M. ASSABGUI

le langage d'assemblage

l'assembleur OS/360

DUNOD UNIVERSITÉ

dégager les définitior
mes utilisés. Mais c'e
posé sur les algorit
vons exprimer notr
fondamentale. L'autet
exposé de l'exemple
avec un jeu de carte
rithme de cette réu
la matérialisation par
superposant une mat
tion informatique (s
positionnement, etc.)
semble typique d'une
ment poussée. Une r
fondie nous montrer
étudiée consiste en
permutation de 52 ob
tation circulaire. Les
support matériel pou
certain nombre d'op
de réaliser cette véri
lyse doit permettre la
l'application et l'acce
gique. Si cela n'est
« qui organise le tra
fort de la bonne utilis

Peut-on imaginer que
le boulier sert de sup
la réalisation des calc
les algorithmes soient
vant la référence à

LE LANGAGE D'ASSEMBLAGE – L'ASSEMBLEUR OS/360

par M. Assabgui, Editions Dunod, Paris 1972, Collection «Dunod Université», 229 pages, broché.

Bien que très utilisé, l'assembleur 360 n'a pas fait l'objet comme Cobol de nombreuses publications. Si l'on en trouve une présentation sommaire dans certains livres à caractère général, il faut reconnaître que son analyse détaillée a été quelque peu négligée. Cela s'explique d'ailleurs par le fait qu'il s'agit d'un produit IBM, auquel les programmeurs peuvent se référer au moyen de la documentation du constructeur.

Cette documentation, dont le moins que l'on puisse en dire, est qu'elle est particulièrement aride, presque illisible, n'est malheureusement abordable qu'au prix d'importants efforts.

C'est pourquoi l'ouvrage de M. Assabgui nous paraît intéressant. S'il ne contient pas la "noble intégralité" du langage, il n'en constitue pas moins un essai louable de simplification.

Notons que les pages sur l'assembleur 360 sont précédées dans ce livre par une importante analyse du langage machine et de l'assemblage en général qui éclaire les processus particuliers du 360 sous OS.

LA COMPTABILITI
par Bernard VILAINE

Editions Dunod
92, *rue Bonaparte, Paris-6e*
1 *volume* 120 x 180 - 115 *pages* - 9,30 *F*

La banque Rothschild, décidément, s'intéresse beaucoup au temps réel. L'auteur, un directeur de cette banque, tente de nous convaincre de l'intérêt d'une comptabilité en temps réel, malgré les procédures coûteuses que celle-ci peut entrainer et des risques de décisions hâtives basées sur les variations à court terme que le système peut faire apparaitre.

La saisie directe des données, la « réhumanisation » des tâches élémentaires nécessaires à la qualité de l'information qui sera traitée sont bien mises en valeur. « L'amélioration de la qualité des données de base est le préalable indispensable à la mise en place d'une gestion en temps réel. » Le temps réel devrait, en fait, permettre d'éponger le bavardage de volumineux listings qui « agressent » les gestionnaires ; listings qu'il s'agit soit d'exploiter, péniblement, soit de classer... Nous notons toutefois une partie de la conclusion du travail :

« L'apprentissage des techniques conversationnelles doit, plus rapidement, permettre — sur la base des informations de synthèse recueillies mensuellement — de prolonger les travaux comptables par l'introduction de diverses instructions complémentaires souples selon les modalités proches du time sharing et de sélectionner ainsi, en fonction de situations concrètes, les informations à éditer. »

'ECTION
.LATIONS

on,

pages - 1971 -

pays aucun ate-
'est protégé de
l'incendie. Voilà
nt à point, d'au-
en langue fran-
problème, pour
nnait à ce jour
ues.
vec l'Association
contre l'incendie
tion britannique,
proche de l'ex-
ement technique.
ajouter quelques

rez-de-chaussée
archives enregis-
rchitectes, à la
sensibilisation à
lernières années,
cupation du rez-
ur et à son envi-

mencement d'in-
en de traitement
ris, le personnel
as chargés des
disques et bandes magnétiques en cours d'utilisation, plusieurs minutes avant l'arrivée des pompiers, et ceci selon d'heureuses consignes rigoureuses édictées à la suite d'un récent exercice de prévention,

— enfin les chefs d'entreprise ignorent encore trop souvent qu'ils peuvent obtenir des rabais importants ou éviter des majorations de primes d'assurances, et par suite bénéficier des limitations d'impôts correspondants. Il leur suffit simplement de consulter les fascicules de la tarification analytique de l'Assemblée plénière des sociétés d'assurances, 11, rue Pillet-Will, Paris-9e - 824-91-70. La prévention, si elle est une cause de dépenses, constitue, elle aussi, un élément important de gestion trop souvent négligé (cf. Informatique et Gestion n° 20 - août-septembre 1970).

G. L.

LE LANGAGE D'ASSEMBLAGE L'ASSEMBLEUR OS/360
par M. ASSABGUI

Editions Dunod
92, *rue Bonaparte, Paris-6e*
1 *volume* 155 x 240 - 229 *pages* - 1972 - 39 *F*

Le premier ouvrage traitant de ce sujet dans la langue française. Depuis le début, introduction élémentaire à la programmation, jusqu'à la fin, l'exposé est rédigé dans un français rigoureux, très clair ; il ne devrait pas rebuter les candidats-

Livres

systèmes auxquels il est destiné. L'auteur a adopté le plan suivant :

- le langage machine (format des caractères et mode d'adressage, représentation des informations) ;
- la programmation en langage d'assemblage ;
- le langage d'assemblage (adressage symbolique, sections de contrôle, pseudo-instructions, macro-instructions et macro-procédures) ;
- le processus d'assemblage ;
- la programmation des entrées et sorties en langage d'assemblage (organisation élémentaire des enregistrements, traitement des fichiers, accès aux enregistrements) ;
- l'assembleur OS/360 ;
- annexes (cartes de commande, messages d'erreurs de l'assembleur F et caractéristiques de l'assembleur H, liaisons dynamiques de programmes, communications avec le superviseur).

COHERENT OPTICAL COMPUTERS

par KENDALL PRESTON Jr

Mc Graw-Hill Book Company G.m.b.H. 4 Düsseldorf
Graf-Adolf - Strasse 43 - Germany
1 volume 145 x 230 - 311 *pages* - $ 16,50

A notre connaissance, le premier ouvrage très technique traitant d'un sujet auquel de nombreuses personnes portent beaucoup d'intérêt ; et ce par un ancien chercheur des Bell Telephone Laboratories. Un chapitre est consacré aux techniques digitales de ce domaine. Le dernier chapitre traite des applications les plus importantes et certaines ont été réservées en primeur au lecteur, par certains services militaires.

