. 79. — Registre à bascules à fonctionnement parallèle.

(*fig. 79*). Un tel registre peut émettre son contenu en numération parallèle au moyen de n portes connectées d'une part aux sorties « 1 » des bascules (anodes des tubes bloqués dans l'état 1) et recevant, d'autre part, une impulsion de lecture. Pour inscrire un nombre transféré en numération

parallèle, il suffit d'appliquer les impulsions correspondantes aux entrées « 1 » des bascules.

Même dans une machine à fonctionnement série, il est quelquefois nécessaire d'inscrire un nombre de N chiffres représenté par un train de N impulsions dans un registre

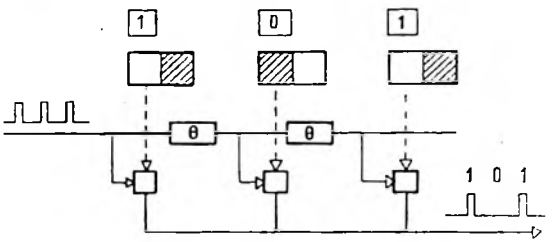


FIG. 80. — Inscription dans un registre à bascules d'un nombre transmis en numération série.

parallèle, par exemple dans le but de commander les organes d'un sélecteur ou d'un décodeur. A cet effet, on envoie le train de N impulsions dans une ligne à retard constituée par $(N - 1)$ cellules introduisant chacune un retard égal à une période d'impulsion 0 (*fig. 80*). Lorsque la première impulsion du train sort de la dernière cellule de retard, les impulsions suivantes émergent de même des cellules précédentes et il suffit d'appliquer une impulsion de commande

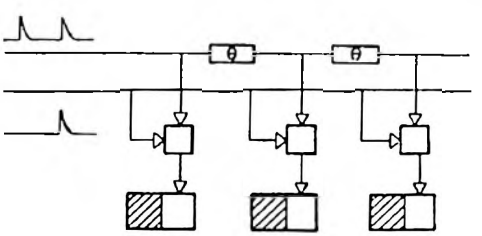
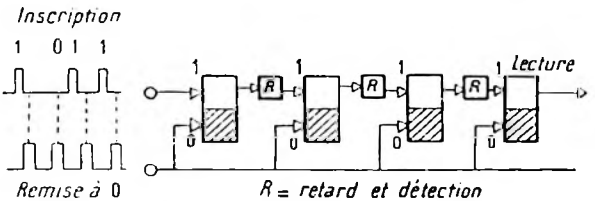


FIG. 81. — Lecture série du contenu d'un registre à bascules.

à N portes pour qu'elles transmettent ces impulsions en parallèle aux entrées « 1 » des N bascules du registre. Inversement, l'information ainsi emmagasinée en parallèle peut être reconvertie dans la représentation série en ouvrant successivement les portes au moyen d'un train d'impulsions de commande et d'une ligne à retard (*fig. 81*).

Les opérations de « staticisation » et de « dynamisation »



Inscription					
	$t = 0$	1	0	0	0
	$t = \theta$	1	1	0	0
	$t = 2\theta$	0	1	1	0
	$t = 3\theta$	1	0	1	1

FIG. 82. — Registre à décalage.

— comme on dit quelquefois — décrites au paragraphe précédent, peuvent également être effectuées au moyen du *registre à décalage* de la *figure 82*. Il suffit de réunir la sortie « 1 » de chaque bascule à l'entrée « 1 » de la bascule suivante, à travers un élément de retard et de détection ne transmettant que des impulsions de même signe — positif dans le cas de la figure. Si une impulsion de remise à zéro est appliquée à toutes les bascules, celles d'entre elles se trouvant préalablement dans l'état 1 provoquent le basculement de la bascule suivante, au cas où cette dernière se trouvait dans l'état 0; ainsi, la configuration de 1 et de 0 inscrite dans le registre se trouve décalée d'un rang vers la droite après chaque impulsion de remise à zéro. Une suite de N impulsions de remise à zéro entraîne donc l'émission par la sortie

« lecture » du contenu du registre sous la forme d'un train d'impulsions, ainsi que la remise à zéro de celui-ci. Inversement, si l'on applique à l'entrée « écriture » un train de N impulsions représentant un nombre binaire de N chiffres, séparées par N impulsions de remise à zéro, le nombre en question se trouve inscrit sous forme statique sur le registre. Un tel dispositif peut donc servir, d'une part, à convertir

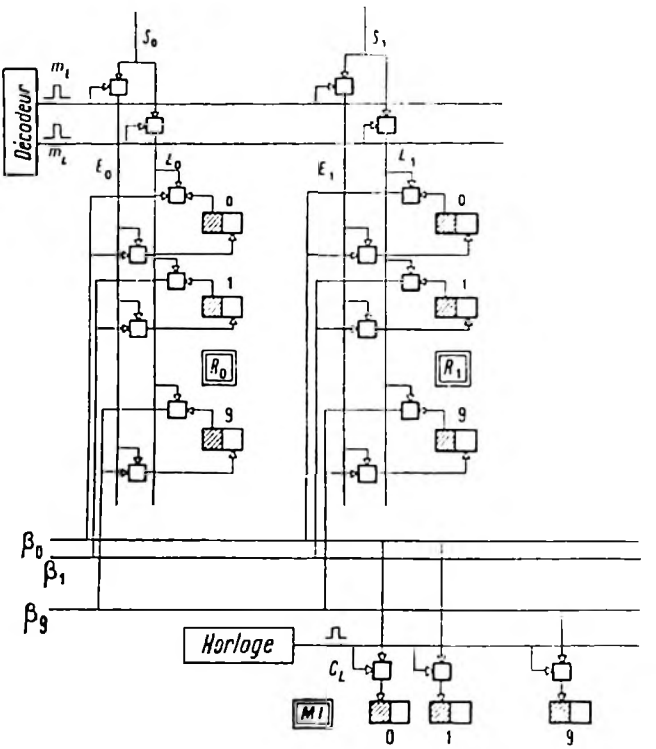


FIG. 83. — Transfert du contenu d'une mémoire à bascules dans le registre de la mémoire d'instruction.

un nombre de la forme série à la forme parallèle et inversement, d'autre part, à modifier la cadence d'un train d'impulsions : il suffit pour cela de le relire à une cadence différente de sa cadence d'écriture.

ORGANISATION D'UNE MÉMOIRE PARALLÈLE A BASCULES. — La *figure 83* n'est pas autre chose que la version électronique du schéma à relais de la *figure* 21. Soit, par exemple, à transmettre, au cours de la phase Φ_1, le contenu du registre R_0 dans la mémoire d'instruction MI. Sous l'effet du compteur ordinal, la sortie s_0 du sélecteur de mémoire se trouve portée au niveau « + », toutes les autres sorties restant au niveau « — ». L'application d'une impulsion m_L aux portes de commande de lecture provoque donc uniquement l'ouverture des portes de lecture des bascules du registre R_0 et l'émission en parallèle du contenu de ce registre sur les barres β. Si, à ce moment, les portes d'écriture de la mémoire d'instruction sont ouvertes par une impulsion c_L, le nombre lu dans R_0 s'inscrit dans MI. Les impulsions m_L et c_L jouent

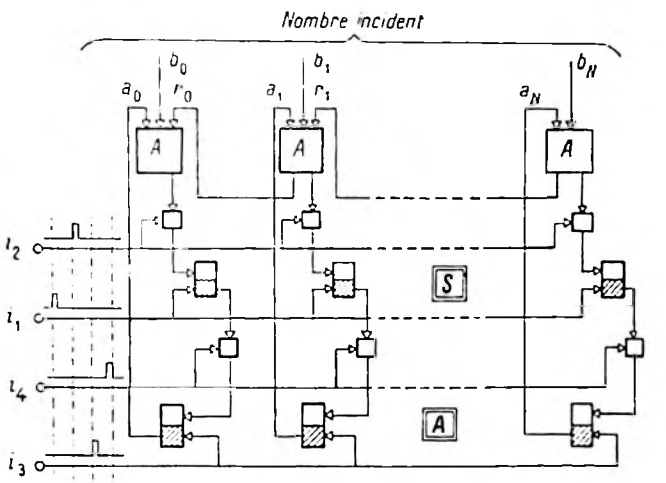


FIG. 84. — Totalisateur binaire.

donc exactement le même rôle que les courants d'excitation des relais de mêmes noms de la machine à relais étudiée précédemment, dont le fonctionnement ne diffère donc pas sensiblement — à la cadence près — de celui d'une calculatrice électronique parallèle. Les contacts y sont simplement remplacés par des portes commandées par des impulsions en provenance de l'horloge, soit directement, soit par l'intermédiaire du décodeur d'instructions.

TOTALISATEUR PARALLÈLE. — Ce type de transfert est mis en œuvre dans le *totalisateur parallèle* de la *figure 84*, dans lequel la somme du contenu du registre A et du nombre incident est emmagasinée provisoirement dans le registre S avant d'être substituée à l'ancien contenu de A. L'opération se fait en quatre temps. En premier lieu, le registre S est remis à zéro par l'impulsion i_1. Les chiffres de la somme fournis par les additionneurs y sont alors transférés par ouverture des portes correspondantes sous l'effet de l'impulsion i_2. A ce moment, le registre A est remis à zéro par l'impulsion i_3, puis reçoit le contenu de S à travers les portes ouvertes par l'impulsion i_4.

AUTRES TYPES DE TOTALISATEURS PARALLÈLES. — Le compteur binaire de la *figure 77* peut facilement être transformé en totalisateur en munissant chaque étage d'une entrée indépendante, de la manière indiquée sur la *figure 85*. Les

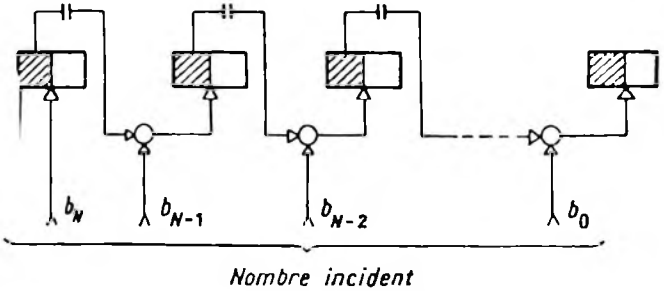


FIG. 85. — Transformation d'un compteur binaire en totalisateur; les chiffres du nombre incident sont appliqués successivement.

chiffres successifs du nombre à additionner au contenu du registre constitué par les bascules sont appliqués *successivement* aux entrées correspondantes sous la forme d'impulsions convenables. Après l'application de chaque impulsion, doit s'écouler un temps suffisant pour permettre la propagation de la retenue, dans le cas le plus défavorable jusqu'à l'extrémité de plus fort poids du registre :

$$\begin{array}{r} \textcircled{1}\,\textcircled{1}\,\textcircled{1} \\ 1\ 1\ 1\ 1 \\ +\qquad\ \ 1 \\ \hline 1\ 0\ 0\ 0\ 0 \end{array}$$

La grande simplicité de ce type de totalisateur — payée par sa lenteur — provient du fait que chaque bascule remplit en fait deux fonctions distinctes : d'une part, elle enregistre automatiquement la somme modulo 2 des impulsions qui lui sont appliquées; d'autre part, elle émet une impulsion de retenue quand son état passe de 1 à 0.

Voyons maintenant comment compléter — et compliquer — ce circuit pour assurer la propagation simultanée des retenues (*fig. 86*). Dans ce circuit, les chiffres *b* du nombre

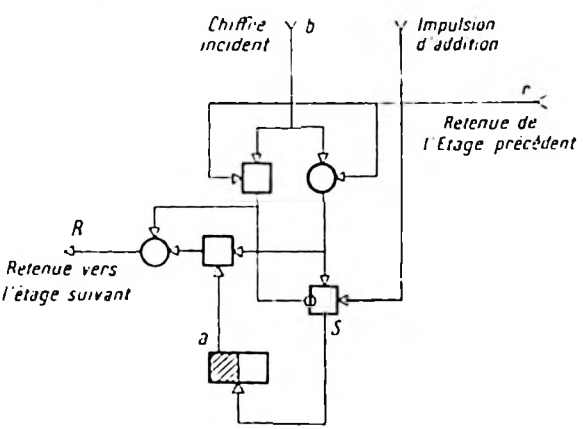


FIG. 86. — Étage de totalisation binaire à circuit logique de retenue.

incident à totaliser sont présentés sous forme statique à un circuit logique qui forme, d'une part la somme modulo 2 de b et de la retenue r en provenance de l'étage précédent, d'autre part, la retenue des 3 chiffres a, b et r, sous la forme :

$$R = b\ r \vee a\ (b \vee r).$$

Après un temps suffisant pour que, dans le cas le plus défavorable, une retenue ait eu le temps de se propager d'un bout à l'autre du registre, une impulsion d'addition est appliquée à la porte P_A, qui transmet alors une impulsion à l'entrée de commande symétrique de la bascule au cas où la somme modulo 2 de b et de r est égale à 1. On notera qu'en ce qui concerne la propagation des retenues, ce circuit est tout à fait analogue à l'additionneur à relais de la *figure 10*.

Lignes à retard; mémoires à circulation.

La *figure 87* représente le fonctionnement logique d'une ligne à retard permettant de mémoriser en numération série un nombre binaire de N chiffres. La ligne proprement dite fait subir aux impulsions qui la parcourent un retard total légèrement inférieur à $N\theta$, θ désignant la période fondamentale des impulsions de rythme. Les impulsions

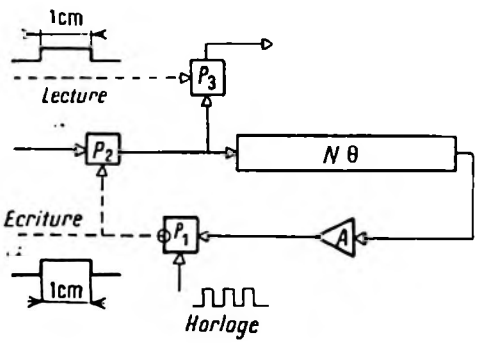


FIG. 87. — Schéma de principe d'une mémoire à circulation.

sortantes sont régénérées selon le processus étudié ci-dessus, puis réinjectées à l'entrée de la ligne. Un tel dispositif constitue donc un registre dynamique, dans lequel le train d'impulsions représentant le nombre inscrit circule indéfiniment en circuit fermé. Pour y inscrire un nombre, il faut ouvrir la porte P_2 et fermer simultanément la porte P_1 du régénérateur pendant la durée d'un cycle mineur. Le nombre entrant se substitue alors au contenu précédent. La lecture du contenu du registre, sans effacement, résulte de l'ouverture de la porte P_3 pendant un cycle mineur.

Si, comme c'est le cas en général, la première impulsion du train représente le chiffre de plus petit poids, l'introduction d'un retard supplémentaire θ a pour effet de décaler le contenu du registre d'un rang vers les forts poids.

