UNIVERSITÉ
DE PARIS

SOCIÉTÉ FRANÇAISE
DES ÉLECTRICIENS

ÉCOLE SUPÉRIEURE D'ÉLECTRICITÉ

PROGRAMMATION

LANGAGE DES MACHINES

CODIFICATION SYMBOLIQUE

par

P. HENRY

N° 2127

1968

SOMMAIRE

I - GENERALITES ET DEFINITIONS

I - 1. DEFINITIONS

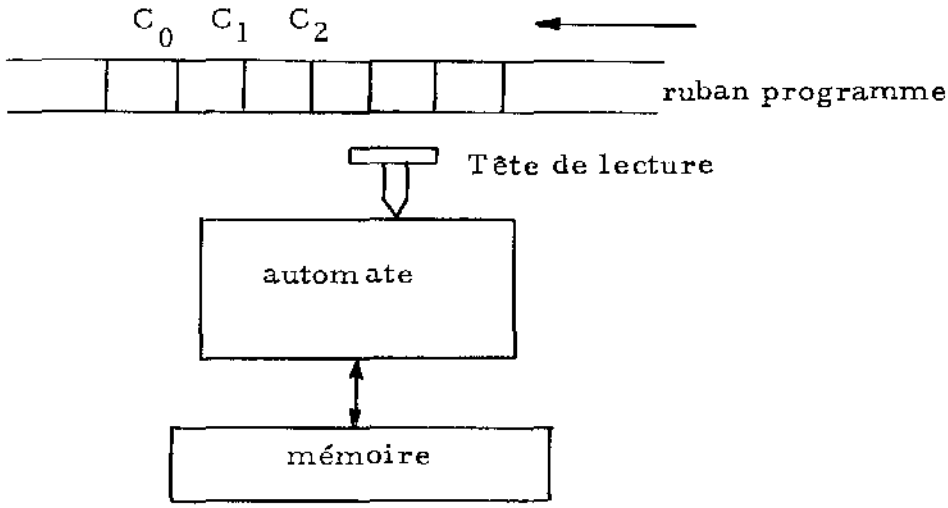
Pour introduire les notions de base utilisées en programmation, il nous parait utile de commencer par définir une machine simplifiée A dont les composants mettent en évidence les fonctions logiques de base que l'on retrouve dans toutes les machines, sans nous soucier de savoir d'ailleurs si une réalité technologique correspond à ces composants. A est une machine constituée d'un mécanisme central que nous appellerons automate, d'une tête de lecture pouvant lire l'information enregistrée sur un ruban dit ruban programme, et d'une mémoire.

L'automate est susceptible de prendre un nombre fini d'états. Nous appellerons E_A l'ensemble (R, S_0 , S_1, ... S_N) de ces états; parmi ceux-ci R et S_0 jouent un rôle particulier.

La mémoire contient une certaine quantité d'information, et sous l'action de l'automate cette information peut être modifiée: nous dirons que la mémoire change d'état et nous appellerons E_M l'ensemble (I_0, I_1, I_2,.... I_N) fini de ces états (leur nombre peut toutefois être très grand.)

Le ruban programme est constitué par des cellules C_0, C_1, C_2, ... C_N, en nombre fini pouvant contenir chacune une petite quantité d'information: nous appellerons $f(C_i)$ l'information se trouvant dans la cellule C_i .

Il peut se déplacer devant la tête de lecture de l'automate de la droite vers la gauche d'une cellule à la fois.



Le fonctionnement de la machine est le suivant:

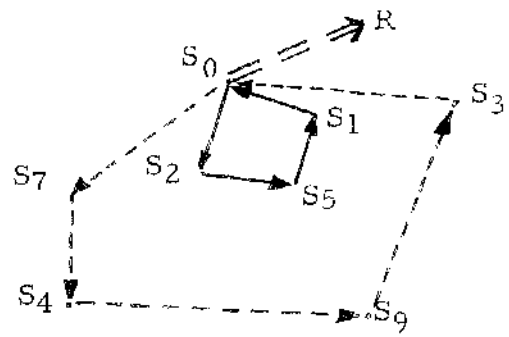
A l'état R l'automate est au repos, c'est-à-dire que la machine est à l'arrêt. Pour la mettre en marche, on positionne la cellule C_0 devant la tête de lecture, on met la mémoire en un état $I\alpha$ et on met l'automate en état S_0 (appui sur le bouton marche par exemple).