Nous ne sommes pas allés regarder sous l'intégrale, mais cet ouvrage nous semble d'une grande valeur.

DICTIONNAIRE DE L'INFORMATIQUE

par Claude CAMILLE et Michel DEHAINE

Volume 1 - anglais-français - 277 pages - 140 x 220 x 28 F.
Volume 2 - français-anglais - 247 pages - 140 x 220 - 24 F.
Bordas, 30, rue Saint-Sulpice, Paris-6°

Ce dictionnaire semble l'un des mieux conçus du genre, malgré quelques imperfections et oublis inévitables. Mais, à vrai dire, avec l'évolution rapide des techniques et de leur jargon, on ne voit pas très bien finalement à qui peut servir un tel outil de travail. Les spécialistes n'y trouveront pas, bien souvent, parce que trop nouveau, le terme qu'ils découvrent au détour d'une brochure technique. Les généralistes risquent, eux, d'être induits en erreur faute de connaitre suffisamment le contexte.

SIMPLIFICATIONS ADMINISTRATIVES ET PRODUCTIVITE

On voudra bien changer le numéro du CCP de la Documentation Française indiqué dans notre numéro de mai, page 21. Il fallait lire 9060-98.

Le langage d'assemblage. L'assembleur OS360, par M. ASSABGUI, Éd. Dunod Université, Paris.

Dans le cadre de l'ordinateur 360, la première partie de cet ouvrage présente les langages machine et plus spécialement des notions d'adressage, de représentation interne des informations et des instructions. Après avoir présenté les langages d'Assemblage, la seconde partie est consacrée à la description des possibilités qu'ils offrent. Elle considère l'emploi des symboles, le sectionnement des programmes, en vue de l'assemblage indépendant des différents modules, les pseudo-instructions de gestion (d'adresses, de mémoire), de contrôle..., les directives d'assemblage conditionnel, les macro-instructions et macro-procédures. A partir de ces notions, la troisième partie présente des méthodes permettant de réaliser le processus d'assemblage. La programmation des entrées/sorties, en langage d'assemblage, est abordée dans la quatrième partie avec les organisations élémentaires des enregistrements, l'étude du programme central des entrées/sorties, le traitement des fichiers et la présentation des accès aux enregistrements.

Enfin, l'Assembleur OS360 (niveau F et même H) est présenté dans une cinquième partie. L'ouvrage s'achève par une suite d'annexes sur les cartes de commandes, les messages d'erreurs, les liaisons dynamiques entre programme sous contrôle de superviseur, et diverses tables... l'ensemble présentant un très vif intérêt pour l'utilisateur.

D'ailleurs, cette présentation se voulant utile, pratique et pédagogique, trois emplois sont proposés pour cet ouvrage. Si on cherche un manuel :

— de référence de l'Assembleur sous OS360, alors :

• les initiés pourront utiliser seulement la partie 5 et les annexes,

• les non-initiés utiliseront en plus les présentations générales en tête des parties 1 et 2,

— d'initiation à la programmation en langage d'assemblage (sans vouloir apprendre particulièrement l'essembleur OS360), ils se limiteront aux parties 1 à 4. Ils se souviendront toutefois que la 3e partie fournit un aperçu du processus d'assemblage et que la 4e partie est destinée à les préparer à la lecture des brochures de constructeurs consacrées à la réalisation des Entrées/Sorties en langage d'assemblage.

On peut reprocher à cet ouvrage d'être orienté vers une machine et un système particuliers. L'auteur l'a bien ressenti, et devant les trois alternatives suivantes, qu'il avait, pour présenter chaque notion :

— la recherche d'exemples de réalisation choisis,

— la présentation en parallèle de plusieurs systèmes retenus comme référence,

— l'exposé des possibilités d'un système particulier,

il a retenu la troisième méthode. Il la justifie par un souci pédagogique. Son choix s'est porté sur l'assembleur 360 parce qu'il est à la fois très perfectionné et très diffusé... et peut-être aussi parce qu'il disposait de cet outil.

G. BAZERQUE
(Toulouse)

INSTRUCTIC

Modalités de publication

Tout auteur désireux de sou
revue est prié d'en faire parvenir
suivante :

Secrét
Universit
7577

Tout article doit être accompa
indication écrite de l'auteur pr
recherche) sous lesquels il désire

Les auteurs étrangers ont le c
important : aucun article n'e
un rédacteur en chef.

Liste provisoire des rubriques

— Méthodologie et technolog
— Langages de programmati
— Analyse et conception des
— Banques de données.
— Conception de machines.
— Architecture des machines.
— Systèmes graphiques.
— Modèles, simulations, évai
— Intelligence artificielle.
— Informatique appliquée au
— Informatique appliquée au

M. ASSABGUI

le langage d'assemblage

l'assembleur OS/360

DUNOD UNIVERSITÉ

Avant-propos

Programmer en langage d'assemblage c'est écrire les instructions-machine en représentant leur code interne par des symboles, sous une forme concise mais facilement compréhensible.

La programmation en langage d'assemblage utilise un ensemble de noms symboliques, d'opérateurs, de délimiteurs et d'indicateurs : le vocabulaire du langage.

Les règles qui définissent l'écriture des instructions à l'aide de ce vocabulaire constituent la syntaxe du langage.

Ces règles sont nécessairement nombreuses et complexes parce que marquées par les caractéristiques technologiques des machines ; leur apprentissage lent et progressif : la pratique aisée d'un langage orienté vers une machine exige une connaissance profonde, une habitude de cette machine.

Un programmeur expérimenté, pratiquant avec aisance un ou plusieurs langages d'assemblage, peut cependant en apprendre un nouveau dans de très courts délais ; de l'ordre de quelques jours à une semaine s'il s'agit d'une machine simple ou semblable à celles qu'il a déjà utilisées. Il se distinguera durant son étude du nouveau langage, du programmeur non expérimenté, par son attitude « active ». Prenant comme points de repère les fonctions qui lui sont familières dans les langages qu'il connaît, il se posera au sujet de chacune de ces fonctions les deux questions suivantes : existe-t-elle dans le nouveau langage ? Si oui, sous quelle forme ? Il parcourra ensuite le manuel décrivant l'outil à maîtriser afin de savoir par la suite, lorsqu'il programmera, où trouver les informations concernant le détail de l'écriture de chacune des instructions du langage. Il n'aura à fournir de véritable effort de compréhension que celui nécessaire à l'appréhension des fonctions nouvelles.