Si une même ligne à retard contient plusieurs nombres, son temps de parcours total est appelé *cycle majeur*. Les portes d'entrée et de sortie doivent alors être ouvertes seulement pendant le cycle mineur correspondant au nombre à inscrire ou à lire. La position de ce cycle mineur dans le

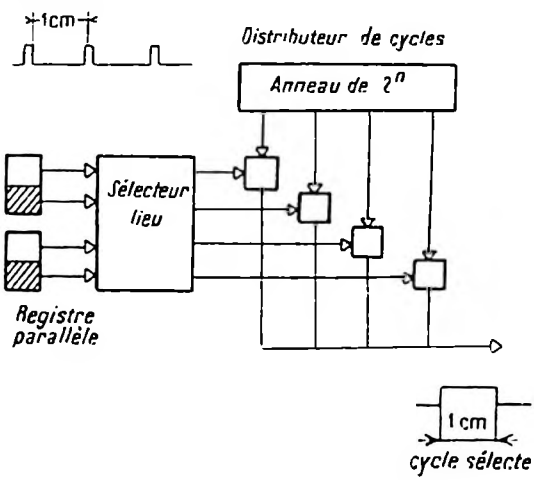


FIG. 88. — Sélecteur de cycle mineur faisant usage d'un sélecteur-lieu.

cycle majeur constitue l' « *adresse-temps* » du nombre correspondant. Si une mémoire est constituée par un ensemble de lignes à retard identiques, le numéro de la ligne contenant un nombre donné constitue ce que l'on appelle son « *adresse-lieu* ».

La sélection d'une adresse dans une mémoire constituée par une batterie de lignes à retard comporte donc deux opérations distinctes : la sélection d'une des lignes ou *sélection-lieu*, que nous avons déjà étudiée, et la sélection d'un cycle mineur dans le cycle majeur de cette ligne ou *sélection-temps*. Cette dernière opération peut se faire, par exemple, de la manière indiquée sur la *figure 88*. Le numéro d'ordre du cycle mineur à sélecteur est inscrit sur un registre parallèle à n bascules commandant un sélecteur-lieu. Ce dernier électe l'une des 2^n sorties d'un distributeur de cycles mineurs onstitué en principe par un compteur en anneau à 2^n positions attaqué par des impulsions dites de cycles mineurs intercalées entre les cycles mineurs successifs. On recueille donc à la sortie un signal demeurant au niveau « + » pendant la durée du cycle mineur sélecté, signal qu'il suffit d'appliquer à la porte de lecture ou à la porte d'écriture de la ligne sélectée, selon les indications du décodeur d'instructions.

Une autre méthode de sélection-temps est schématisée sur la *figure 89*. Ici, le numéro du cycle mineur à sélecter est inscrit dans un registre dynamique constitué par une ligne à retard. Une autre ligne est organisée en compteur de cycles mineurs, comme nous le verrons plus loin : son contenu est égal à chaque instant au numéro du cycle mineur suivant. L'égalité entre les contenus des deux lignes est

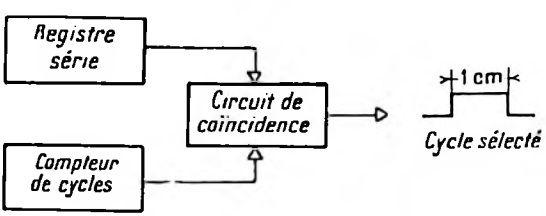


FIG. 89. — Sélecteur de cycle mineur à compteur et circuit de coïncidence.

repérée par un circuit de coïncidence qui engendre alors un signal de porte demeurant au niveau « + » pendant toute la durée du cycle mineur suivant. Ce fonctionnement est illustré sur la *figure 90* sur l'exemple de l'adresse-temps 7;

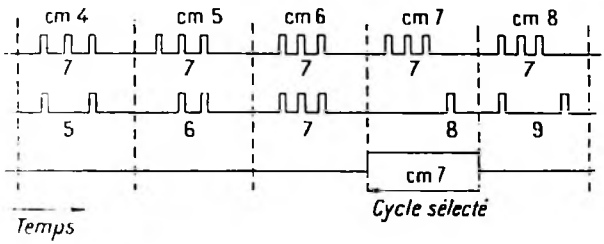


FIG. 90. — Schéma de fonctionnement du sélecteur de cycle mineur de la *figure 89*.

n'oublions pas que les premiers chiffres d'un train d'impulsions sont ceux de faibles poids.

Une mémoire à circulation douée d'un retard égal à une période fondamentale 0 constitue une mémoire unitaire ou cellule de mémoire (*fig. 91*). Une impulsion unique y circule en circuit fermé si elle est dans l'état 1. Pour la mettre dans l'état 0, on applique une impulsion d'inhibition à la porte *P* de régénération. Pour la mettre dans l'état 1 si elle se trouve dans l'état 0, on applique une impulsion au mélangeur *M* qui reçoit sur son autre entrée l'impulsion retardée à régénérer. Un ensemble de *N* mémoires unitaires constitue un registre parallèle équivalent à un ensemble de *N* bascules,

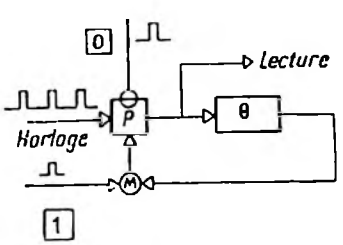


FIG. 91. — Mémoire unitaire à circulation.

à cette différence près que son information n'est pas disponible en permanence, mais seulement aux instants de test définis par les signaux d'horloge.

Du point de vue technologique, les mémoires à circulation diffèrent par la nature de leur ligne à retard. Les *lignes électriques* à paramètres localisés (*fig. 92*) ou répartis ne sont

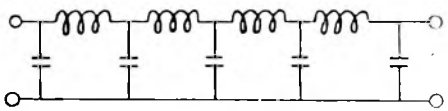


FIG. 92. — Ligne à retard électrique.

guère utilisées que comme registres série ou comme mémoires unitaires dynamiques. Elles présentent, en effet, un affaiblissement et une distorsion importants, ce qui oblige à régénérer les signaux, non seulement au moment du bouclage, mais encore en un certain nombre de points intermédiaires. Pour réaliser des mémoires à circulation de plus grande capacité, on a fait appel à la propagation d'ébranlements

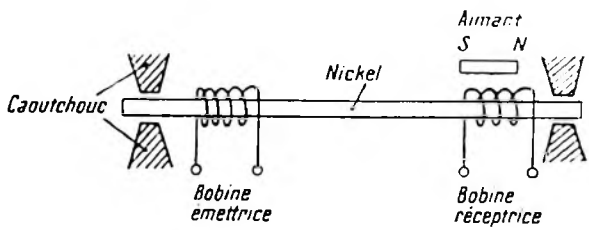


FIG. 93. — Ligne à retard à magnétostriction (nickel).

mécaniques dans des milieux solides ou liquides : nickel, quartz ou mercure. Dans le cas du *nickel*, on utilise l'effet de *magnéto-striction*, qui consiste en l'apparition d'une contrainte mécanique sous l'effet d'un champ magnétique,

pour engendrer une onde qui se propage dans un fil ou un tube de nickel; à l'autre extrémité de la ligne, l'effet inverse permet d'induire un signal électrique dans une bobine réceptrice soumise à un champ permanent de polarisation (*fig. 93*). De même, l'*effet piézo-électrique* permet de produire

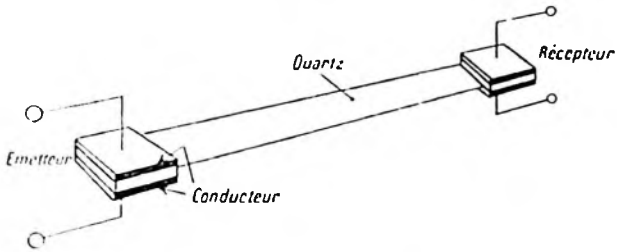


FIG. 94. — Ligne à retard piézo-électrique (quartz).

un ébranlement à une extrémité d'une barre de *quartz* au moyen d'un champ électrique bref, et l'effet inverse permet de recueillir une impulsion électrique à l'autre extrémité aux bornes d'un condensateur (*fig. 94*). Enfin, dans les lignes à retard à *mercure*, les impulsions à transmettre sont

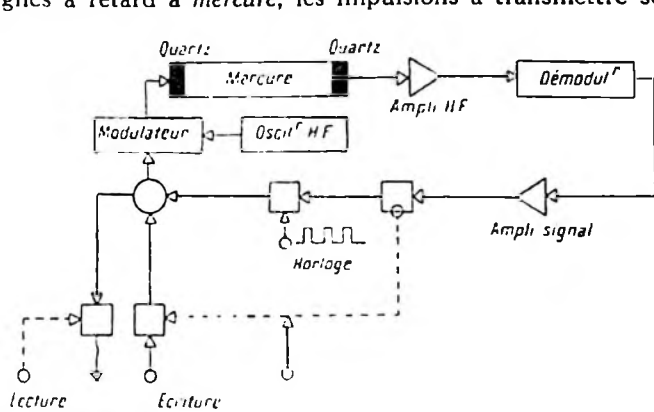


FIG. 95. — Ligne à retard à mercure.

d'abord modulées à une fréquence d'une dizaine de mégahertz avant d'être appliquées à un cristal de quartz qui crée dans la colonne de mercure des trains d'ultra-sons (*fig. 95*); le signal recueilli aux bornes du quartz récepteur est amplifié, puis démodulé, avant d'être régénéré. On peut ainsi réaliser des mémoires à circulation d'une capacité de quelques centaines de chiffres binaires.

On ne peut évidemment écrire ou lire un nombre dans une mémoire à circulation que pendant la durée du cycle mineur correspondant à son adresse-temps. Le *temps d'accès* moyen, égal à un demi-cycle majeur, est donc d'autant plus important que la capacité de la mémoire est plus élevée. C'est là un inconvénient que l'on rencontre également dans d'autres types de mémoires, tels que les tambours magnétiques, dont nous parlerons ultérieurement. On peut y remédier dans une certaine mesure par une répartition judicieuse des instructions et des nombres dans la mémoire, répartition telle que les informations se trouvent disponibles autant que possible au moment même où on en a besoin. Cette méthode, dite *programmation optimum*, exige toutefois que l'on dispose d'un code permettant de spécifier, dans chaque instruction, l'adresse de l'instruction suivante, ou d'une facilité similaire.

QUELQUES APPLICATIONS DES LIGNES A RETARD. — Les lignes à retard sont très utilisées dans les organes de calcul des machines à fonctionnement série. Donnons-en quelques exemples caractéristiques.

Additionneurs binaires série (*fig. 96*). — Nous avons déjà vu qu'un étage d'addition binaire peut être constitué, soit par un additionneur à trois entrées, soit par deux demi-additionneurs. Pour effectuer l'addition de deux nombres émis simultanément sous la forme de deux trains d'impulsions de poids croissants, il suffit de réinjecter la retenue sortante sur l'entrée « retenue », après l'avoir retardé d'une période d'impulsion 0 (*fig. 96 a* et *b*).

Totalisateur binaire série (*fig. 97*). — Le totalisateur série de la *figure 97* est constitué par un additionneur série et un

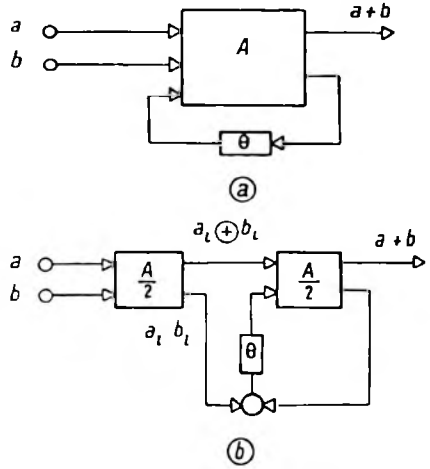
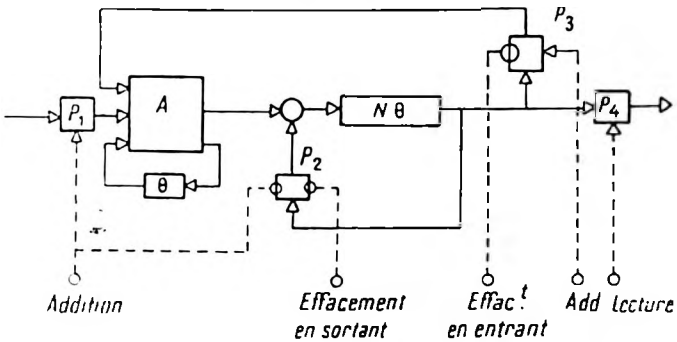


FIG. 96. — Additionneurs série.



	P_1	P_2	P_3	P_4
Addition	1	0	1	0
Remplacement	1	0	0	0
Lecture	0	1	0	1
Lecture avec effacement	0	0	0	1

FIG. 97. — Totalisateur série.

registre série (ligne à retard bouclée sur elle-même) dont le contenu peut être déversé dans l'additionneur à travers la porte P_3 en synchronisme avec le nombre incident à totaliser, qui rentre par la porte P_1. Le résultat de l'addition devant se substituer à l'ancien contenu du registre, la porte de régénération P_2 doit être fermée pendant le cycle mineur d'addition. On réalise l'opération de remplacement en maintenant fermé la porte P_3, auquel cas le nombre incident se substitue purement et simplement à l'ancien contenu du registre. Enfin, l'ouverture de P_4 pendant un cycle mineur permet de lire le contenu du registre sans l'effacer, tandis que la fermeture simultanée de P_2 entraîne la remise à zéro du registre. L'état ouvert ou fermé des quatre portes pour les quatre opérations possibles est résumé dans le tableau de la *figure 97.*

Extracteur de complément série. — On sait que l'opération de soustraction est équivalente à l'addition du *complément vrai.* Ce dernier résulte, d'autre part, de l'addition d'une

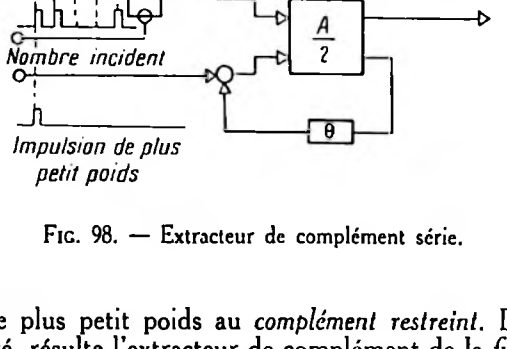


FIG. 98. — Extracteur de complément série.

unité de plus petit poids au *complément restreint.* De cette propriété, résulte l'extracteur de complément de la *figure 98.* Le complément restreint est obtenu, comme nous le savons, par intersection du signal complémentaire du nombre

incident avec les signaux d'horloge jouant le rôle d'impulsions de test. On lui ajoute une impulsion de plus petit poids, émise par les circuits de commande, au moyen d'un demi-additionneur dont la sortie est bouclée sur la seconde entrée à travers un élément de retard 0. Un additionneur à deux entrées est ici suffisant, puisqu'il ne peut exister

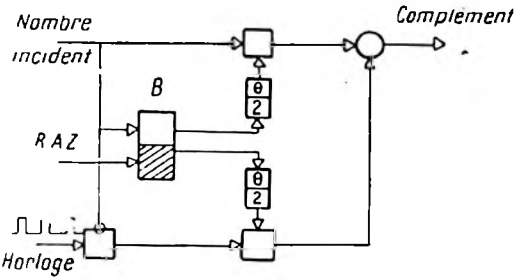


FIG. 99. — Autre forme d'extracteur de complément série.

simultanément qu'une impulsion du nombre incident et une retenue éventuelle.