En état de marche, l'automate part toujours d'un état S_0, lit $f(C_i)$ et avance le ruban d'une cellule. En fonction de $f(C_i)$ il passe par un certain nombre d'états S_a, S_b, S_c, etc... pour revenir à l'état S_0; au cours de ces opérations, la mémoire qui était à l'état $I\alpha$ peut passer à un autre état $I\beta$ sous l'action de l'automate.

Puis le cycle recommence jusqu'à ce qu'une cellule C_i contienne une information $f(C_i)$ qui mette l'automate en état R.

On peut expliciter la marche de la machine A, en d'autres termes plus précis:

a) Le fonctionnement de l'automate est symbolisé par un graphe $G(E_A, \Gamma)$: à chaque cycle correspond un circuit ayant S_0 pour sommet et qui est déterminé par $f(C_i)$. Il y a une exception: il existe un $f(C_i)$ qui fait correspondre le sommet R au sommet S_0.



b) L'évolution du contenu de la mémoire est symbolisé par un graphe univoque $H(E_M, \Gamma')$. L'application Γ' sur l'ensemble E_M est fonction du circuit de l'automate, c'est-à-dire déterminée par $f(C_i)$.

A l'aide de cette représentation, il nous est maintenant possible de définir certaines notions:

- $f(C_i)$ s'appelle une instruction.

- la séquence ordonnée $f(C_0)$, $f(C_1)$ ... $f(C_i)$ ... $f(C_n)$ qui figure sur le ruban programme s'appelle le programme.

- l'état de l'information $I\alpha$ dans la mémoire M au moment de l'appui sur le bouton marche constitue les données du problème à résoudre, et l'état final $I\beta$, lorsque l'automate est revenu à l'état R, constitue les résultats.

Le programmeur dont le rôle est d'écrire le programme permettant de résoudre un problème doit connaitre les différentes valeurs que peut prendre $f(C_i)$, c'est-à-dire l'ensemble des instructions $(e_0, e_1, \ldots\ldots e_N)$ que la machine peut interpréter et l'application Γ'_i qui correspond à chaque e_i. C'est le manuel de programmation.

Les applications Γ'_i sont les opérations élémentaires que la machine peut effectuer automatiquement; on dit opérations câblées pour les calculateurs électroniques. La connaissance des circuits S_a, S_b, S_c .... S_0 correspondant à chaque e_i n'est pas indispensable; elle peut toutefois être utile pour certaines opérations complexes.

Exemple:

La plupart des machines électroniques utilisant la codification binaire, on represente l'information par des nombres binaires. Supposons que la machine A ait les caractéristiques suivantes:

Capacité d'une cellule C_i : 4 positions binaires

Capacité de la mémoire : 16 positions binaires.

Dans ces conditions, l'ensemble des instructions $(e_0 \ldots\ldots e_N)$ est l'ensemble des nombres entiers de 0 à 15 codifiés en binaire et l'ensemble E_M des états de la mémoire est l'ensemble des nombres entiers de 0 à 32.767 codifiés en binaire. Une règle du manuel de programmation sera par exemple:

Instruction $e_{15} = 15$

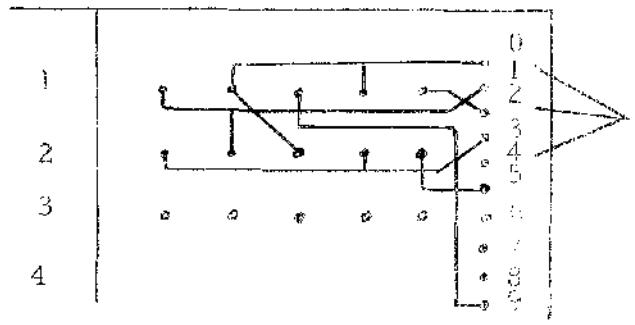
Application Γ'_{15} : elle fait correspondre à tout nombre N de E_M le nombre N + 1, si N = 32.767 elle lui fait correspondre zéro.