Cette manière de procéder présuppose une certaine normalisation des fonctions et de la forme des instructions des langages d'assemblage. Elle présuppose également une formation du programmeur lui permettant de comparer et d'abstraire.

Le premier objet de ce livre est la présentation des principales fonctions des langages d'assemblage. Cette présentation, rendue possible à l'heure actuelle par la normalisation des fonctions des assembleurs, nous paraît nécessaire car presque tous les ouvrages consacrés à la programmation au niveau des instructions-machine décrivent surtout ces dernières. Ils ne rendent compte des possibilités des assembleurs que d'une manière rapide et superficielle.

Nous pensons qu'il est possible de parler du langage d'assemblage en tant que tel et qu'il est pédagogiquement souhaitable que les langages d'assemblage évolués soient présentés le plus indépendamment possible des langages-machine qui les sous-tendent.

Notre description du langage d'assemblage du système OS/360 illustre ce principe. Elle pourrait servir de base à un modèle descriptif des langages d'assemblage.

* * *

La première partie de cet ouvrage est une introduction générale au langage-machine. Les notions d'adressage et de représentation interne des informations y sont en particulier expliquées. Les caractéristiques des ordinateurs IBM 360 y sont signalées explicitement. Celles des autres ordinateurs, que nous avons jugé nécessaire d'exposer au lecteur débutant pour lui permettre de disposer de points de comparaison, figurent dans le texte en italiques.

La deuxième partie expose les principales possibilités des langages d'assemblage actuels. Cette partie, comme la précédente, souligne les propriétés des systèmes 360 dans le cadre d'une présentation générale.

Les règles et les contraintes qui définissent un langage de programmation ne se justifient pas à l'aide de considérations purement fonctionnelles. Elles ne s'expliquent d'une manière satisfaisante que si l'on connaît la façon dont les instructions qu'elles autorisent sont analysées et prises en charge par l'ordinateur. La troisième partie est donc consacrée à la description du processus d'assemblage. Nous avons essayé d'exposer ce que doit savoir tout programmeur d'un certain niveau du processus d'assemblage, afin d'être en mesure d'exploiter à bon escient toutes les possibilités de l'assembleur dont il dispose.

La quatrième partie traite des principes de la programmation des entrées et sorties en langage d'assemblage.

Les opérations d'entrées et sorties sont commandées, lorsque l'on programme en langage d'assemblage, au moyen d'appels de sous-programmes eux-mêmes générés par des appels de macroprocédures. La description de l'usage de ces macroprocédures est obligatoire dans une présentation comme la nôtre bien qu'elle ne relève pas de l'étude du « langage ». Le programmeur d'application ne dispose en effet d'aucun autre moyen de réaliser des entrées et sorties que celui de l'appel de ces macroprocédures. Du point de vue du modèle descriptif, les règles définissant leur mode d'emploi et l'écriture de leur ligne d'appel pourraient être incluses dans la syntaxe du langage. Nous ne sommes toutefois pas entrés dans le détail des possibilités existant dans le système OS/360. L'examen de toutes les options réalisables dans le système OS/360 et de leur contrôle par l'utilisateur eût démesurément allongé l'ouvrage. Notre introduction aux entrées et sorties en langage d'assemblage est destinée au lecteur débutant à qui elle fournira les éléments lui permettant d'aborder sans trop de difficulté les manuels des constructeurs traitant de la question.

Nous voulons justifier pour finir le choix que nous avons fait de présenter d'une manière complète et autonome l'assembleur de l'ordinateur IBM 360 dans la cinquième et dernière partie de l'ouvrage.

Décrire les possibilités d'une classe de langages ou de systèmes de programmation sans ponctuer sa description d'exemples ne peut être envisagé dans un ouvrage pratique destiné à être utilisé comme ouvrage de référence. On a dès lors le choix entre trois méthodes. La première consiste à appuyer ce que l'on avance à l'aide de l'exemple de réalisation qui l'illustre le mieux. La seconde met constamment en parallèle plusieurs systèmes choisis a priori comme références. Ces deux approches d'une technique ont l'avantage de permettre d'en faire le tour d'une manière exhaustive. Elles ont néanmoins l'inconvénient d'être d'un abord difficile pour le débutant. Celui-ci souvent se découragera avant d'être rendu là où l'auteur avait voulu le conduire.

La troisième méthode consiste à s'en tenir à un système particulier et à exposer les fonctions de chacun de ses éléments à l'occasion de l'étude du modèle général.

Cette méthode a l'inconvénient de favoriser quelque peu le constructeur du système auquel on se réfère, car souvent, « c'est le système que je connais qui est le meilleur ».

Nous avons décidé de donner l'assembleur OS/360 en exemple pour deux raisons. D'abord parce que c'est un outil perfectionné comportant une grande partie des possibilités qui existent dans les assembleurs évolués actuels ; son étude permet donc de se faire une bonne idée de ces dernières.

La seconde raison de notre choix est la grande diffusion de l'ordinateur IBM 360. Le présent ouvrage est destiné en effet à tous ceux, et en particulier aux étudiants des Universités francophones et des CEGEP québécois, qui souhaitent apprendre à programmer dans le langage d'assemblage du 360, tout en s'initiant d'une manière plus fondamentale aux concepts utilisés dans les langages d'assemblage actuels.

La partie consacrée à l'assembleur OS/360 se présente comme un tout et peut servir de manuel de référence indépendamment des autres parties.

Signalons enfin que la version qui y est décrite est celle de l'assembleur « de niveau F » mais que les nouveautés de « l'assembleur H » (Version de juin 1970) y sont clairement indiquées au moyen de renvois, sous les paragraphes appropriés, à l'Annexe 2 qui les regroupe. Cette annexe contient une liste des erreurs correspondant à des restrictions propres à « l'assembleur F » avec une indication, à chaque fois que le cas se présente, de leur modification telle qu'elle apparaît dans « l'assembleur H ».