Un autre type de circuit de complémentation est représenté sur la *figure 99*. La première impulsion du nombre incident, qui représente le premier chiffre de plus petit poids non nul, est transmise sans modification, mais provoque le basculement du basculeur. La porte P_1 se trouve alors fermée, tandis que la porte P_2 transmet vers la sortie le complément des chiffres suivants, engendré par la porte P_3 associée à un circuit de négation. A la fin de l'opération, la bascule est remise à zéro par une impulsion *RAZ*.

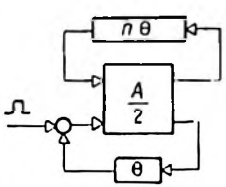


FIG. 100. — Compteur série à demi-additionneur.

Compteur série. — Le circuit de la *figure 100* permet d'ajouter une impulsion de plus petit poids au contenu du

registre à circulation constitué par la ligne à retard $n\,\theta$. Ici encore, l'addition peut être effectuée par un simple demi-additionneur. Ce circuit peut constituer, soit un *compteur ordinal*, si l'impulsion additionnée est émise après l'exécution de chaque instruction, soit un *compteur de cycles mineurs* si on lui applique des impulsions de cycles mineurs.

Multiplieur série. — L'organisation schématisée sur la *figure 101* est celle du multiplieur de la machine *Edsac* de l'Université de Cambridge. Le multiplicateur et le multiplicande, tous deux de N chiffres binaires, sont contenus dans les moitiés de faible poids de deux registres à circulation de période $2\,N$. Les chiffres du multiplicateur sont testés successivement, dans l'ordre des poids croissants, par une impulsion de test engendrée par les circuits de commande et progressant à chaque cycle mineur d'un rang vers les forts poids.

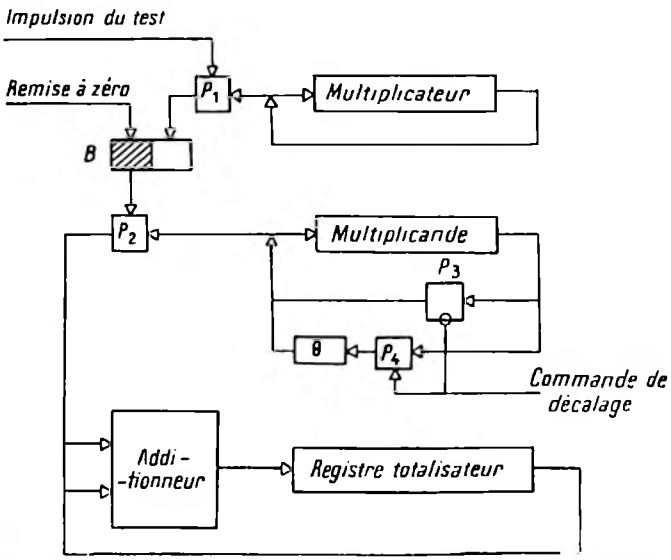


FIG. 101. — Multiplieur série à addition successive des produits partiels.

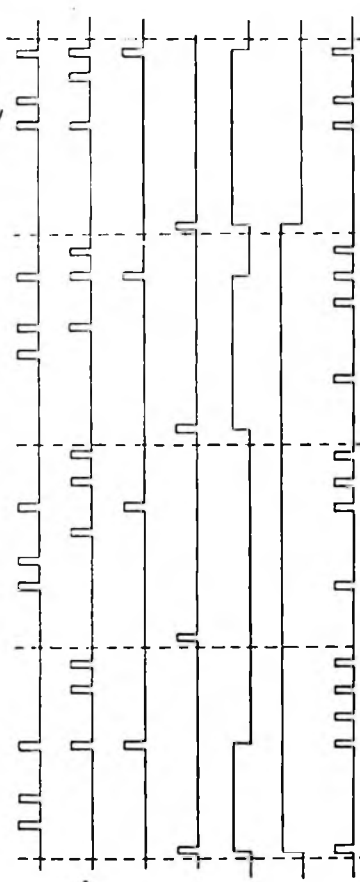


FIG. 102. — Diagramme de fonctionnement du multiplieur de la *figure 101*.

Si le chiffre du multiplicateur est un « 1 », la bascule B bascule et ouvre la porte P_2, à travers laquelle le multiplicande est transmis au totalisateur. Pendant ce temps, le multiplicande a été décalé d'un rang vers la gauche par introduction d'un retard 0 supplémentaire provenant de la fermeture de P_3 et de l'ouverture de P_4 pendant toute la durée de la multiplication, à l'exception du premier cycle. Bien entendu, la bascule B est remise à zéro à la fin de chaque cycle mineur. Les produits partiels se totalisent donc successivement, comme le montre clairement le schéma de la *figure 102* sur l'exemple du produit :

$$1101 \times 1011 = 10001111.$$

L'impulsion de test est engendrée par un registre à circulation de période N, dans lequel circule une impulsion

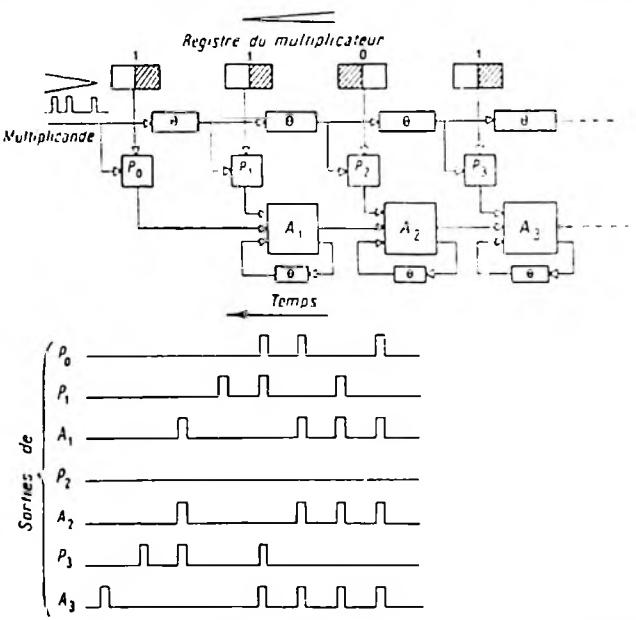


FIG. 103. — Multiplieur série à addition simultanée des produits partiels.

unique décalée d'un rang vers les forts poids à chaque cycle mineur, comme il a été indiqué plus haut pour le multiplicande. Après N cycles mineurs, cette impulsion se retrouve dans sa position initiale, ce qui indique que l'opération est terminée.

Multiplieurs série à additions simultanées. — La durée de la multiplication de deux nombres de N chiffres peut être réduite à $2\,N\,0$ en formant les produits partiels simultanément et en les totalisant au moyen de $(N - 1)$ additionneurs, comme le montre la *figure 103*. Le train d'impulsions du multiplicande est transmis ou non par les portes P, selon que le chiffre correspondant du multiplicateur, inscrit sur un registre statique, est un 1 ou un 0. Les produits partiels se trouvent automatiquement décalés d'un rang vers la gauche lorsque l'on passe d'une porte à la porte de poids immédiatement supérieur. Chaque produit partiel est additionné à la somme des produits partiels de poids inférieurs déjà partiellement formés. Mais il faut bien noter que ces diverses opérations se déroulent simultanément, comme le montre clairement le schéma de la *figure 103*, sur le même exemple que celui de la *figure 102*.

Circuits statomagnétiques.

Les matériaux magnétiques utilisés dans les organes des machines à calculer sont de deux types. Les *ferrites*, mélanges de cristaux mixtes de formules $M\,Fe_2\,O_4$, M étant un métal bivalent (Cu, Mg, Mn, Ni, Zn), sont caractérisées par une résistivité élevée et une faible induction rémanente; on les utilise sous la forme de tores dont le diamètre intérieur peut être inférieur à 1 mm, en particulier pour constituer des mémoires matricielles. D'autre part, certains *alliages spéciaux*, tels que le *Deltamax* américain et le *Rectimphy* français, possèdent une forte induction rémanente (15 000 gauss); ils se présentent sous la forme de bandes minces que l'on enroule sur des noyaux en céramique (1).

(1) Aux fréquences moyennes, jusqu'à quelques dizaines de kilohertz, les ferrites ont des pertes plus importantes que les alliages métalliques spéciaux; par contre, elles permettent d'atteindre des fréquences de quelques mégahertz.

Ces matériaux ont le caractère commun de présenter un cycle d'hystérésis d'allure rectangulaire, comme le montre la *figure 104*. Un noyau ainsi constitué peut donc se trouver, après magnétisation, dans l'un de deux états de magnétisation rémanente de signes contraires, auxquels on peut affecter arbitrairement les valeurs binaires 0 et 1. L'état

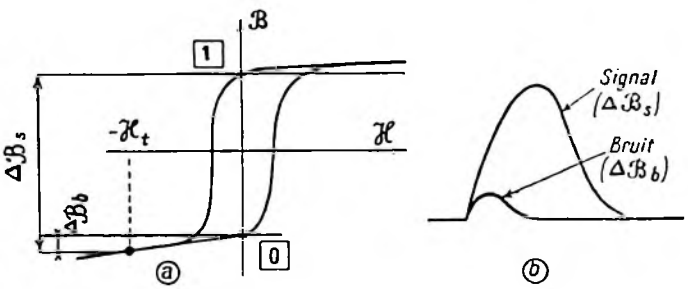


FIG. 104. — Principe de fonctionnement d'une cellule de mémoire statomagnétique.

d'un noyau peut être testé en appliquant, au moyen d'un enroulement approprié, une impulsion de champ — $\mathcal{H}_t$. On peut alors recueillir aux bornes d'un second enroulement une tension proportionnelle à la variation de flux. On recueillera donc un signal de faible amplitude (signal parasite ou de bruit) si le noyau testé se trouvait préalablement dans l'état 0 (variation d'induction $\Delta\ \mathcal{B}_b$), et un signal utile d'amplitude notablement supérieure s'il se trouvait dans l'état 1 (variation d'induction $\Delta\ \mathcal{B}_s$). Dans la suite, nous négligerons les signaux de bruit.

Considérons maintenant le circuit élémentaire de la *figure 105*. Des signaux d'horloge H sont appliqués à un premier enroulement. Le noyau étant initialement dans l'état 0, son point figuratif oscille périodiquement entre les points A et B. Supposons la diode branchée de telle manière

qu'une diminution de l'induction $\mathfrak{B}$ entraîne le passage d'un courant dans la charge R. Au cours du passage de A en B, on recueille donc aux bornes de R un signal de faible amplitude, que nous supposons négligeable.

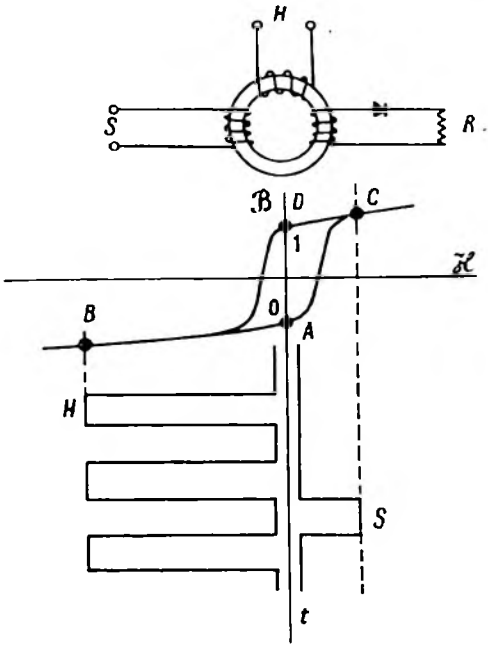


FIG. 105. — Un signal de sortie apparaît aux bornes de la résistance R lorsqu'un signal d'entrée S précède une des impulsions d'horloge H.

Supposons maintenant qu'entre deux impulsions d'horloge, nous appliquions à un second enroulement de commande un signal S de polarité magnétique opposée capable de faire passer le noyau de A en C, puis en D après la fin du signal S. Ce signal S n'engendre aucun signal de sortie aux bornes de R, par suite de la présence de la diode. Mais l'impulsion

d'horloge suivante fera décrire au noyau le trajet $D\,B\,A$ et entraînera l'apparition d'un signal utile aux bornes de R.

En résumé, si le noyau a été mis préalablement dans l'état 1

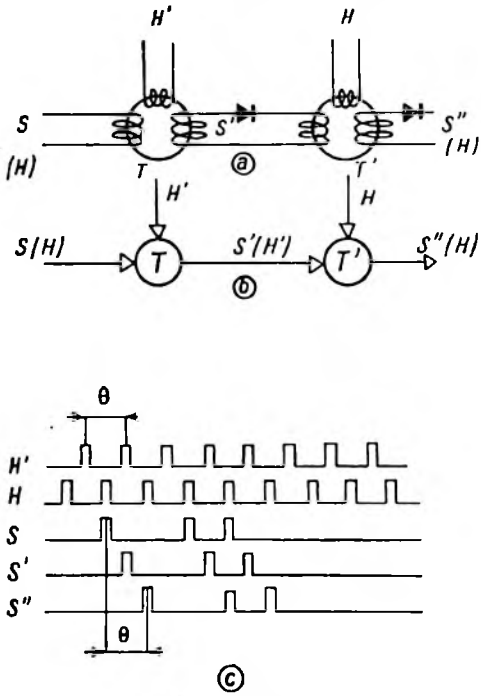


FIG. 106. — Transfert d'une information binaire d'un tore à un autre sous l'effet de deux signaux d'horloges entrelacés H et H'.

par un signal S, le signal d'horloge suivant entraîne l'apparition d'une impulsion dans le circuit de sortie. De plus, ce circuit constitue un amplificateur de puissance, et l'énergie transférée dans la charge, sous l'effet de l'énergie extérieure

apportée par les signaux d'horloge, peut être notablement supérieure à l'énergie apportée par le signal S. Ceci permet d'alimenter plusieurs noyaux au moyen du signal engendré par un noyau unique. De plus, la forme rectangulaire du cycle d'hystérésis permet de donner au signal de sortie une forme rectangulaire analogue à celle du signal d'entrée. Ce circuit constitue donc en fait un régénérateur d'impulsions.

TRANSFERT, LIGNES A RETARD STATOMAGNÉTIQUES. — Envisageons maintenant le *transfert* dans un noyau T' de l'information binaire inscrite dans un noyau T sous l'effet d'un signal S (*fig. 106*). Comme l'état d'un noyau ne peut être testé qu'après l'application du signal de commande, un tel transfert exige l'emploi de deux signaux d'horloge H et H' entrelacés. Dans ces conditions, l'application à T d'un signal S de rythme H engendre un signal S' de rythme H', qui crée à son tour un signal de sortie S'' de rythme H. Le signal de sortie S'' est donc de même rythme que le signal d'entrée S, mais il se trouve retardé d'une période de rythme 0. La *figure 106 b* montre la représentation schématique d'un tel circuit de transfert; chaque signal est accompagné de l'indication entre parenthèses de son rythme H ou H'.