I - 2. EVOLUTION DE LA PROGRAMMATION DES MACHINES

Les premiers calculateurs électroniques étaient à "programme câblé". Ces machines possédaient un tableau de connexions qui permettait d'afficher le programme au moyen de fils de connexion à fiches: c'est l'équivalent du ruban programme de la machine A.

Le principe était le suivant:

Une instruction correspondait à une ligne. En marge une colonne de plots émetteurs d'impulsion numérotés 0 à 9 permettait de codifier l'instruction de chaque ligne.



Les instructions étaient éxécutées dans l'ordre 1, 2, 3, etc... des lignes jusqu'à la fin du tableau et la machine revenait à l'instruction 1

La taille de ces tableaux ne permettait pas de traiter des problèmes importants exigeant un grand nombre d'instructions. L'apparition des machines à programme par cartes fut la première amélioration

Le programme était perforé sur carte (une instruction par carte par exemple) et les cartes étaient lues par le lecteur de cartes de la machine A chaque fois qu'une instruction était lue, le calculateur l'exécutait.

Cette solution permettait d exécuter des programmes beaucoup plus importants, le nombre de cartes n'étant pas limité; elle présentait encore des inconvénients: la lecture des cartes était relativement lente, la vitesse de calcul risquait d'être diminuée D'autre part, les retours en arrière dans le programme étaient impossibles (comme pour la machine A), ce qui ne permettait pas d'effectuer les calculs de type itératif qui sont des procédés de calcul fréquemment utilisés.

Les calculateurs modernes sont à programmes enregistrés, c'est-à-dire que le programme est introduit dans la mémoire et y est conservé à volonté.

La mémoire est structurée en cellules ou "mots" numérotés de 0 à N. ces numéros sont les "adresses" des mots. Chaque mot peut contenir une information à traiter ou une instruction.

Le programme est introduit par carte dans la mémoire du calculateur, les instructions étant rangées dans des mots d'adresses croissantes consécutives. a, a + 1, a + 2, etc.. L'exécution du programme ne débute que lorsque l'introduction est terminée, en commençant par la première adresse a, puis a + 1, a + 2, etc... Le ruban programme et la mémoire de la machine A sont donc regroupés. Cette solution présente le maximum de souplesse.la séquence normale selon laquelle les instructions sont exécutées peut être modifiée grâce à des instructions provoquant des retours en arrière ou des sauts dans le programme.

Certains calculateurs industriels sont encore à programme câblé pour des raisons de sécurité: les calculateurs industriels sont destinés à contrôler l'évolution d'un phénomène physique en temps réel, il est donc nécessaire d'avoir une très grande fiabilité: les programmes câblés ne risquent pas d'être perturbés, contrairement aux programmes enregistrés qui peuvent être détruits par suite d'une erreur de manipulation de la machine, ou du mauvais fonctionnement d'un élément.

I - 3. EVOLUTION DES LANGAGES DE PROGRAMMATION

I - 3. 1. Codification numérique.

Nous avons déjà dit que généralement les calculateurs utilisaient la codification binaire.

L'instruction qui correspond à la capacité d'un mot de la mémoire comprend deux zones principales:

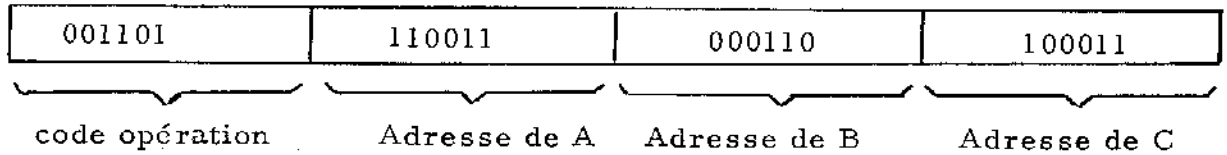
a) La zone type d'opération : elle caractérise le type de l'opération élémentaire que le calculateur doit effectuer.