* * *

Dans le cadre de cette présentation trois plans de lecture peuvent être proposés :

Le lecteur connaissant un ou plusieurs langages d'assemblage et souhaitant utiliser le 360 pourra ne lire que la cinquième partie et s'en servir comme d'un simple outil de référence.

Le lecteur souhaitant s'initier à la programmation en langage d'assemblage tout en apprenant celui du système OS/360 devra lire la présentation générale des première et seconde parties en se reportant à la suite de chacun des para-

graphes introductifs de la seconde partie aux exemples des paragraphes correspondants de la cinquième.

Le lecteur enfin qui souhaite s'initier à la programmation en langage d'assemblage mais n'a pas besoin de connaître plus particulièrement le langage du système OS/360 pourra ne lire que les quatre premières parties.

Dans les deux derniers cas il est vivement recommandé au lecteur de ne pas aborder les chapitres consacrés à l'assemblage conditionnel et à l'utilisation des macroprocédures avant d'avoir parfaitement compris ceux consacrés à l'assemblage élémentaire. L'aperçu que nous donnons, dans la troisième partie, du processus d'assemblage lui permettra de vérifier qu'il a bien compris ce qu'est l'assemblage conditionnel. Le complément de la quatrième partie le préparera enfin à la lecture des manuels des constructeurs consacrés à la réalisation des entrées et sorties dans le contexte de la programmation en langage d'assemblage.

*
* *

Le présent ouvrage est le fruit d'un travail réalisé dans le cadre des activités des équipes « systèmes » et « compilateurs » de l'Institut de Mathématiques Appliquées de Grenoble. Je remercie vivement la Direction de l'IMAG de m'avoir permis de l'entreprendre et mes collègues de l'IMAG pour les remarques qu'ils m'ont communiquées et l'aide qu'ils m'ont apportée au cours de nos nombreuses discussions.

Monsieur O. LECARME, Professeur-assistant à l'Université de Montréal, a bien voulu relire les deux versions manuscrites successives. Je lui en suis très redevable. Je le remercie en particulier de ses précieux conseils en ce qui concerne la terminologie adoptée.

L'édition de cet ouvrage est due en grande partie à l'initiative du groupe des Professeurs d'Informatique du CEGEP d'Ahuntsic de Montréal et aux encouragements de la Direction générale de Dunod-Québec. Je leur en suis très reconnaissant et espère que le présent livre répond à leur attente.

Le Professeur J. ARSAC me fait l'honneur d'accueillir l'ouvrage dans sa collection. Je l'en remercie vivement.

Monsieur J. P. BERGER, ingénieur à la Compagnie Internationale pour l'Informatique, m'a aidé à corriger les épreuves. Je le prie de trouver ici l'expression de mes sincères remerciements.

Table des matières

Introduction générale

1.1 LE LANGAGE MACHINE

1.1.1 LES INSTRUCTIONS MACHINE. INTRODUCTION ÉLÉMENTAIRE

L'instruction machine est, du point de vue programmation, l'unité de travail de l'ordinateur.

C'est, en mémoire, un profil binaire adressable comprenant une **zone d'opération** et une **zone d'arguments**.

Dans la zone d'opération se trouve un **code d'opération** qui détermine la nature de l'instruction, c'est-à-dire, l'action à entreprendre, le travail élémentaire à réaliser.

Dans la zone d'arguments se trouvent représentés un ou plusieurs **arguments**. Ces arguments définissent, nous verrons plus tard de quelle manière, sur quoi porte la commande d'opération correspondant au code d'opération.

Un programme est constitué d'une suite d'instructions dont l'enchaînement se traduit par le fait qu'elles sont dans des emplacements consécutifs de la mémoire. Certaines instructions dites de **branchement** permettent de rompre l'ordre d'enchaînement normal et d'effectuer un **saut** ou branchement à un emplacement de la mémoire qu'elles indiquent d'une manière explicite.

Lorsqu'ils sont soumis à la réalisation d'une condition logique, résultant elle-même de l'exécution antérieure d'une autre instruction, les branchements sont dits **conditionnels** et permettent de faire prendre par l'ordinateur des décisions logiques.

Lors de l'exécution d'une séquence ne comportant pas de saut, les instructions sont extraites l'une après l'autre de la mémoire par **l'unité de commande** et exécutées.

L'unité de commande conserve et met à jour, au fur et à mesure qu'elle exécute les instructions du programme, un compteur qui lui indique à chaque appel, l'emplacement de l'instruction à appeler. C'est le **compteur ordinal** intitulé aussi **registre d'adresse d'instruction**. Le compteur ordinal pointe constamment sur l'instruction qui suit celle en cours d'exécution. Le pro-

grammeur n'y a pas accès directement. C'est un registre de travail interne à l'unité de commande. On modifie toutefois son contenu lorsque l'on écrit des instructions de branchement. Celles-ci ont justement pour effet de rompre l'évolution séquentielle de la valeur du compteur ordinal, pour le faire pointer sur l'emplacement de mémoire désigné par le programmeur (dans la zone d'arguments).

Nous allons maintenant expliquer ce qu'est pour la machine un « emplacement de mémoire » : le prochain paragraphe est consacré aux techniques et aux dispositifs d'adressage.

1.1.2 ADRESSAGE

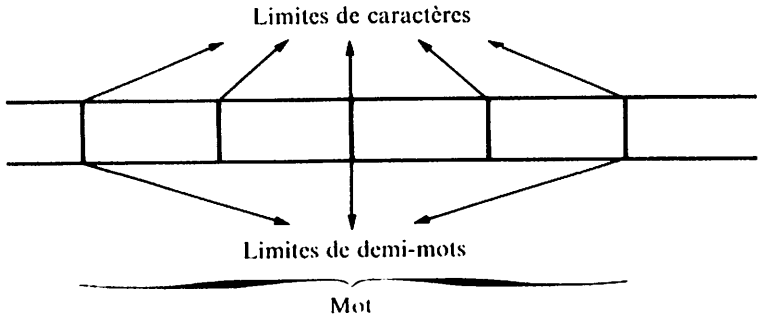
1.1.2A Format des caractères et mode d'adressage

Par « format des caractères » nous désignons le nombre d'éléments binaires nécessaires à la représentation interne, en mémoire, des caractères.