L'association en cascade d'un certain nombre de ces circuits permet de constituer une *ligne à retard statomagnétique.*

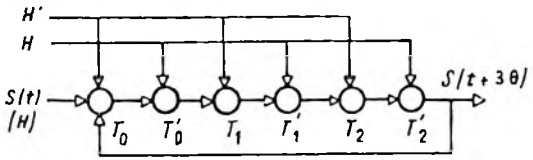


FIG. 107. — Mémoire à circulation statomagnétique.

Ainsi, la ligne à six noyaux de la *figure 107* introduit un retard de trois périodes de rythme. Si, de plus, cette ligne est bouclée sur elle-même, elle constitue une *mémoire à circulation* dans laquelle circule indéfiniment la configuration binaire initialement introduite en S.

Un cas particulier de mémoire statomagnétique à circulation est constitué par la *mémoire unitaire* de la *figure 108*, qui émet des impulsions d'horloge H à partir du moment où elle a reçu une impulsion de rythme H sur son entrée S_1.

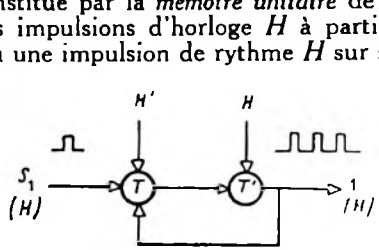


FIG. 108. — Cellule de mémoire à circulation statomagnétique.

CONSTITUTION DE CIRCUITS LOGIQUES A NOYAUX MAGNÉTIQUES. — Montrons maintenant comment les fonctions logiques peuvent être réalisées au moyen de noyaux magnétiques.

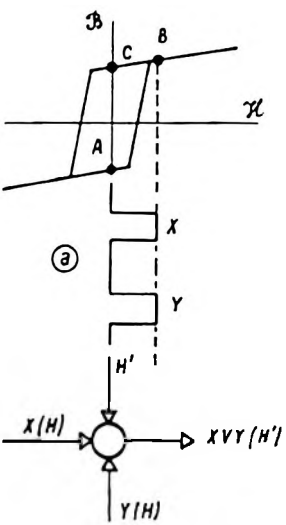


FIG. 109. — Circuit de réunion.

Réunion (*ou*) (*fig. 109*). — Si un noyau porte deux enroulements de commande tels que chacun des signaux X et Y agissant isolément soit capable de le porter de l'état 0 dans l'état 1, son signal de sortie constitue la réunion des deux signaux d'entrée (X ou Y ou les deux simultanément).

Intersection (*et*) (*fig. 110*). — Un noyau constituera un intersecteur si sa courbe d'hystérésis est telle qu'il soit possible de la faire passer de l'état 0 à l'état 1 sous l'effet des deux signaux X et Y appliqués simultanément, mais non sous

l'effet de l'un de ces signaux considéré seul. Cela est possible dans le cas des ferrites, mais ne l'est guère avec les alliages métalliques, dont l'emploi est préférable dans les circuits logiques à fréquence moyenne, en raison du gain plus élevé que permet leur forte induction rémanente.

C'est pourquoi l'opération d'intersection n'est guère utilisée dans les circuits logiques magnétostatiques; elle est remplacée par l'opération d'inhibition ($X\bar{Y}$), particulièrement aisée.

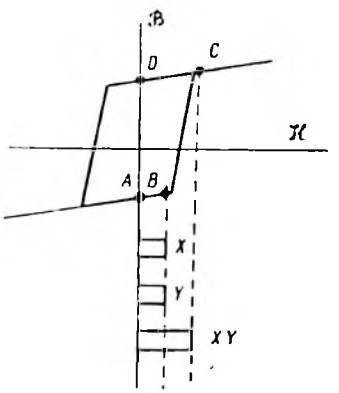


FIG. 110. — Circuit d'intersection.

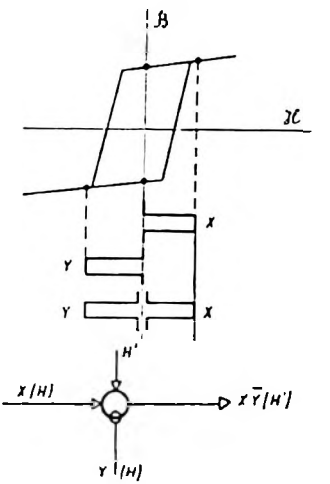


FIG. 111. — Circuit d'inhibition.

Inhibition (*fig. 111*). — L'action d'un signal X peut être annulée par l'application simultanée d'un signal Y de polarité opposée. On réalise ainsi la fonction $X\bar{Y}$.

En particulier, si le signal X est constitué par des signaux d'horloge H (*fig. 112*), le signal de sortie représente le complément restreint de Y dans le rythme H', qu'il est facile de convertir, si besoin est, dans le rythme H au moyen d'un second tore.

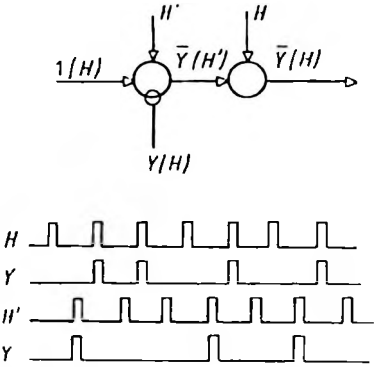
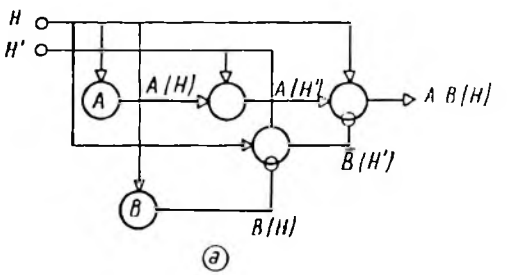


FIG. 112. — Circuit de complémentation.



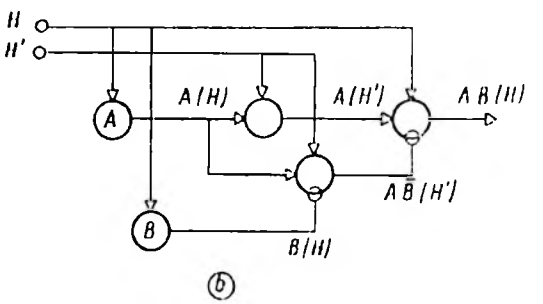


FIG. 113. — Circuits d'intersection ne faisant usage que de la réunion et de l'inhibition.

Circuits d'intersection. — La *figure 113* représente deux circuits d'intersection fondés sur l'emploi exclusif des fonctions de *réunion* et d'*inhibition*. Le premier traduit l'équation logique :

$$A \cdot \overline{(\bar{B})} = A \cdot B.$$

A cet effet, le signal A, de rythme H, retardé de 0, est inhibé par le signal B, lui-même obtenu par inhibition du signal d'horloge H' par le signal B de rythme H.

Le second circuit d'intersection de la *figure 113* est fondé sur l'équation logique :

$$A \cdot \overline{(A \cdot \bar{B})} = A \cdot B.$$

Son fonctionnement se comprend tout aussi aisément.

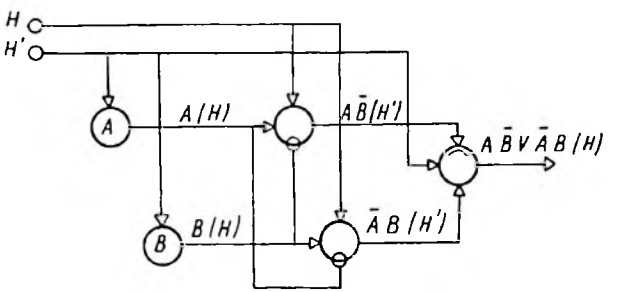


FIG. 114. — Circuit de dilemme.

Dilemme (ou exclusif) (fig. 114). — La facilité de l'opération d'inhibition donne une importance particulière aux circuits de dilemme :

$$A \oplus B = A \cdot \bar{B} \vee \bar{A} \cdot B.$$

La combinaison d'un tel circuit avec un circuit d'intersection du type de la *figure 113 a* donne immédiatement le demi-additionneur de la *figure 115*.

Montage en bascule ou mémoire unitaire (*fig. 116*). — L'adjonction au montage de la *figure 108* d'une entrée d'inhibition S_0 conduit au montage en bascule de la *figure 116*,

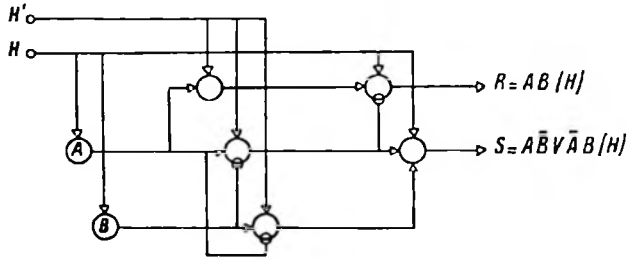
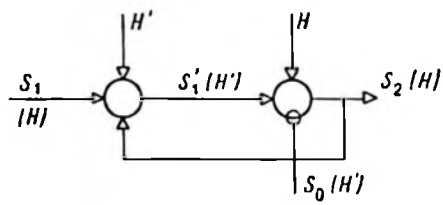


FIG. 115. — Demi-additionneur.



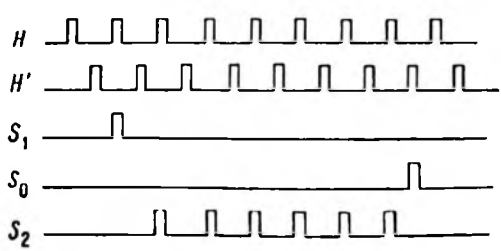


FIG. 116. — Générateur de « 1 ».

qui engendre des signaux H dès qu'il a été amorcé par une impulsion S_1 de rythme H et tant qu'il n'est pas désamorcé par une impulsion d'inhibition S_0 de rythme H'.

Générateur de complément (*fig. 117*). — Le signal de sortie du montage de la *figure 117* est constitué par le signal A, retardé de 2 0, si la bascule (encadrée par un trait mixte)

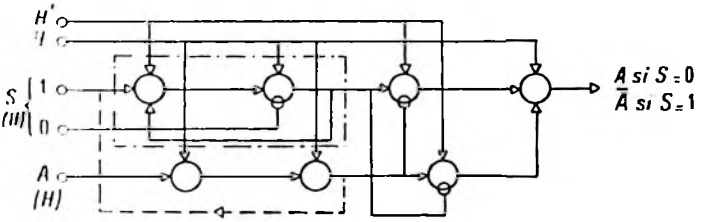


FIG. 117. — Extracteur de complément.

est dans l'état 0, et son *complément restreint* si la bascule est dans l'état 1. Amorçons maintenant la bascule, supposée initialement dans l'état 0, par la première impulsion (1) du

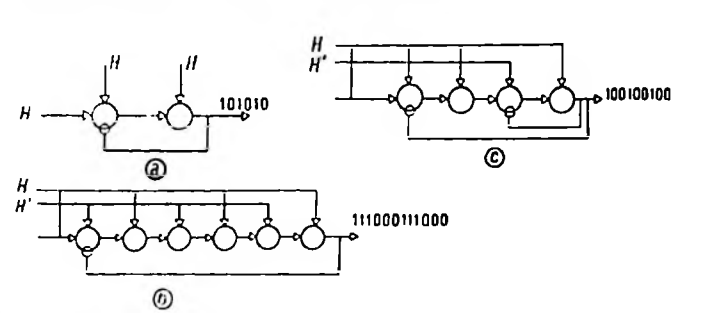


FIG. 118. — Générateurs de codes.

(1) Cette première impulsion représente le chiffre binaire de plus faible poids.

signal appliqué en A, au moyen de la liaison représentée en trait interrompu. Cette première impulsion est ainsi transmise telle quelle à la sortie, tandis que la suite du signal A se trouve remplacée par son complément restreint : le signal de sortie est donc le *complément vrai* de A.

Générateurs de signaux (*fig. 118*). — On peut voir, sur la *figure 118*, quelques exemples d'emploi de la fonction d'inhibition pour engendrer diverses configurations d'impulsions.

Mémoire matricielle a noyaux magnétiques. — Le fonctionnement des mémoires matricielles à tores de ferrite est fondé sur l'emploi du circuit d'intersection de la *figure 110*.

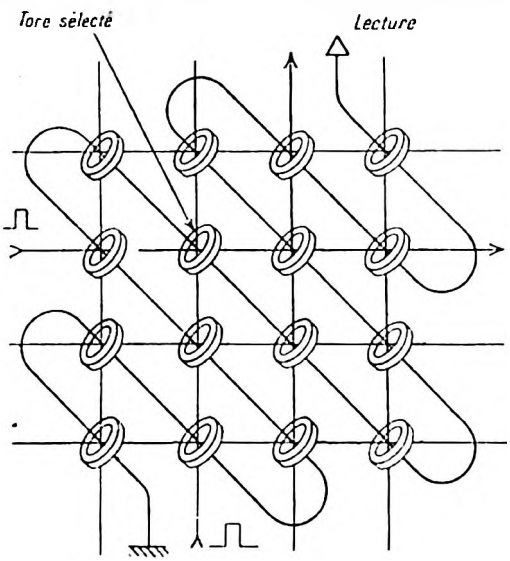


Fig. 119. — Mémoire matricielle à tores de ferrite.

Les tores, d'environ 2 mm de diamètre extérieur, sont disposés sous la forme d'une matrice carrée ou rectangulaire (*fig. 119*). Les enroulements sont constitués par de simples

fils traversant les tores et formant les lignes et les colonnes de la matrice. Supposons qu'initialement tous les tores soient dans l'état 0. L'application d'une impulsion positive à une ligne ou à une colonne est supposée insuffisante pour que les tores intéressés changent d'état. Par contre, l'application simultanée d'une impulsion positive à une ligne et à une colonne entraîne le changement d'état du tore placé au point de croisement, pour lequel les deux effets se cumulent (voir *fig. 110*).

Inversement, pour tester l'état d'un tore, on appliquera simultanément une impulsion négative aux deux fils qui s'y croisent. Si le tore est dans l'état 0, le fil de lecture qui traverse tous les tores en série sera le siège d'un signal de faible amplitude. Si, au contraire, le tore testé se trouve dans l'état 1, il passera dans l'état 0 et la variation de flux correspondante induira un signal notablement plus grand dans l'enroulement de lecture.

Malheureusement, ce processus de lecture détruit l'information lue, si le tore est dans l'état 1, information qui doit donc être rétablie immédiatement par des circuits annexes.

Des mesures doivent également être prises en vue de réduire l'influence des tensions parasites induites dans l'enroulement de lecture par les petites variations de flux qui se produisent dans les tores non sélectés, par suite de la forme imparfaitement rectangulaire de la boucle d'hystérésis. Il ne faut évidemment pas que le cumul de ces signaux parasites risque de produire un signal d'amplitude comparable à celle des signaux utiles. On peut, par exemple, faire passer l'enroulement de lecture à travers les tores alternativement dans un sens et dans l'autre, de manière que les signaux parasites s'annulent mutuellement en moyenne. Nous n'insisterons pas sur ces questions, dont l'importance technologique est cependant considérable.