Exemple: supposons que cette zone comprenne 6 positions binaires: on pourra avoir 64 codes opération différents: dont l'addition, soustraction, multiplication, etc. . . La codification de l'addition sera par exemple:001101.

b) La zone adresse. Elle sert à indiquer les adresses des mots contenant les opérandes, c'est-à-dire les informations sur lesquelles porte l'opération.

Exemple: supposons l'instruction à trois adresses. Si l'on veut effectuer l'opération A + B = C, on écrit dans la partie adresse les adresses des mots contenant: les nombres A, B, C.

En reprenant le code opération du paragraphe a), l'instruction s'écrit;



La longueur de la zone adresse dépend du nombre maximum de mots de la mémoire. De nombreuses machines peuvent avoir une capacité de 32. 768 mots (et certaines plus) ce qui nécessite 15 positions binaires ($32.768 = 2^{15}$) . Dans ces conditions, pour une machine à trois adresses il faut une zone de $3 \times 15 = 45$ positions binaires, à laquelle vient s'ajouter la zone type d'opération.

Le prix de revient d'une telle machine serait trop élevé; il est donc nécessaire de trouver un compromis: l'instruction à une adresse sera de la forme

code opération adresse

les deux autres adresses sont implicites.

La programmation dans cette codification est un travail long et pénible, c'est pourquoi dès les premières machines cette codification a été remplacée par une codification décimale où le nombre binaire est remplacé par le nombre décimal équivalent:

Exemple : l'instruction précédente s'écrit:

Cette représentation s'adapte particulièrement bien aux machines utilisant le code décimal codé binaire (code où chaque chiffre décimal est représenté par son équivalent binaire).

Pour les machines utilisant la numération binaire une traduction est nécessaire.

Ce langage présente encore de nombreux inconvénients: il faut apprendre la codification numérique de chaque type d'opération, les adresses de chaque instruction et des opérandes doivent être fixées dès le début de l'écriture du programme, et les modifications ultérieures de ces adresses sont extrêmement difficiles. La codification symbolique a apporté une nouvelle amélioration.

I - 3.2. Codification symbolique.

Les instructions ont toujours la même articulation, on retrouve toujours les mêmes zones. Mais, à chaque instruction en langage machine correspond une instruction symbolique.

L'instruction permettant d'effectuer l'opération A + B = C, où les trois nombres A, B, C, seront rangés dans des mémoires, s'écrira par exemple:

ADD, A, B, C,

A, B, C, servent à représenter des adresses de la mémoire ; on lit donc:

additionner le contenu du mot d'adresse A au contenu du mot d'adresse B, ranger le résultat dans le mot d'adresse C.

Il n'est plus nécessaire de numéroter les instructions: il suffit de les ranger dans l'ordre où l'on désire les voir exécuter.

La machine ne peut comprendre directement un tel langage: elle a donc en mémoire un programme de traduction. Ce programme est souvent appelé "assembleur".

Malgré cette codification symbolique, on travaille toujours sur les instructions machines: il est toujours nécessaire de découper les problèmes à traiter en une succession d'opérations élémentaires pour écrire le programme.

I - 3. 3. Langages symboliques de programmation.

C'est la dernière étape. Ces langages sont voisins du langage usuel et n'ont plus qu'un lointain rapport avec le langage de la machine.

Par exemple si l'on veut calculer $x = \frac{a \cdot (a+b)}{24\ d_i}$

on écrira : x = (A * (A + B)) / (24 * D (I))
(les parenthèses servent à éviter les ambiguités).

Un tel langage doit être traduit par la machine en son propre langage avant d'être exécuté: le programme qui fait cette traduction s'appelle un "compilateur".