La mémoire de l'ordinateur peut se concevoir comme une succession d'éléments binaires, les **bits**, groupés dans une suite de cases numérotées séquentiellement de 0 à une **dimension** finie. Une case donnée est repérable (voir ci-dessous pour plus de précision) par son numéro de séquence : son **adresse**.

La dimension en nombre de bits d'une case élémentaire correspond au format des caractères tel que défini ci-dessus. On s'oriente actuellement vers l'adoption de formats à huit bits (c'est le cas du 360). Les formats à six bits sont également courants.

Les cases élémentaires peuvent être groupées. Elles le sont dans le 360 par groupes de quatre pour constituer des **mots**. Elles peuvent également être groupées par deux ou par huit. On parle alors de **demi-mots** et de **double-mots** respectivement.



Lorsque la numérotation des cases évoquée ci-dessus est effective au niveau des formats des caractères, on dit que la **mémoire est adressable par caractères**.

Si la numérotation s'effectue au niveau des mots, on dit que la mémoire est **adressable par mots**. La résolution s'arrête alors aux mots.

*Dans beaucoup de machines coexistent les deux modes d'adressage par mot et par caractères. On parle quelquefois alors d'**adressage de mot** et d'**adressage de caractère** d'un élément d'information.*

Certaines machines autorisent également un adressage par demi-mot.

Nous avons décrit les instructions-machine comme des « profils binaires adressables ». Elles sont de plus souvent obligatoirement cadrées dans des limites de mots. Dans le 360, elles le sont dans des limites de demi-mots.

1.1.2B Dispositifs et mécanismes d'adressage

a) L'adressage simple

Une adresse peut être explicite dans la zone d'arguments d'une instruction. Elle est alors dite **simple**. Si une telle adresse est représentée (voir 1.1.4*b* pour la représentation) par la valeur 400 dans la zone d'arguments et si l'adressage se fait par caractères, on a tout simplement une référence au quatre centième caractère de la mémoire.

L'adressage simple est loin d'être le seul type d'accès à une zone déterminée de la mémoire. Il est possible d'indiquer à l'unité de commande une adresse en la décomposant en plusieurs éléments représentés chacun par un constituant de l'argument dans la zone d'arguments.

Les dispositifs d'adressage qui autorisent l'accès à une adresse à partir d'une telle composition n'existent pas dans tous les ordinateurs. Ils correspondent à des possibilités de programmation diverses qui, en général, déterminent la physionomie des programmes de base des ordinateurs qui en sont pourvus. A cela s'ajoute évidemment l'intérêt qu'elles peuvent avoir en soi dans le cadre d'une programmation particulière. Leur absence ou leur présence dans un ordinateur est de ce fait une caractéristique essentielle de cet ordinateur.

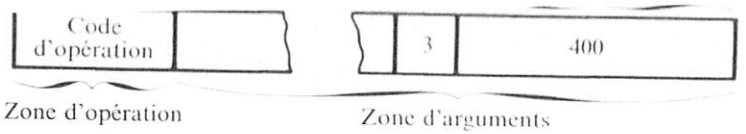
b) L'indexation

L'indexation est la première de ces possibilités. Elle consiste à associer à une adresse simple un **registre d'index** ou une **mémoire d'index** dont le contenu est ajouté à cette adresse au moment de l'exécution de l'instruction. L'adresse effective obtenue est une **adresse indexée**.

Le programmeur place dans le registre ou la mémoire d'index au cours d'une séquence qui précède l'instruction qui l'utilise, une valeur de son choix. Cette valeur peut résulter d'un calcul. Elle est généralement modifiée au cours de l'exécution du programme (*).

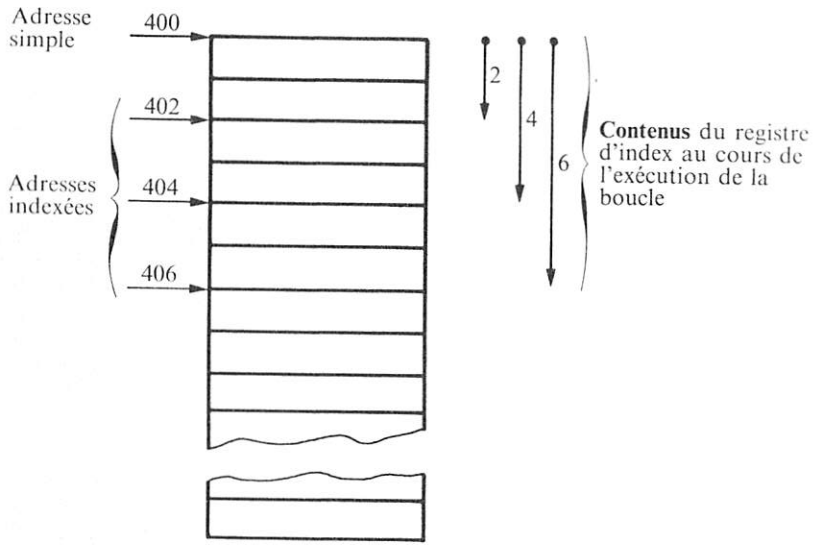
Dans l'exemple qui suit, nous avons supposé que le premier constituant de l'argument désigne, dans l'instruction considérée, un registre d'index identifié comme le « registre numéro 3 ».

(*) Une telle modification est dite **dynamique**. La valeur qu'elle affecte est également qualifiée de dynamique. Une valeur dynamique est une valeur qui varie au cours de l'exécution.



Nous avons également supposé que l'adresse figurant dans la zone affectée au second constituant de l'argument est l'adresse simple 400. Si à l'exécution de cette instruction le registre n° 3 contient la valeur 10, l'opération commandée porte alors sur la zone de mémoire d'adresse 400 + 10 = 410.

L'utilisation d'adresses indexées sert généralement à parcourir un tableau à l'intérieur d'une boucle. La valeur d'un compteur dans la boucle ou un multiple de cette valeur servent d'index et permettent d'accéder aux éléments consécutifs du tableau. On pourrait ainsi avoir comme valeurs d'index dans l'exemple ci-dessus les valeurs 2, 4, 6 etc. si on suppose qu'en 402, 404, 406 etc. sont rangées les données constituant le tableau à traiter.



Les dispositifs d'index existent sur presque toutes les machines, généralement en même temps que l'un des dispositifs décrits ci-dessous.