On sait actuellement réaliser des matrices de quelques milliers de tores, douées d'un temps d'accès de l'ordre de la microseconde. Cependant, la nécessité de rétablir l'information initiale après lecture ne permet pas d'effectuer deux lectures successives séparées par un intervalle de temps inférieur à une dizaine de microsecondes. Les matrices de

tores se prêtent bien à la réalisation de mémoires parallèles. On utilise alors autant de matrices que de chiffres dans les nombres à traiter, les chiffres d'un même nombre occupant des positions homologues dans les différentes matrices. On envisage, dès maintenant, la possibilité de construire ainsi des mémoires à accès rapide d'une capacité d'un million de chiffres binaires.

Mémoires matricielles ferro-électriques.

Certains diélectriques, tels que le titanate de barium, présentent un phénomène d'hystérésis diélectrique analogue à l'hystérésis magnétique, le champ et la polarisation élec-

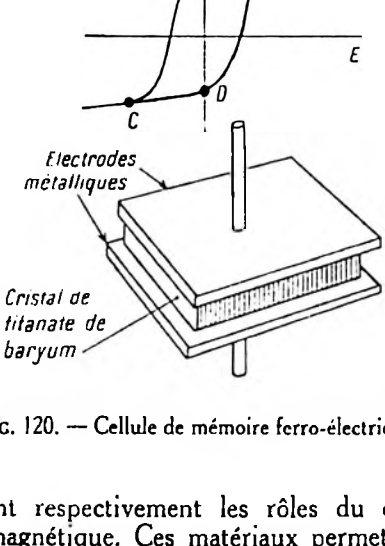


FIG. 120. — Cellule de mémoire ferro-électrique.

triques jouant respectivement les rôles du champ et de l'induction magnétique. Ces matériaux permettent donc de constituer des cellules de mémoire, de construction analogue à des condensateurs (*fig. 120*). Si la tension aux bornes est

amenée à une certaine valeur, puis réduite à zéro, une charge résiduelle subsiste dans le diélectrique, charge dont la polarité dépend du signe de la tension appliquée.

La lecture de l'information binaire contenue dans une telle cellule peut se faire au moyen du circuit de la *figure 121*.

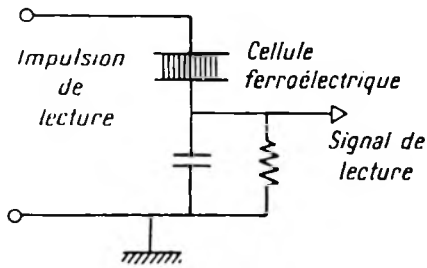


FIG. 121. — Lecture de l'information binaire contenue dans une mémoire ferro-électrique.

Si l'on applique à l'entrée de ce circuit une impulsion de même signe que la charge résiduelle de la cellule *A B A*, la résistance sera traversée par un courant très faible et on recueillera à la sortie un signal négligeable. Pour une polarité

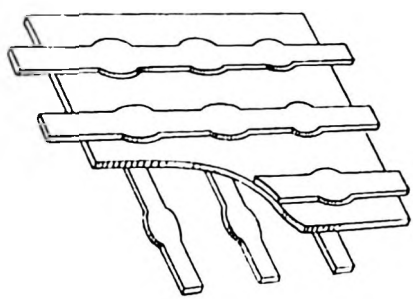


FIG. 122. — Mémoire matricielle ferro-électrique.

opposée (*A C D*), la résistance sera le siège d'un courant notablement plus élevé et on recueillera un signal de lecture.

Ce type de mémoire se prête bien à une disposition matricielle, comme le montre la *figure 122*. Une plaque de titanate de baryum est enserrée entre deux jeux de barres métalliques dont les points de croisée définissent les cellules élémentaires. La sélection d'une cellule particulière résulte de la mise sous tension des deux barres correspondantes.

Mémoires magnétiques dynamiques, tambours et rubans magnétiques.

Les tambours et rubans magnétiques permettent de constituer des mémoires de grande capacité — de l'ordre

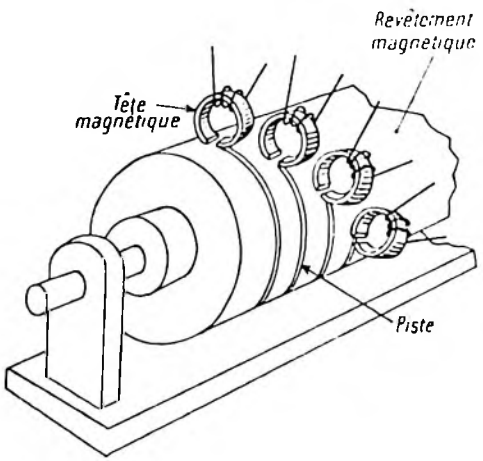


FIG. 123. — Tambour magnétique.

d'un million de chiffres binaires pour les tambours et d'une dizaine de millions de chiffres pour les rubans. Cependant, la réduction du prix des mémoires à tores de ferrite, qui tend

à rejoindre, pour une même capacité, celui des tambours magnétiques, entraînera à plus ou moins brève échéance la disparition de ces derniers, en raison de l'inconvénient constitué par leur temps d'accès relativement long. Les rubans magnétiques, ou des dispositifs spéciaux tenant à la fois du tambour et du ruban, continueront à être utilisés pour les très grandes capacités de mémoire.

Dans ce type de mémoire, l'information binaire est emmagasinée sous la forme du signe de l'aimantation résiduelle

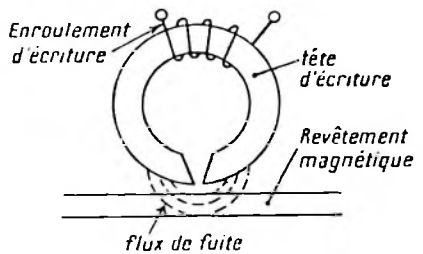


FIG. 124. — Disposition schématique de la tête d'écriture et du revêtement magnétique.

qui règne dans une couche magnétique déposée à la surface du tambour ou du ruban; cette couche est, généralement, constituée par des mélanges d'oxydes des métaux bivalents, de fer en particulier et, quelquefois, par du nickel dans le cas des tambours. Un même ruban peut porter plusieurs pistes parallèles associées chacune à une tête magnétique. Un même tambour peut porter quelques centaines de pistes (*fig. 123*).

Le signal d'écriture fondamental est constitué par l'inversion du flux magnétique dans une tête d'écriture présentant un entrefer d'une dizaine de microns avec le matériau magnétique (*fig. 124*). Il en résulte une inversion de l'induction rémanente subsistant dans le matériau après son passage sous la tête. Inversement, lorsque cette information passera ultérieurement sous une tête de lecture, qui peut d'ailleurs

être confondue avec la tête d'écriture, l'inversion de l'induction rémanente entraînera l'inversion du flux à travers l'enroulement de lecture, aux bornes duquel apparaîtra donc un signal dont la polarité dépendra du sens de l'inversion de l'induction (de 0 à 1 ou de 1 à 0). Le processus de lecture est schématisé sur la *figure 125*.

A partir de ces signaux élémentaires, l'information binaire à mettre en mémoire peut être codée de diverses manières. Dans le système de la *figure 126*, dit « *avec retour à zéro* »,

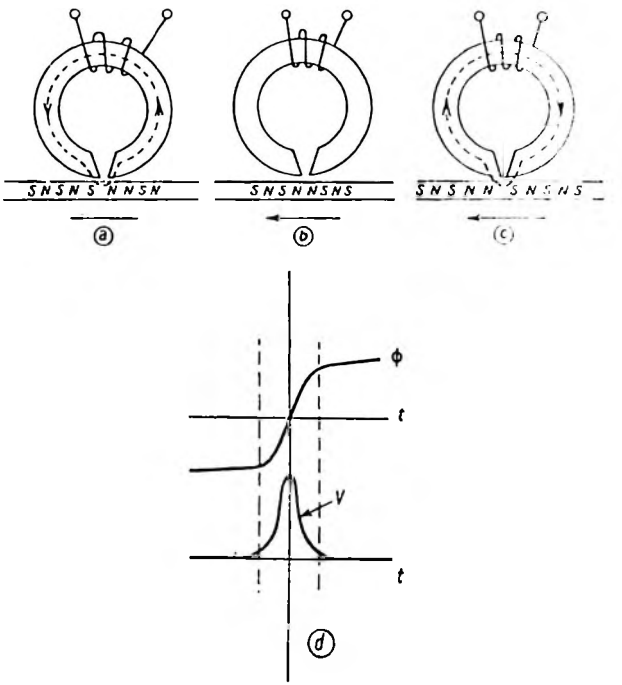


FIG. 125. — Processus de lecture : l'enroulement de lecture est le siège d'une force électromotrice d'induction lorsque change le signe de l'induction rémanente.

le courant d'écriture se présente sous la forme habituelle de la représentation série des nombres binaires, les niveaux 0 et 1 correspondant respectivement aux courants de saturation négatif et positif. Un « 1 » est donc enregistré sous la forme d'un *dipôle* positif dans un fond de magnétisation négative. A la lecture, un 1 est caractérisé par l'apparition

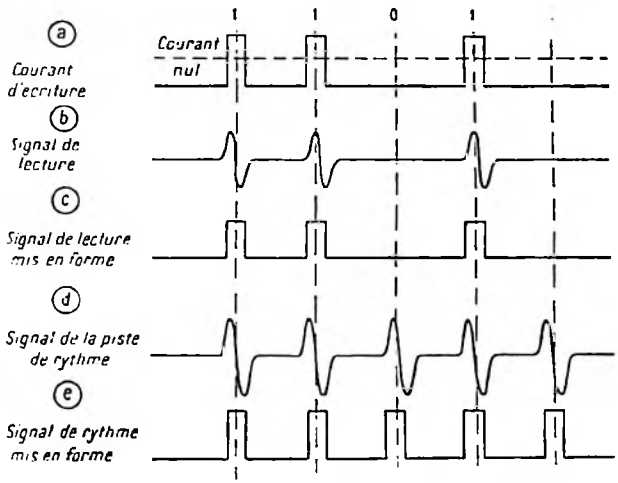


FIG. 126. — Processus d'écriture et de lecture dans le système d'inscription avec retour à zéro.

d'une double impulsion d'abord positive, puis négative, et un 0 par l'absence de signal. Après mise en forme, une impulsion positive représente un 1 et l'absence d'impulsion un 0. Afin de distinguer un 0 ou une suite de 0 de l'absence totale d'information, il est nécessaire de disposer d'un signal de référence repérant les endroits où se trouve une information. Ce signal de référence est lu sur une piste spéciale, dite « *piste de rythme* », sur laquelle a été enregistrée une suite de 1 dans les régions porteuses d'information. Cette

information est un 1 en cas de coïncidence d'un signal de lecture avec un signal de rythme. Nous retrouvons ici la notion de *test* dont nous avons signalé l'importance dès le début de cette partie technologique.

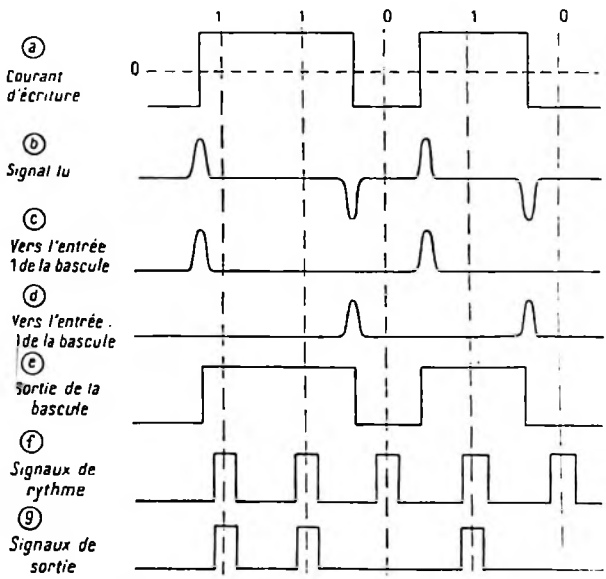


FIG. 127. — Processus d'écriture et de lecture dans le système d'inscription sans retour à zéro.

Dans le système de la *figure 127*, dit « *sans retour à zéro* », le courant d'écriture conserve sa valeur positive pour une suite de 1. Ce système présente, par rapport au précédent, l'avantage de permettre une densité d'information double. A la lecture, il est nécessaire de rétablir, à partir du signal lu, la forme du signal de lecture. A cet effet, les impulsions positives du signal de lecture sont appliquées à l'entrée « 1 »

d'une bascule, et les impulsions négatives à son entrée « 0 ». Il suffit de tester le signal de sortie de la bascule au moyen des signaux de rythme pour rétablir l'information binaire initiale.

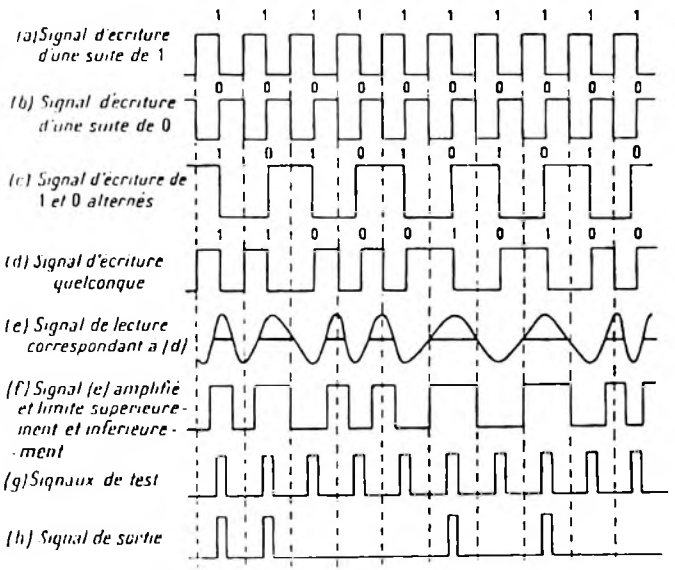


FIG. 128. — Processus d'écriture et de lecture dans le système d'inscription à modulation de phase.

Dans un troisième système, dû à WILLIAMS, KILBURN et THOMAS (Univ. de Manchester) et connu sous le nom de *système à modulation de phase*, un « 1 » est écrit par un courant positif pendant la première moitié de la période de rythme et négatif pendant la seconde moitié, et inversement pour un « 0 ». On peut voir, sur la *figure 128*, les signaux d'écritures correspondant à une suite de 1, une suite de 0, une suite alternée de 1 et de 0, et, enfin, une configuration quelconque.

On voit que la valeur moyenne du courant d'écriture est nulle quelle que soit la configuration écrite. C'est là le principal avantage de ce système, qui permet l'emploi de couplages à courant alternatif dans les circuits d'écriture.