Grâce à ce type de langage, la programmation est grandement facilitée et rendue accessible même à des personnes qui ne sont pas des spécialistes des calculateurs. Il existe de nombreux langages symboliques: les plus largement utilisés sont:

ALGOL, FORTRAN pour les applications scientifiques

COBOL pour les applications comptables.

I - 4. RAPPEL DES COMPOSANTS D'UNE MACHINE

Nous allons maintenant décrire une machine théorique beaucoup plus proche de la réalité physique que la machine A.

Nous avions la possibilité de prendre en exemple une machine réelle ou de définir une machine purement fictive.

Nous avons pris un compromis entre les deux solutions et nous allons décrire une machine fortement inspirée de la série IBM 360. Cette solution nous permettra de décrire une machine suffisamment générale sans pour autant descendre dans les détails inutiles.

C'est sur cette machine que nous travaillerons dans la suite du cours.

Le schéma comprend les éléments logiques suivants:

a) Pupitre de commande

C'est l'élément qui permet à l'opérateur de surveiller la marche de la machine et, éventuellement, d'intervenir (pour les départs et les arrêts en particulier).

b) Eléments de commande

Ce sont les éléments qui interprètent les instructions du programme et commandent l'exécution des opérations effectuées par les autres éléments de la machine.

Les registres principaux sont:

- le registre instruction: qui contient l'instruction provenant de la mémoire: c'est dans ce registre que l'instruction est interprétée.

- le compteur ordinal: qui contient l'adresse de l'instruction suivante à exécuter.
  Le contenu de ce compteur est modifié automatiquement après l'alimentation d'une instruction dans le registre instruction..