Le nombre des registres d'index varie d'un ordinateur à l'autre (1, 3, 7, 16, 128). Dans certains (cas du 360), les registres généraux peuvent servir occasionnellement d'index. Dans d'autres les registres d'index sont distincts qu'à des opérations simples de l'un des types : « chargement », « incrémentation » ou « interrogation de valeur ».

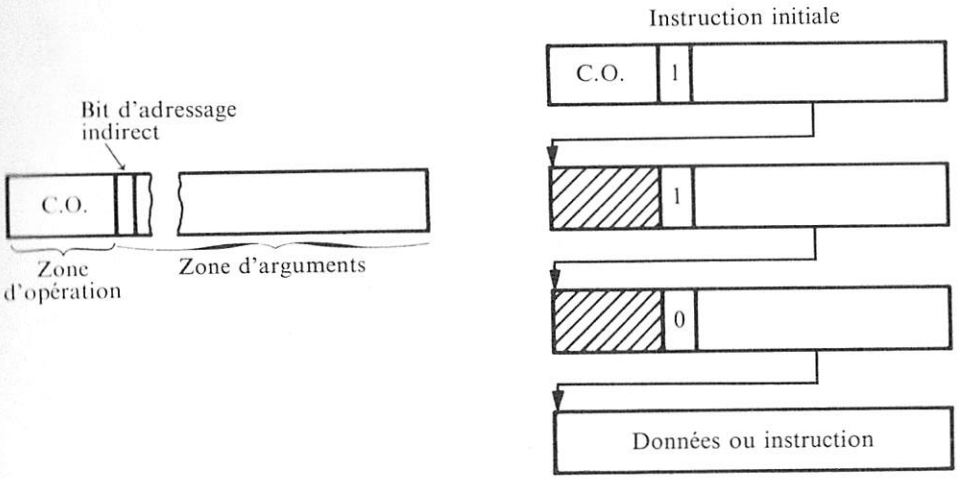
Dans certains ordinateurs tous les mots de la mémoire peuvent servir d'index. Dans d'autres enfin, une partie seulement des éléments de la mémoire peuvent servir d'index. L'adresse de la mémoire d'index utilisée est alors indiquée dans l'instruction. C'est une partie de l'adresse de l'instruction qui (lorsqu'elle n'est pas nulle) peut être considérée comme un constituant autonome au même titre que les registres d'index.

c) L'adressage indirect

L'adressage indirect *est la deuxième technique de construction dynamique d'adresses que nous étudierons. Elle consiste à placer dans la zone d'arguments l'adresse d'un emplacement qui, à l'exécution, contient à son tour, dans sa zone d'arguments, une seconde adresse. Cette seconde adresse constitue l'adresse effective et c'est son contenu qui est l'objet de l'instruction. L'emplacement désigné par cette « adresse effective » peut contenir des données ou une instruction. Il peut aussi quelquefois contenir dans sa zone d'arguments une adresse servant elle-même à accéder à une nouvelle zone de mémoire.*

Autrement dit, l'adressage indirect peut parfois se réaliser à plusieurs niveaux.

Dans les machines comportant un tel dispositif, un indicateur, dans les instructions, permet de savoir si l'adresse figurant dans la zone d'arguments se réfère à un emplacement de mémoire contenant une donnée ou une instruction, ou seulement une adresse de référence à un autre emplacement.



Adressage indirect à plusieurs niveaux

Cet indicateur est un bit, le **bit d'adressage indirect**. *Il occupe une position déterminée, toujours la même, dans les instructions. Pour réaliser l'adressage indirect à un niveau n, on considère le bit correspondant à cette position dans l'emplacement adressé à l'étape n − 1 du processus. Ce dernier emplacement est analysé par l'unité de commande de la même manière que l'instruction en cours d'exécution avec la différence suivante : ne sont pris en charge dans cette analyse que le nouveau bit d'adressage indirect et la nouvelle adresse. Les zones d'arguments et les arguments autres que les deux cités (sauf exception) ainsi que la zone d'opération sont complètement ignorés. La commande d'opération reste donc toujours celle réalisée au premier niveau dans l'instruction réelle.*

L'utilisation conjointe d'un index et de l'adressage indirect est possible dans les machines pourvues d'un dispositif réalisant ce dernier. L'adressage indirect peut être prioritaire par rapport à l'indexation. Celle-ci s'applique alors à l'adresse du maillon final de la chaîne de mots reliés entre eux dans l'adressage indirect. Dans d'autres machines l'indexation a lieu avant la prise en charge de l'indication d'adressage indirect.

d) Adressage relatif technologique

L'adressage relatif *consiste en la possibilité de désigner une adresse relativement à celle de l'instruction en cours d'exécution. L'adresse de cette dernière est ajoutée à l'adresse relative pour former l'adresse effective. Les dispositifs d'adressage relatif sont beaucoup moins courants que ceux réalisant l'adressage indirect. Il est nécessaire de mettre en garde le lecteur : il ne s'agit pas ici de l'adressage relatif autorisé par tous les assembleurs. Nous étudierons ce dernier en* 2.2.3. *Les dispositifs dont nous parlons dans le présent paragraphe permettent de composer, rappelons-le, les adresses d'une manière dynamique : au moment de leur utilisation, lors de l'exécution de l'instruction où elles figurent. Un bit dans une position donnée de l'instruction permet comme pour l'adressage indirect de commander l'adressage relatif.*

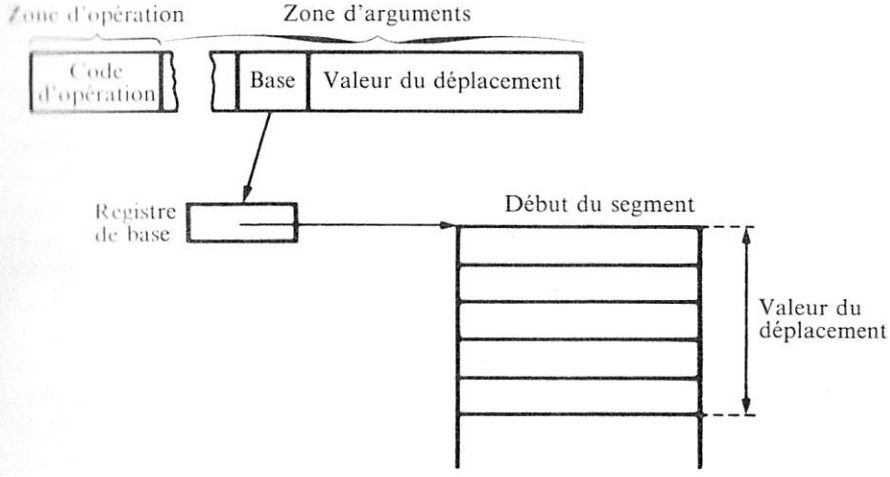
e) Adressage avec base d'adresses

L'**adressage avec base d'adresses** est une technique d'écriture des adresses d'une manière également relative. Mais relative cette fois au contenu d'un registre dit **registre de base**. L'adresse relative est désignée alors par le terme de **déplacement**. Le registre de base est du point de vue technologique tout à fait analogue à un registre d'index. Il en diffère fondamentalement du point de vue fonctionnel.