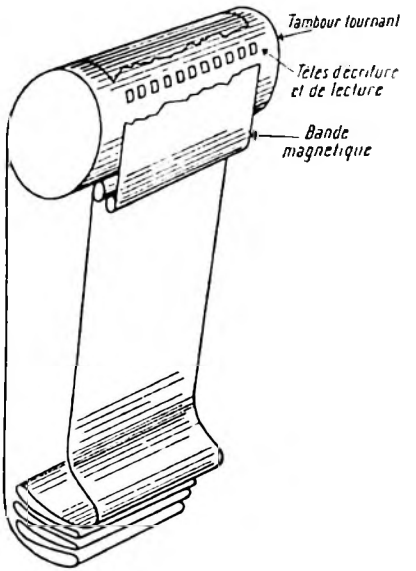


FIG. 129. — Représentation schématique du « *Tapedrum* ».

Les auteurs de ce système ont imaginé la méthode de lecture suivante. Le transformateur associé à la tête de lecture est accordé à la fréquence de rythme F, avec toutefois un amortissement assez important. Pour une suite de 1 ou de 0, le signal de lecture est approximativement une sinusoïde de fréquence F; pour une alternance de 0 et de 1, c'est encore un signal approximativement sinusoïdal, mais de fréquence $F/2$. Un réglage de l'amortissement permet de

donner à ces deux sinusoïdes des amplitudes approximativement égales. La *figure 128* montre le signal obtenu dans le cas d'une suite quelconque de 1 et de 0. Si ce signal est amplifié, puis limité positivement et négativement, on obtient une succession de créneaux que l'on peut tester par les signaux de test indiqués sur la figure pour restituer l'information binaire enregistrée. Afin d'éliminer le retard introduit par ce processus de lecture, les auteurs utilisent une tête de lecture placée un peu en avant de la tête d'écriture.

La densité des informations que l'on peut inscrire sur un ruban ou un tambour magnétique est couramment de l'ordre de 4 chiffres binaires par millimètre et atteint parfois 8 à 10 chiffres par millimètre.

Signalons enfin le *Tapedrum*, de *Brush Development Co.*, qui possède une capacité comparable à celle de plusieurs rubans, avec un temps d'accès du même ordre qu'un tambour (*fig. 129*). Le support magnétique est constitué ici par une large bande séparée par un mince entrefer d'un tambour tournant portant les têtes de lecture et d'écriture. La bande est divisée en « pages » de longueur égale à la demi-circonférence du tambour porte-têtes. Tant que l'on travaille sur une même page, le temps d'accès moyen est égal à un demi-tour du tambour. Un servo-mécanisme à commande numérique permet de choisir une page quelconque.

Circuits logiques à transistors.

Le fonctionnement des transistors repose sur les propriétés des semi-conducteurs tels que le germanium et le silicium. Ils présentent, sur les tubes électroniques, les avantages d'un encombrement moindre et d'une consommation beaucoup plus faible. On trouve actuellement sur le marché des transistors à jonction doués de caractéristiques suffisamment reproductibles pour leur emploi pratique.

Considérons le montage dit « émetteur à la masse » d'un transistor à jonction du type *p-n-p* (*fig. 130 a*). La base forme avec l'émetteur et le collecteur deux diodes dont la

première est polarisée dans le sens direct et la seconde dans le sens inverse. De plus, l'impédance inverse de la diode base-collecteur est sous la dépendance du courant direct à travers la diode base-émetteur : elle est d'autant plus faible que ce courant est plus élevé. Ce fonctionnement est donc analogue à celui d'une triode à vide, dans laquelle l'impédance entre la cathode et la plaque dépend de la tension appliquée

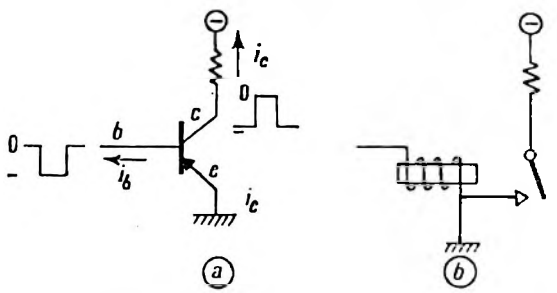


FIG. 130. — Analogie entre un relais et un transistor monté « émetteur à la masse »; circuit de négation.

à la grille de commande. Il s'accompagne d'une amplification de puissance, ce qui permet à un même transistor d'en alimenter plusieurs autres.

Examinons le fonctionnement par tout ou rien du montage de la *figure 130 a*. Si la base est à la masse, le courant de base i_b est nul et le courant de collecteur i_c très faible, de sorte que le collecteur est pratiquement à la tension d'alimentation « moins », d'une valeur de quelques volts. Si maintenant nous portons la base à une tension négative de quelques dizièmes de volts, et à *fortiori* au « moins », l'impédance entre la base et le collecteur devient très faible devant la résistance de charge, et le collecteur se trouve virtuellement mis à la masse. On voit que ce fonctionnement est tout à fait analogue à celui du relais de la *figure 130 b*, dont la paillette se trouve portée à la tension « moins » d'alimentation tant que le relais n'est pas excité, et à la masse quand

il est excité. Cette analogie permet de transposer aisément tout circuit à relais en un circuit à transistors (1).

On remarque que le fonctionnement du circuit de la *figure 130 b* s'accompagne d'une inversion de phase : c'est donc un circuit de négation ou complémentation; si le signal de base représente l'information binaire x, la tension de collecteur représente $\bar{x}$. On peut, par exemple, donner à la tension « moins » la valeur binaire 1 et à la tension de la masse la valeur 0. Un second transistor permet de reproduire l'information initiale x (*fig. 131*). On remarquera la simplicité de la liaison directe entre les deux étages, impossible avec des tubes à vide.

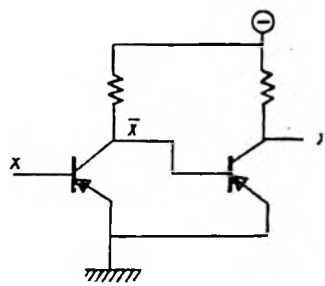


FIG. 131. — Couplage direct de deux transistors.

Dans le circuit de réunion à relais de la *figure 132 a*, la résistance de charge est parcourue par un courant si l'un au moins des relais est excité; de même, dans la version à transistors de la *figure 132 b*, il circule un courant de charge, et la borne de sortie se trouve virtuellement à la masse, si l'un au moins des transistors est rendu conducteur par l'application d'une tension négative à sa base. Si x, y et z désignent les signaux de base, le signal de sortie S s'écrit donc :

$$S = \overline{x \vee y \vee z} = \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

De même, la *figure 133* représente un circuit d'intersection à relais et sa version transistorisée. Le courant de

(1) R. B. BROWN et R. H. BETER : Transistors : a new class of relays, *Control Engineering*, December 1956.

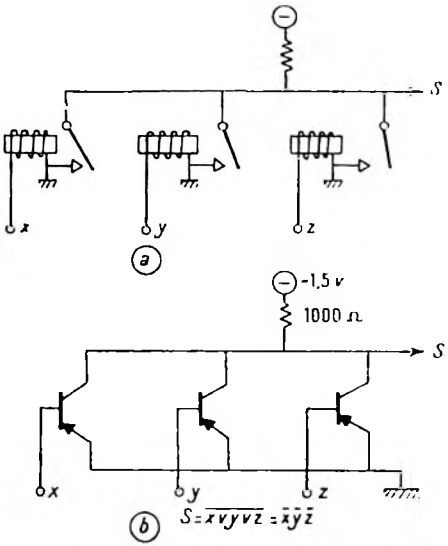


FIG. 132. — Circuits de réunion-négation.

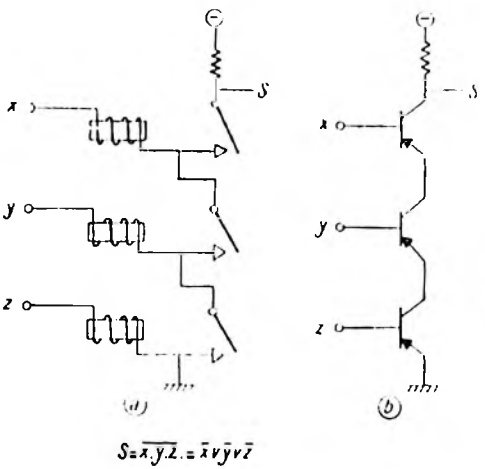


FIG. 133. — Circuits d'intersection-négation.

charge ne peut exister que si les trois transistors sont rendus simultanément conducteurs. Le courant d'émetteur de chaque transistor, égal à la somme de ses courants de base et de collecteur, est fourni par le collecteur du précédent. Le signal de sortie s'écrit :

$$S = x \cdot y \cdot z = \bar{x} \vee \bar{y} \vee \bar{z}.$$

Enfin, l'association de deux relais ou transistors s'alimentant l'un l'autre (*fig. 134*) permet de constituer un organe de mémoire binaire. Dans le montage à relais représenté en (*a*), le relais 2 est excité et son contact court-circuite l'enroulement d'excitation du relais 1. Pour faire passer le système dans l'état symétrique, il suffit de mettre un instant le point *B* à la masse : l'enroulement du relais 2 se trouve alors court-circuité, son contact s'ouvre, le relais 1 s'excite et son contact vient court-circuiter l'enroulement du relais 2; on peut alors séparer le point *B* de la masse.

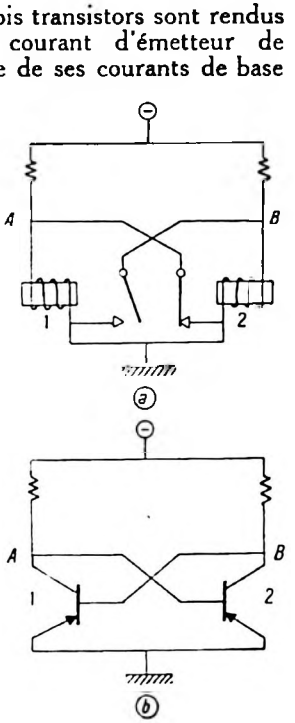


FIG. 134.
Montages en bascule.

Supposons de même que le transistor 2 du montage de la *figure 134 b* soit conducteur; son collecteur est à la masse, ainsi donc que la base du transistor 1, qui se trouve ainsi bloqué. Mettons un instant le point *A* à la masse : le transistor 2 se bloque, son collecteur prend la tension « moins », ainsi que la base de 1, qui devient ainsi conducteur, bloquant le transistor 2 par mise à la masse de sa base. On peut alors libérer le point *A*.

Mémoires électrostatiques.

Dans ce domaine, nous nous contenterons de décrire brièvement la mémoire à tube cathodique de WILLIAMS. Son fonctionnement repose sur les modifications de potentiel de l'écran d'un tube cathodique sous l'effet de son bombardement par le faisceau électronique. En appliquant des signaux convenables aux plaques déviatrices et à la grille

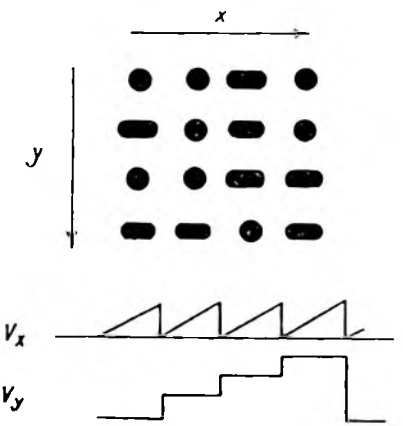


FIG. 135. — Configuration de points et de traits sur l'écran du tube cathodique d'une mémoire électrostatique de WILLIAMS, et signaux de balayage correspondants.

de commande, on peut inscrire sur l'écran du tube, ligne par ligne et en synchronisme avec une horloge, une configuration de points et de traits représentant respectivement des 0 et des 1 (*fig. 135*). Si, après avoir été inscrite, une telle ligne est relue par un nouveau passage du spot, on peut recueillir, par couplage capacitif sur une électrode disposée devant l'écran du tube (*fig. 136*), des signaux dont la polarité dépend initialement de la nature du chiffre lu (*fig. 137*). Si ce dernier était un 1, il convient de le reproduire en

allongeant le signal de lecture au moyen d'une porte commandée par l'amplificateur (*fig. 136*).

Une telle configuration de charges électriques, étant essentiellement évanescente, doit être régénérée fréquemment par lecture effectuée à intervalles réguliers. Afin de diminuer le temps d'accès, les lignes sont lues dans l'ordre 0, *n*, 1, *n*, 2, *n*, 3, *n*, . . . *p*, *n*, 0, *n*, 1, *n* . . ., la ligne *n* étant

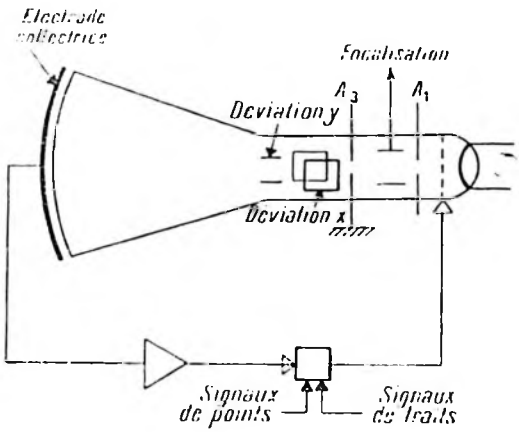


FIG. 136. — Schéma de principe d'une mémoire électrostatique de WILLIAMS.

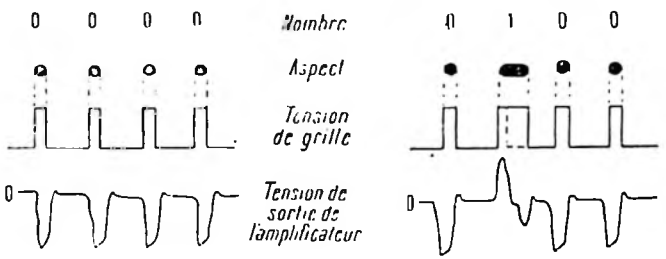


FIG. 137. — Signaux de lecture d'une mémoire de WILLIAMS.

celle que l'on désire inscrire ou lire et les autres étant simplement régénérées. Ainsi, pour 32 lignes de 32 chiffres binaires chacune et 8 μs par chiffre, le balayage d'une ligne dure environ 300 μs et chaque ligne est régénérée toutes les 20 μs. Le temps d'appel moyen est de 150 μs.

Quelques indications sur les machines décimales.

Pour représenter les 10 chiffres de la numération décimale, on doit disposer d'un organe susceptible de prendre dix états distincts. Un tel système peut être constitué de diverses manières.

CHIFFREUR DÉCIMAL PUR. — Un chiffreur décimal peut tout d'abord être constitué par dix éléments binaires. Chaque chiffre décimal est alors représenté par une configuration de ces dix organes dans laquelle l'un d'eux est dans l'état 1 et les neuf autres dans l'état 0. On obtient ainsi les dix configurations suivantes :

0 : 0000000001
1 : 0000000010
2 : 0000000100
3 : 0000001000
4 : 0000010000
5 : 0000100000
6 : 0001000000
7 : 0010000000
8 : 0100000000
9 : 1000000000

Ce mode de représentation brutale est rarement utilisé, en raison de son caractère onéreux en matériel.