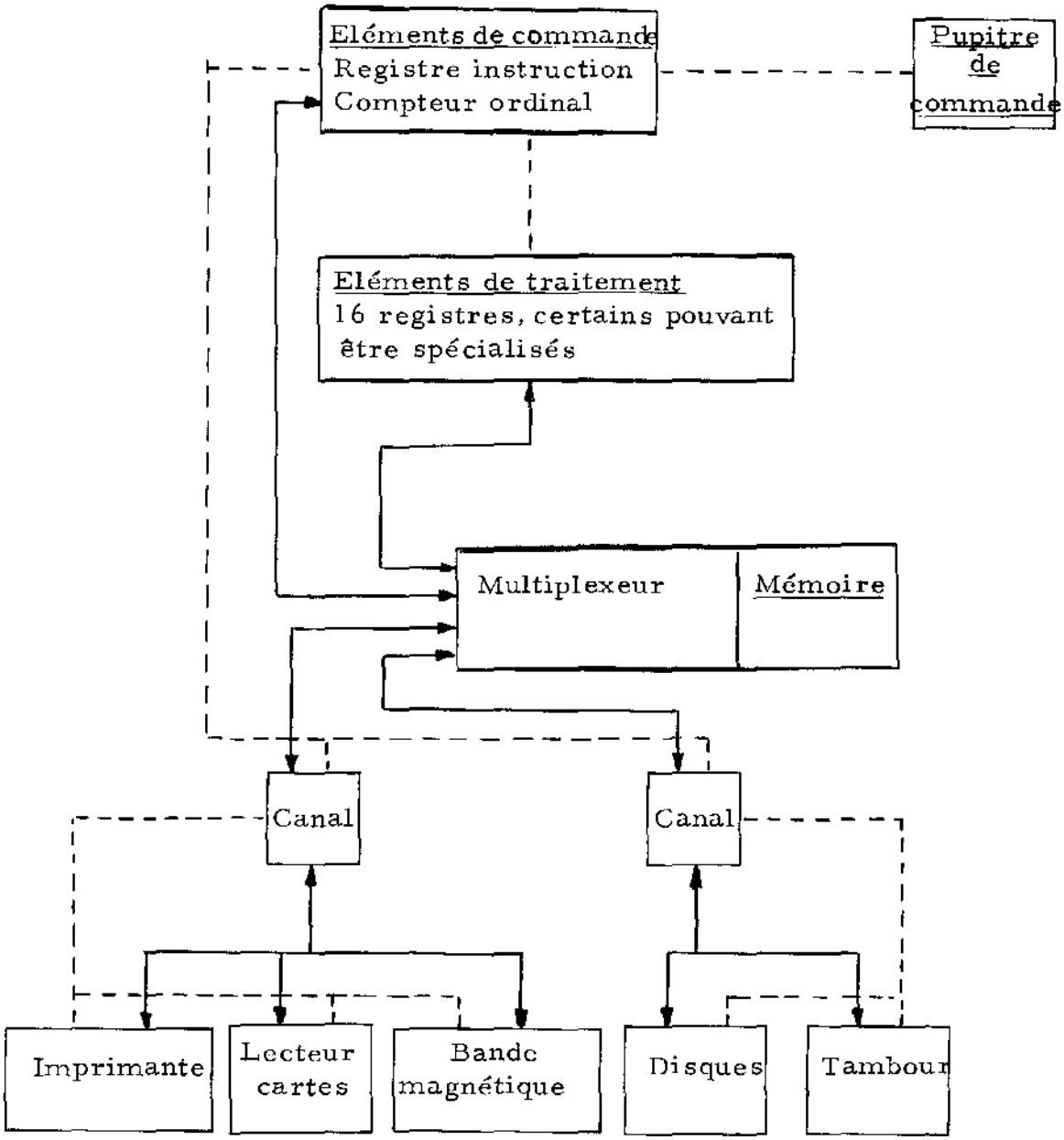
c) Eléménts de traitement

Ce sont les éléments qui exécutent les opérations internes du calculateur: nous appelons opérations internes celles qui ne mettent pas en jeu les éléments périphériques: mémoires auxiliaires, imprimante, etc...

Trois registres au moins sont nécessaires:

- l'accumulateur AC: dans lequel s'effectue la majorité des opérations.

- le registre MQ: dans certains cas il est nécessaire de disposer d'un registre supplémentaire; c'est le registre multiplicateur - quotient MQ- Son nom provient du fait qu'il est utilisé avec l' AC en particulier pour les multiplications et divisions.



- le registre d'index. il est utilisé pour modifier la partie adresse des instructions.

Toutefois, pour des raisons de commodité, les machines modernes sont dotées d'un nombre de registres supérieur à trois, pouvant jouer chacun un de ces rôles avec des spécialisations particulières.

Nous noterons la machine théorique de 16 registres.

d) Mémoire

Nous allons préciser son rôle: c'est une plaque tournante pour tous les mouvements d'information entre les différents éléments de la machine: supposons que le calculateur doive effectuer l'opération A + B = C , A et B étant deux nombres sur une bande magnétique, et C devant être rangé sur bande magnétique

L'opération se découpera en plusieurs temps.

1/ transfert de A et B de la bande magnétique en mémoire centrale,

2/ calcul de A + B - C par les éléments centraux et rangement de C en mémoire centrale,

3/ transfert de C sur bande magnétique.

Comme la mémoire ne peut desservir qu un élément à un instant donné, nous avons mis en évidence sur le schéma un élément complémentaire, le multiplexeur, qui assure les connexions des circuits d'information en fonction des demandes des éléments.

e) Canaux d'entrée-sortie

Le canal est un élément de liaison entre les éléments centraux de la machine que nous venons de voir et les éléments périphériques d'entrée-sortie. C'est un véritable petit calculateur spécialisé qui décharge les éléments centraux des opérations d'entrée-sortie consistant essentiellement en échange d'informations entre éléments périphériques et mémoire. Nous préciserons ultérieurement leur fonction.

f) Eléments périphériques d'entrée-sortie

Ils comprennent soit les mémoires auxiliaires: disques magnétiques, tambours magnétiques, bandes magnétiques, soit les éléments qui permettent de communiquer avec la machine: lecteurs de cartes, perforateurs de cartes, lecteurs de bandes perforées, imprimantes, machines à écrire, etc .