L'adressage relatif permet essentiellement d'atteindre de très grandes zones de mémoires sans qu'il soit nécessaire de réserver à cet effet une partie d'adresse, dans la zone d'arguments des instructions, d'une dimension prohibitive. Un des aspects du problème de la compatibilité des ordinateurs de tailles différentes à l'intérieur d'une même gamme se trouve ainsi résolu. Le constructeur n'est pas obligé de prévoir la taille de la partie d'adresse des zones d'arguments en fonction des plus grands modèles. Il évite ainsi un gaspillage de place en ce qui concerne les petits modèles de la série.

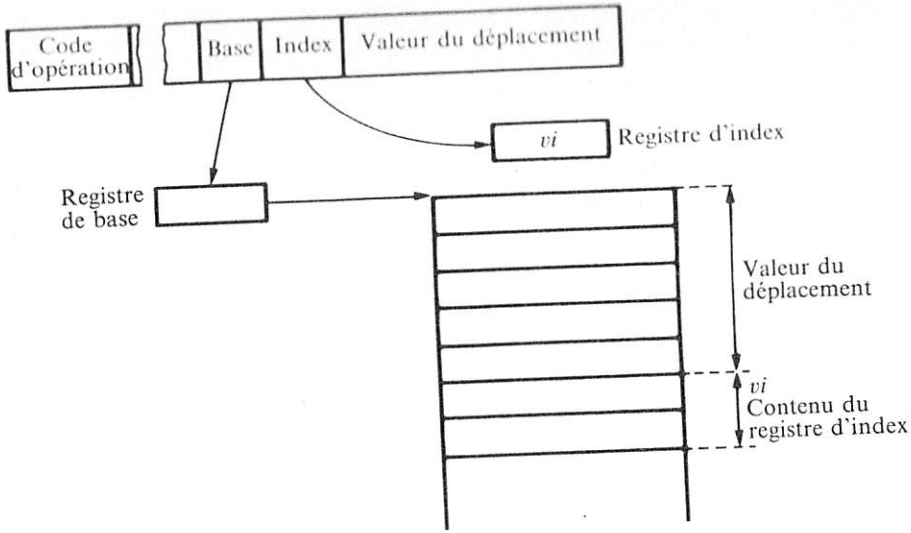
Un autre intérêt de l'adressage avec une base d'adresse est qu'il permet de réaliser d'une manière commode et performante le découpage des programmes selon une technique dite de **segmentation**. La segmentation (*) est une technique d'emploi du dispositif technologique d'adressage au moyen de bases d'adresses. Un des aspects de cette technique, celui de la **modularité** des programmes est exposé dans le chapitre consacré à l'assemblage indépendant des programmes (en 2.3). La segmentation permet de conserver sur un support externe des « segments » d'un même programme qui ne sont introduits dans la mémoire qu'au moment de leur utilisation. Le découpage en segments est contrôlé par le programmeur et reflète la structure logique du programme (ce qui n'est pas le cas de la pagination exposée ci-dessous).

L'utilisation conjuguée d'un registre de base et d'un registre d'index est possible. Elle correspond technologiquement à une double indexation ; fonctionnellement à une indexation simple à l'intérieur d'un segment (voir figures ci-dessous).



Adressage avec base sans indexation

(*) Nous donnons ces quelques précisions bien qu'elles dépassent le cadre du chapitre afin que le lecteur situe les uns par rapport aux autres les notions et les termes introduits.



Adressage avec base et indexation

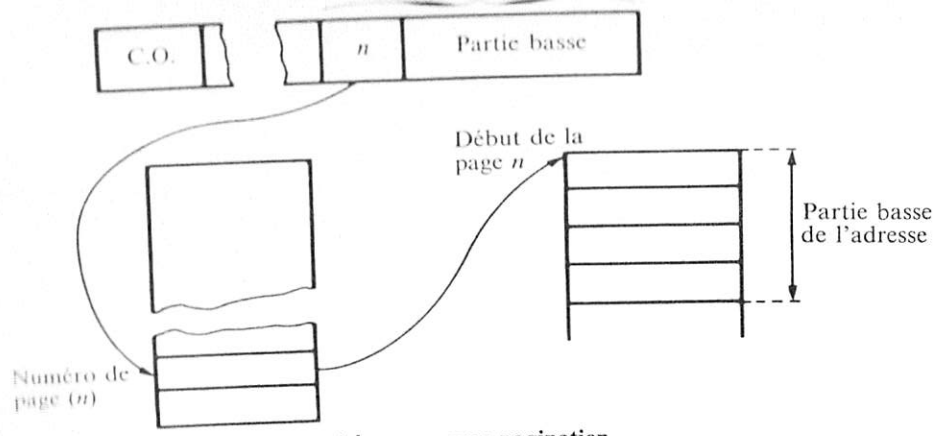
f) La pagination

Le cinquième dispositif d'adressage de la mémoire que nous étudierons est le dispositif de **pagination**. *Ce dernier terme désigne tout à la fois un mécanisme technologique et le système programmé d'organisation de la mémoire centrale qui l'utilise. La pagination, en tant que système de gestion de la mémoire, n'intéresse qu'indirectement le programmeur d'application car elle est généralement prise en charge par le système, donc réalisée par le constructeur et les programmeurs de système uniquement. Il est toutefois intéressant de connaître le fonctionnement du dispositif de pagination d'un ordinateur lorsque l'on programme en langage machine cet ordinateur. Certaines parties de la zone d'arguments y sont en effet souvent liées.*

La pagination consiste en un découpage de la mémoire en « pages » de dimension fixe. Les adresses se composent alors de deux parties. La première partie de l'adresse donne accès à un numéro de page ; la seconde à une adresse relative à l'intérieur de la page désignée. Le mécanisme d'accès au numéro de page est en général simple : la partie haute de l'adresse sert de numéro de page et désigne un élément d'une table de bases d'adresses : ce sont les adresses des débuts de page. La valeur d'adresse extraite du tableau est ajoutée à la seconde partie de l'adresse de l'instruction pour donner l'adresse physique, effective.