CODE DÉCIMAL BIQUINAIRE. — Comme son nom l'indique, ce mode utilise à la fois la base 2 et la base 5. Le chiffreur décimal *biquinaire* est constitué par l'association d'un chiffreur *binaire* et d'un chiffreur *quinaire* formé par un ensemble

de cinq organes binaires; pour des raisons qui apparaîtront dans un instant, le chiffreur binaire est lui-même constitué par deux organes binaires. Le chiffreur biquinaire comprend donc en tout sept organes binaires et la suite des 10 chiffres décimaux est représentée par les configurations suivantes de ces organes :

0 = 0 + 0 : 01 00001
1 = 0 + 1 : 01 00010
2 = 0 + 2 : 01 00100
3 = 0 + 3 : 01 01000
4 = 0 + 4 : 01 10000

5 = 5 + 0 : 10 00001
6 = 5 + 1 : 10 00010
7 = 5 + 2 : 10 00100
8 = 5 + 3 : 10 01000
9 = 5 + 4 : 10 10000

On voit que chacune de ces configurations contient deux 1 et cinq 0. La vérification de cet état de choses après tout transfert ou toute opération constitue une méthode de contrôle très efficace. Cette méthode de contrôle, mise en œuvre avec succès dans diverses calculatrices électroniques modernes, a été utilisée pour la première fois dans la machine à relais des *Bell Telephone Laboratories* installée au terrain d'expériences d'Aberdeen (Maryland, U.S.A.). Au cours d'un service régulier, jour et nuit, d'une dizaine d'années, il n'est jamais arrivé que cette machine commette une erreur sans s'en apercevoir immédiatement.

La même méthode est mise en œuvre dans la machine électronique binaire du *Telecommunication Research Establishment* (Malvern, Royaume-Uni), dans laquelle chaque chiffreur binaire est constitué par deux bascules pouvant constituer les deux configurations 01 et 10; les configurations 00 et 11 indiquent que s'est produit un défaut de fonctionnement.

CODES DÉCIMAUX BINAIRES. — Il est évidemment possible de représenter chacun des 10 chiffres décimaux par son équivalent binaire, au moyen de quatre organes binaires. Mais ce mode de représentation présente l'inconvénient de ne pas permettre l'extraction facile du complément à 9, utilisé couramment pour la soustraction; le complément restreint des nombres binaires représentatifs des 10 chiffres décimaux est en effet différent du complément à 9 de leur équivalent décimal.

Parmi les codes satisfaisants à la condition ci-dessus, le code d'AIKEN (*Harvard University*, U.S.A.) est remarquable par sa simplicité : les chiffres inférieurs à 5 y sont représentés par leurs équivalents binaires, tandis que les chiffres égaux ou supérieurs à 5 sont d'abord majorés de 6 avant leur conversion en numération binaire :

0 : 0000	5 = 11 — 6 : 1011	
1 : 0001	6 = 12 — 6 : 1100	
2 : 0010	7 = 13 — 6 : 1101	
3 : 0011	8 = 14 — 6 : 1110	
4 : 0100	9 = 15 — 6 : 1111	

On remarque qu'on peut affecter aux 4 chiffres binaires les poids 2, 4, 2, 1, dans l'ordre des poids décroissants; ainsi : 7 (1101) = 2 + 4 + 0 + 1. Enfin, la parité est respectée, puisque 6 est pair.

On se souvient également du code spécial à cinq moments, que nous avons étudié au cours de notre exposé sur les circuits de codage (page 136). Ce code offre les mêmes possibilités de contrôle que le code biquinaire, puisque chaque chiffre décimal y est représenté par une configuration de deux 1 et trois 0. Ce code est également utilisé dans plusieurs machines modernes.

SOUSTRACTION DÉCIMALE, COMPLÉMENTS. — On se souvient que, dans le cas du système binaire, nous avions affecté la gamme de tous les nombres représentables dans une machine donnée pour moitié aux nombres positifs et pour

moitié aux nombres négatifs. On peut faire de même pour une machine décimale. Ainsi, pour une machine à 3 chiffres décimaux, capables de traiter tous les nombres compris entre 0 et 999, nous laisserons inchangés les nombres compris entre 0 et 499, et nous utiliserons les nombres compris entre 500 et 999 pour représenter les nombres négatifs de valeurs absolues égales à leurs compléments à 1000 (*fig. 138*). Ainsi, les nombres + 374 et — 374 sont représentés respectivement par les nombres positifs 374 et 626. Le complément vrai s'obtient en ajoutant une unité au complément restreint constitué par les compléments à 9 de chacun des chiffres.

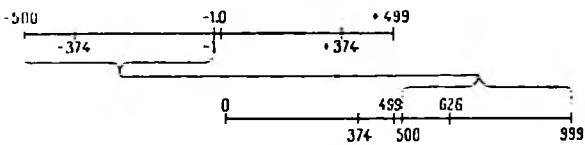


FIG. 138. — Correspondance entre les nombres négatifs et leurs compléments dans le système décimal.

La somme des nombres ainsi codés fournit la somme algébrique des nombres qu'ils représentent, à condition de ne pas tenir compte de la retenue qui peut éventuellement se produire du côté des forts poids :

$$374 - 253 : 374 + 747 = (1)121, \text{ soit } 121;$$
$$253 - 374 : 253 + 626 = 879, \text{ soit } -121.$$

Ici encore, on peut faire usage des compléments restreints, à condition de procéder à la retenue circulaire :

$$374 - 253 : 374 + 746 = (1)120, \text{ soit } 120 + 1 = 121;$$
$$253 - 374 : 253 + 745 = 878, \text{ soit } -121.$$

EXEMPLE D'ADDITIONNEUR DÉCIMAL (*fig. 139*). — Dans un but de simplification, adoptons le simple codage binaire pur. Soit à effectuer la somme de deux chiffres décimaux d'équivalents binaires (forts poids à gauche) $a_1\ a_2\ a_3\ a_4$

et $b_1\ b_2\ b_3\ b_4$, dont l'addition donne la somme $s_1\ s_2\ s_3\ s_4$ et la retenue R (0 ou 1). Nous procéderons en deux temps, séparés par une discrimination (*fig. 139*) :

Premier temps. — On commence par effectuer, au moyen de quatre additionneurs binaires à trois entrées, la somme brute $c_1\ c_2\ c_3\ c_4$.

Discrimination. — Il y aura retenue ou non selon que la somme brute est supérieure ou égale à 10, ou bien inférieure à 10; examinons le tableau des nombres binaires de 4 chiffres égaux ou supérieurs à 10 (10 à 15 inclus) :

10 : 1 0 1 0
11 : 1 0 1 1
12 : 1 1 0 0
13 : 1 1 0 1
14 : 1 1 1 0
15 : 1 1 1 1

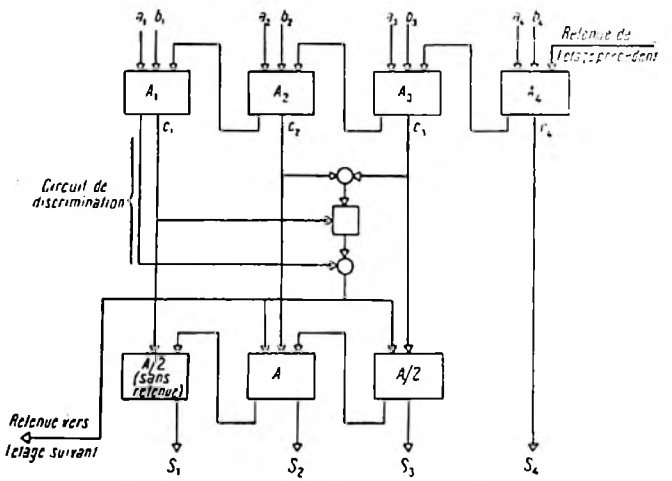


FIG. 139. — Étage d'addition de deux chiffres décimaux exprimés en numération binaire pure.

Les deux colonnes du milieu constituent le tableau de la réunion, répété deux fois; chacune des trois combinaisons se trouve associée, dans la colonne de droite, d'abord avec 0, puis avec 1; la colonne de droite est donc sans importance; enfin, le chiffre de gauche est toujours 1. Par conséquent, la fonction logique :

$$s_1 \cdot (s_2 \vee s_3)$$

est égale à 1 pour les six combinaisons ci-dessus. Elle est réalisée au moyen d'un mélangeur et d'un intersecteur; de plus, le résultat de cette discrimination est mélangé avec la retenue du premier additionneur, qui doit bien entendu engendrer aussi une retenue décimale.

Troisième temps. — Si le signal de sortie du circuit ci-dessus est 1, c'est-à-dire s'il y a une retenue décimale, il faut soustraire 10 de la somme brute ou, ce qui revient au même, lui ajouter 6 (modulo 16); l'équivalent binaire de 6 étant 110, ce résultat est obtenu au moyen de deux demi-additionneurs et d'un additionneur à trois entrées; de plus, le demi-additionneur de gauche ne comporte pas de circuit de retenue.

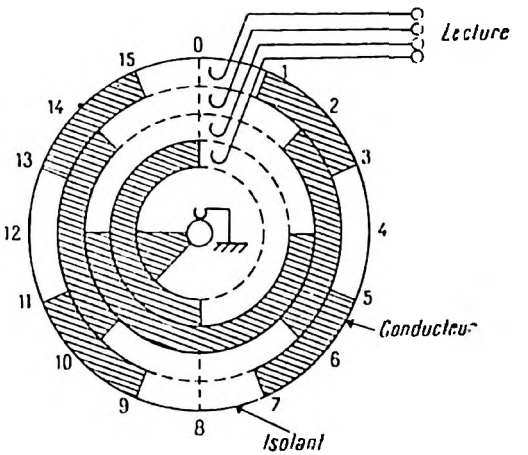


FIG. 140. — Disque codé en numération binaire réfléchie.

Codes cycliques ou réfléchis.

Considérons le code binaire suivant :

Décimal	Binaire réfléchi	Schéma
0	0 0 0 0 0	
1	0 0 0 0 1	
2	0 0 0 1 1	
3	0 0 0 1 0	
4	0 0 1 1 0	
5	0 0 1 1 1	
6	0 0 1 0 1	
7	0 0 1 0 0	
8	0 1 1 0 0	
9	0 1 1 0 1	
10	0 1 1 1 1	
11	0 1 1 1 0	
12	0 1 0 1 0	
13	0 1 0 1 1	
14	0 1 0 0 1	
15	0 1 0 0 0	
16	1 1 0 0 0	
17	1 1 0 0 1	
18	1 1 0 1 1	
19	1 1 0 1 0	
20	1 1 1 1 0	
21	1 1 1 1 1	
22	1 1 1 0 1	

Ce code présente la propriété intéressante suivante : un chiffre et un seul se trouve modifié lorsque l'on passe d'un chiffre au suivant. Ce code trouve son emploi dans les commutateurs destinés à fournir une représentation binaire de la position angulaire d'un arbre (*fig. 140*). Une petite rotation de l'arbre modifiera au plus l'un des chiffres du nombre représentatif, alors que, si l'on utilisait le code binaire normal, plusieurs chiffres peuvent être modifiés

par une petite rotation de l'arbre (de 0111 à 1000, par exemple); pour qu'il n'y ait pas d'ambiguïté, il faudrait alors que tous les chiffres modifiés le soient simultanément, résultat qu'il est difficile d'atteindre en pratique, de sorte que le nombre lu pourra présenter, au cours du processus de commutation, des erreurs transitoires importantes par rapport à sa vraie valeur (1111 par exemple dans le cas considéré ci-dessus). On utilise également des disques codés dont les plages blanches et noires sont observées par des cellules photoélectriques.

Pour convertir un nombre binaire x dans le code réfléchi, on peut faire la somme sans retenue de x et $2x$, puis abandonner le chiffre de plus petit poids; ainsi, si $x = 110$:

$$\begin{array}{r} 1\,1\,0 \\ +\,1\,1\,1\,0 \\ \hline 1\,0\,1\,(0) \end{array}$$

Plus généralement, on vérifiera par l'examen du tableau ci-dessus que la correspondance entre les chiffres a, b, c, d, e, d'un nombre binaire réfléchi et les chiffres x, y, z, t, u, de son équivalent binaire normal, est donnée par les relations :

$$\begin{array}{llll} a = x \oplus y & & x = a \oplus b \oplus c \oplus d \oplus e \\ b = y \oplus z & & y = b \oplus c \oplus d \oplus e \\ c = z \oplus t & & z = c \oplus d \oplus e \\ d = t \oplus u & & t = d \oplus e \\ e = u & & u = e \end{array}$$

On notera qu'une somme modulo 2 de plusieurs variables binaires est égale à 1 si un nombre impair de ces variables sont égales à 1.

Enfin, on vérifiera que l'équivalent décimal d'un nombre binaire réfléchi s'obtient en donnant aux chiffres successifs, en partant des faibles poids, les poids 1, 3, 7, 15, 31, etc., obtenus en retranchant 1 aux puissances successives de 2 et en affectant ces nombres de signes alternés.

Le principe des codes cycliques n'est pas limité au système binaire. Voici, par exemple, un code décimal réfléchi :

0	0	10	19	20	20	30	39
1	1	11	18	21	21	31	38
2	2	12	17	22	22	32	37
3	3	13	16	23	23	33	36
4	4	14	15	24	24	34	35
5	5	15	14	25	25	35	34
6	6	16	13	26	26	36	33
7	7	17	12	27	27	37	32
8	8	18	11	28	28	38	31
9	9	19	10	29	29	39	30

On voit peut-être mieux, dans ce système, les raisons qui motivent le nom de « codes réfléchis » donné à ces représentations.

Maintenance et contrôles.

Il y a quelques années, l'opportunité de l'introduction d'*organes de contrôle interne* dans les calculatrices numériques électroniques était très controversée. L'insécurité des organes était alors telle qu'on pouvait craindre, à juste titre, l'accroissement du nombre des pannes par suite de la présence d'organes supplémentaires de contrôle. Aujourd'hui, il n'en va plus de même, et les utilisateurs, tant comptables que mathématiciens, insistent pour que le nombre des erreurs non décelées soit réduit à une valeur incompatible avec la seule sécurité de fonctionnement des organes de mémoire, de calcul et de programme. C'est pourquoi les machines les plus récentes sont toutes munies d'organes de contrôle plus ou moins élaborés.

Bien entendu, l'existence d'organes de contrôle ne dispense pas d'introduire dans les programmes, toutes les fois que cela est possible, des contrôles d'ordre mathématique : vérification d'une identité existant entre diverses variables,

contrôle de la continuité d'une fonction par le calcul de ses différences successives, vérification par substitution des racines d'une équation, etc. Dans d'autres cas, il sera avantageux d'effectuer le même calcul deux fois de suite, de préférence au moyen de deux programmes différents.

MAINTENANCE PRÉVENTIVE, ESSAIS MARGINAUX. — L'exécution à intervalles réguliers (toutes les semaines par exemple) de programmes d'essais dans des conditions de fonctionnement marginales constitue un excellent moyen pour déceler les organes présentant une certaine probabilité d'entraîner des défauts de fonctionnement au cours de la période d'exploitation suivante. Le principe des *programmes de test* consiste à vérifier le fonctionnement d'un organe, d'un groupe d'organes ou de l'ensemble de la machine en lui faisant exécuter des programmes spécialement conçus pour permettre la localisation et le remplacement rapides des organes défectueux.

De plus, ces essais sont effectués dans des *conditions marginales*, c'est-à-dire dans des conditions telles que se trouvent réduites les marges de sécurité que l'on s'accorde toujours lors de la conception d'un organe électronique, compte tenu des tolérances admises pour les éléments du montage. Selon la nature du montage, on pourra, par exemple, réduire l'amplitude des impulsions de commande, diminuer les tensions de plaque, les tensions de polarisation de grille, les courants de chauffage, etc... On mettra ainsi en évidence les marges qui sont devenues insuffisantes, et on les rétablira à leurs valeurs normales par un réglage ou par le remplacement d'un organe reconnu défectueux. Ce mode de *maintenance préventive* permet en particulier d'éliminer presque totalement les pannes dues au vieillissement des tubes électroniques.

CONTROLE AU MOYEN DE NOMBRES PONDÉRÉS. — Dans la calculatrice électronique binaire *Raytheon* (U.S.A.), chaque nombre binaire est accompagné d'un nombre pondéré, obtenu en calculant modulo 16 le nombre obtenu en donnant aux rangs successifs les poids 1, 2, 4, 1, 2, 4, etc...; ce nombre

est constitué par les 4 chiffres de plus faible poids de la somme des poids des rangs où figure le chiffre 1. Tout transfert est contrôlé en formant le nombre pondéré du nombre arrivé à destination et en le comparant au nombre pondéré transmis avec le nombre transféré.

Le contrôle des opérations arithmétiques repose sur le fait qu'il existe une relation mathématique entre la somme, différence, produit ou quotient des nombres pondérés associés à deux nombres et le nombre pondéré associé à la somme, différence, produit ou quotient de ces deux nombres. Toutefois, dans l'opérateur arithmétique, les poids successifs sont 1, 2, 4, 8, 16, 1, 2, 4, 8, 16, etc..., et la somme n'est pas réduite modulo 16, c'est-à-dire que tous ses chiffres sont conservés. Ainsi, dans le cas du produit $x\ y = z$, la relation suivante existe entre les nombres pondérés x_p, y_p et z_p associés aux trois nombres x, y et z :

$$[|(x_p \cdot y_p)_p - z_p| + 31]_p = 31.$$

Les mêmes méthodes sont utilisées pour contrôler les organes d'entrée et de sortie, la sélection spatiale et temporelle des mémoires et le fonctionnement des codeurs d'opérations.

Codes détecteurs et correcteurs d'erreurs. — Le plus simple des codes détecteurs d'erreurs est le contrôle de parité utilisé dans de nombreuses machines modernes. Le principe de cette méthode consiste à associer à un nombre binaire de n chiffres un $(n + 1)^e$ chiffre auquel on donne la valeur 0 ou 1, de façon que le nouveau nombre de $(n + 1)$ chiffres qui en résulte soit, par exemple, impair. Si l'un des chiffres de ce nouveau nombre est faux, l'erreur se manifestera immédiatement par un changement de parité. Des codes plus élaborés permettent de mettre en évidence des erreurs portant sur plus d'un chiffre.

D'autre part, nous avons déjà signalé les possibilités de contrôle offertes par certains codes spéciaux, tels que le code biquinaire ou le code binaire à deux 1 et trois 0.

Mais on peut aller plus loin encore et imaginer des codes permettant de localiser les chiffres faux, et, par conséquent, de les corriger automatiquement. Donnons un exemple

d'un code permettant la détection et la correction d'une erreur portant sur un seul des 4 chiffres du nombre binaire $n = a\,b\,c\,d$ (1). On forme, à partir du nombre n, le nombre binaire de 7 chiffres $N = A\,B\,C\,D\,E\,F\,G$

$$A = a + b + d \text{ (modulo 2).}$$
$$B = a + c + d \text{ (modulo 2).}$$
$$C = a.$$
$$D = b + c + d \text{ (modulo 2).}$$
$$E = b.$$
$$F = c.$$
$$G = d.$$

Il est donc toujours possible de reformer le nombre initial n à partir du nouveau nombre N. Supposons maintenant que l'un des 7 chiffres de ce dernier soit faux; pour le localiser, on forme le nombre binaire à 3 chiffres $w = x\,y\,z$:

$$x = D + E + F + G$$
$$y = B + C + F + G$$
$$z = A + C + E + G$$

Si $w = 0$, le nombre N ne comporte pas d'erreur; si $w \neq 0$, le chiffre de rang w de N est faux et il convient donc de le corriger en le remplaçant par son complément restreint.

Ainsi, si $n = 1101$: $N = N' = 1010101$ et $w = 000$. Changeons maintenant le cinquième chiffre de N : $N' = 1011000\,1$; nous obtenons alors : $w' = 101$, équivalent binaire de 5, rang du chiffre faux, dont il suffit de prendre le complément pour rétablir la vérité.

Dans le cas général, n chiffres de contrôle permettent de localiser une erreur unique dans un nombre binaire de $(2^n - n + 1)$ chiffres.

---

(1) RUTISHAUSER, SPEISER et STIEFEL : *Programmgesteuerte digitale Rechenmaschine*, Separatdruck aus der Z. f. angew. Mathematik, Verlag Birkhäuser, Bâle, 1951.

Application de l'algèbre binaire au calcul des propositions.

Les valeurs 1 et 0 du calcul binaire peuvent être attribuées à la véracité et à la fausseté d'une proposition. Ainsi, la proposition résultant de l'intersection de deux propositions élémentaires n'est vraie que si ces dernières sont simultanément vraies, tandis que la proportion résultant de leur réunion n'est fausse que si elles sont simultanément fausses. Ainsi, l'algèbre binaire permet de juger de la véracité ou de la fausseté d'une proposition résultant de certaines hypothèses et de l'application de certaines règles constituant en quelque sorte la règle du jeu.

Donnons-en un exemple amusant très simple. Il est bien connu que, dans une société, les membres du service commercial disent toujours la vérité, alors que les ingénieurs mentent toujours. Soient sept personnes de cette société, que nous appellerons par discrétion *A*, *B*, *C*, *D*, *E*, *F* et *G*. On a recueilli les informations suivantes, au cours de leurs nombreuses périgrinations dans les couloirs pendant les heures de service :

C dit que *D* est ingénieur;

A dit que *B* déclare que *C* prétend que *D* lui a dit que *E* affirme que *F* répand le bruit que *G* n'est pas au service commercial.

De plus, on a réussi à savoir, par l'observation de leurs trains de vie respectifs, que *A* est ingénieur et que *B* et *E* sont des commerciaux. Peut-on tirer de ces diverses informations quelque conclusion sur le nombre d'ingénieurs de la société?

Tout d'abord, le fait d'être ou de ne pas être ingénieur étant une variable binaire, nous affecterons (tout à fait arbitrairement, cela va sans dire) la valeur 1 à l'ingénieur et la valeur 0 au commercial. L'énoncé nous fournit donc déjà les trois égalités :

$$A = 1 \qquad B = 0 \qquad E = 0$$

Pour mettre notre problème en équation, nous devons traduire logiquement des propositions telles que : « A dit X ». Il n'y a que deux cas possibles : ou bien A est ingénieur, et il ment, donc X est faux; ou bien A est commercial et X est vrai. Il existe donc, entre les variables binaires A et X, la relation du dilemme :

$$a \oplus X = A \cdot \overline{X} \vee \overline{A} \cdot X = 1.$$

Les deux propositions de l'énoncé nous fournissent donc les deux relations logiques :

$$C \oplus D = 1$$
$$A \oplus (B \oplus (C \oplus (D \oplus (E \oplus (F \oplus G) = 1.$$

La première équation nous apprend que, de C et de D, l'un d'eux seulement est ingénieur. Essayons maintenant de comprendre ce que signifie la seconde équation. Or, l'opération de dilemme, qui s'identifie avec l'addition modulo 2, est commutative et associative, ce qui permet de supprimer les parenthèses; de plus, son résultat est égal à 1 si un nombre impair de termes de la somme sont égaux à 1. Or, nous savons déjà que A est égal à 1, que B et E sont nuls et que, de C et de D, l'un des deux est égal à 1; il en résulte immédiatement que, de F et de G, l'un des deux est égal à 1.

En résumé, il y a quatre combinaisons possibles, et chacune d'elle comprend trois ingénieurs :

$$A\,C\,F \qquad A\,C\,G \qquad A\,D\,F \qquad A\,D\,G$$

Les combinaisons possibles des valeurs binaires possibles des sept variables sont donc conformes au tableau ci-dessous :

A	B	E	C	D	F	G
1	0	0	0	1	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	0	1	0

Si l'on veut bien reconnaître la grande difficulté de la résolution directe du problème posé, on reconnaîtra du même coup toute la puissance de l'algèbre de BOOLE pour la résolution des problèmes de *logique* et de *décision*. L'une de ses principales applications pratiques en ce domaine réside dans la détermination des conditions susceptibles de satisfaire simultanément un ensemble de règles complexes et dans la recherche des contradictions éventuelles entre ces

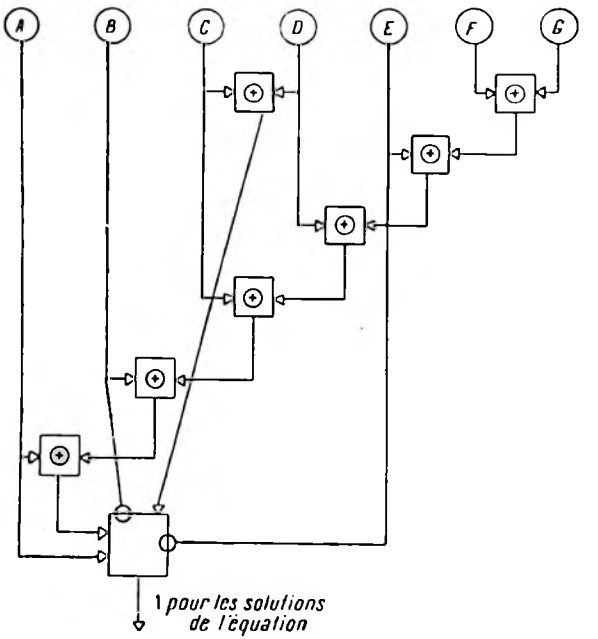


FIG. 141. — Connexions d'éléments logiques pour la résolution du système d'équations :

$$C \oplus D = 1,$$
$$A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G = 1,$$

avec : $A = 1 \quad B = 0 \quad E = 0.$

règles; des problèmes de cet ordre se posent, par exemple, dans le domaine des assurances, ainsi que dans les chemins de fer (enclenchements, sécurités, signaux, etc...).

D'autre part, les opérations logiques étant représentables par des circuits électriques ou électroniques, on peut construire une machine dont les divers organes logiques peuvent être interconnectés entre eux et avec les variables de manière à représenter les relations logiques contenues dans l'énoncé du problème. On passe ensuite en revue, au moyen d'un dispositif analogue à un compteur, toutes les combinaisons possibles des variables libres, et on repère, au moyen d'un indicateur convenable (voyant, enregistreur, etc...), celles d'entre elles qui satisfont aux conditions imposées. Des machines logiques de ce type ont été construites, mais il ne semble pas qu'elles aient rencontré beaucoup de succès jusqu'à présent; cela tient peut-être à ce que les problèmes de logique peuvent également être traités par un codage convenable des calculatrices numériques.

A titre d'exemple, la *figure 141* représente un montage correspondant à la résolution du problème traité ci-dessus et comprenant un circuit d'intersection, deux circuits de compléments et sept circuits de dilemme.

Conclusion.

Bien que les principes qui gouvernent leur fonctionnement soient connus depuis fort longtemps, les grandes machines mathématiques numériques n'ont vu réellement le jour que depuis une dizaine d'années. C'est dire, d'une part, que leur technologie, et même leur logique, sont loin d'être stabilisées, d'autre part, que leurs applications dans les divers domaines scientifiques, techniques, économiques et commerciaux n'ont été qu'effleurées. Un énorme travail reste à accomplir tant de la part des constructeurs de machines que de leurs utilisateurs. Mais on peut d'ores et déjà prévoir que leur emploi généralisé modifiera profondément les méthodes de travail du laboratoire, de l'usine et du bureau.

Cependant, les principes logiques et les moyens techniques que nous avons étudiés ne se limitent pas au domaine du

calcul automatique proprement dit. L'organisation logique des calculatrices numériques universelles en fait le modèle général des automatismes à séquences à fonctionnement conditionnel selon des règles bien déterminées. Les notions de code, de programme et de rupture de séquence conditionnelle permettent de transcrire tout processus se déroulant conformément à des règles établies dans le langage de la logique et de l'exprimer au moyen des opérations logiques fondamentales. Ces dernières trouvent à leur tour leur expression matérielle dans les diverses technologies électriques, électroniques et magnétiques que nous avons examinées. Ainsi l'algèbre de BOOLE, langage de la logique par excellence, est-il destiné à devenir le mode d'expression commun et le trait d'union de tous ceux, savants, ingénieurs et techniciens, qui sont aux prises avec des processus discontinus dont l'enchaînement des phases est régi par des consignes et des conditions logiques.

BIBLIOGRAPHIE SOMMAIRE

BRILLOIN, COUFFIGNAL et PÉRES : *Les grandes machines à calculer*, C.N.R.S., 1949.

HARTREE : *Calculating instruments and machines*, Cambridge University Press, 1950.

Staff of Engineering Research Associates : *High-speed computing devices*, McGraw Hill, 1950.

RUTISHAUSER, SPEISER et STIEFEL : *Programmgesteuerte digitale Rechengeräte*, Birkhäuser, 1951.

FAVIER et THOMELIN : *La mécanographie*, Éditions de Montligeon, 1952.

Les machines à calculer et la pensée humaine (Colloque international du C.N.R.S.), Éditions du C.N.R.S., 1953.

BOOTH et BOOTH : *Automatic digital calculators*, Butterworth, 1953.

BOOTH : *Numerical methods*, Butterworth, 1955.

RICHARDS : *Arithmetic operations in digital computers*, Van Nostrand, 1955.

RAYMOND : *Les calculatrices numériques universelles*, Mémorial de l'Artillerie française, 1955, 3e fasc. et 1956, 4e fasc.

WILKES : *Automatic digital computers*, Methuen & Co., 1956 (traduction française en préparation chez DUNOD).

MILLMAN & TAUB : *Pulse and digital circuits*, McGraw Hill, 1956.

STIBITZ & LARRIVÉE : *Mathematics and computers*, McGraw Hill, 1957.

RAYMOND : *L'automatique des informations*, Masson et Cie, 1957.

INDEX DES MATIÈRES

INDEX DES MATIÈRES

IMPRIMERIE DE MONTLIGEON,
LA CHAPELLE-MONTLIGEON
(ORNE). — 52405-11-57.
DÉPOT LÉGAL 4e TRIMESTRE.
— ÉDITEUR N° 3130. —
IMPRIMEUR N° 3126.