Le tableau des adresses des débuts de pages est souvent appelé « carte topographique de la mémoire ».

La zone d'arguments comporte généralement un bit dans une position particulière, qui rend effective ou au contraire désarme le dispositif de pagination.

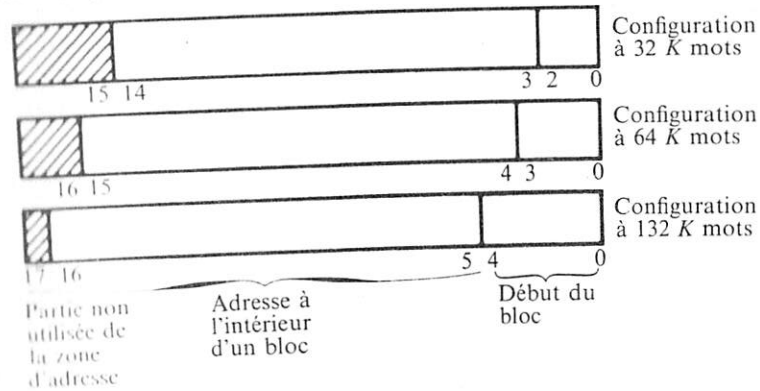


Adressage avec pagination

S'il est à 1, *le processus de substitution de l'élément du tableau, à la partie haute qui sert de pointeur, s'opère tel que décrit. Si le bit de pagination est à* 0 *les adresses ne sont pas modifiées. L'adresse physique effective est celle-là même qui figure dans l'instruction.*

g) Blocs de mémoire

Il nous faut pour finir dire un mot des blocs de mémoires existant dans certaines machines. La structure de la partie d'adresse de la zone d'arguments des instructions reflète en effet dans ces machines la partition de la mémoire en blocs. Les bits de plus haut poids ou les bits de plus faible poids de l'adresse y servent de base d'adresse. Dans le premier cas l'adresse effective est la valeur binaire contenue dans la partie d'adresse de la zone d'arguments. Dans le second cas également (Fig. ci-dessous) mais avec cette différence que deux valeurs d'adresse



Adressage discontinu dans une machine à taille de mémoire modulaire

consécutives désignent alors deux mémoires non contigues physiquement. Les mémoires correspondant à des adresses contigues ne se trouvent pas dans le même bloc. En augmentant séquentiellement la valeur de l'adresse, on passe successivement d'un bloc à l'autre avec une périodicité dépendant du nombre de blocs. Ce procédé est utilisé par certains constructeurs en vue de désynchroniser l'adressage des blocs par l'unité centrale et les unités d'échange. La partition de la mémoire en blocs permet également de définir une protection au niveau de l'adressage inter-bloc.

Certaines techniques d'adressage sont à cheval sur deux ou plusieurs de celles que nous venons de décrire. Dans certaines machines par exemple les adresses sont relatives à l'origine du bloc auquel appartient l'instruction en cours d'exécution. L'adressage inter-blocs se réalise alors au moyen de l'adressage indirect. Notre nomenclature des dispositifs d'adressage n'est pas exhaustive.

1.1.3 REPRÉSENTATION DES INFORMATIONS

1.1.3A Généralités

Les éléments binaires de la mémoire peuvent être utilisés pour représenter n'importe quelle information pour le programmeur. Nous avons vu que pour le 360 ils sont groupés en cellules appelées caractère, demi-mots, mots et double-mots. Les configurations de bits à l'intérieur de ces cellules représentent plus exactement :

— des caractères,
— des nombres,
— des codes quelconques ayant une signification donnée pour le programmeur ou pour l'unité de commande de l'ordinateur.

Nous allons dire deux mots de ces derniers et nous exposerons ensuite les principes de représentation des caractères et des nombres.

Par « codes quelconques » nous entendons une succession de 0 et de 1 auxquels l'on accorde une signification déterminée. *Nous avons déjà rencontré des exemples de représentations de ce type lors de l'étude des mécanismes d'adressage. Nous avons en effet dit qu'un bit dans une position déterminée de l'instruction indique l'utilisation, le cas échéant, de l'adressage indirect. C'est là un exemple d'indication définie par le constructeur. Dans une cellule quelconque utilisée comme objet d'une instruction (une « donnée ») le programmeur peut également attacher une signification arbitraire à chaque bit.*

Les opérations au niveau des éléments binaires sont dites **opérations logiques**. Il est essentiel de remarquer que si le programmeur peut à l'intérieur de ses cellules grouper les bits d'un point de vue logique, à sa fantaisie, il n'en demeure pas moins contraint, pour accéder aux informations, de s'y référer au niveau des cellules (caractères, demi-mots, etc.).

Nous verrons plus loin de quelle manière se réalisent les opérations logiques.

1.1.3B Représentation des caractères

La cellule élémentaire appelée « caractère » comporte par définition (voir 1.1.2A) un nombre de bits permettant de représenter les caractères utilisés par une machine donnée. Nous avons appelé « format » du caractère ce nombre de bits. Les formats les plus courants sont de six et huit bits.

Dans un système de codage à huit bits (cas du 360), on pourra par exemple représenter les lettres *a*, *b* et *c* par les trois codes

1 0 0 0 0 0 0 1
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 1

et ainsi de suite. La valeur de la configuration binaire (numération à base 2) correspond généralement au rang de la lettre considérée, dans l'alphabet, à une constante de translation près (10000000 ci-dessus). Cette règle n'est pas absolue. Dans le code EBCDIC dont est tiré l'exemple ci-dessus, l'alphabet est découpé en trois tranches laissant apparaître une périodicité dans la progression de la valeur associée aux quatre bits de plus faible poids (voir Annexe 4).

L'essentiel à connaître au sujet de la représentation interne des caractères se résume dans les quatre points suivants :

a) Tous les codes reconnus par l'ordinateur comme étant des codes de caractères peuvent tenir dans la cellule élémentaire appelée par définition « caractère » ; nous l'avons déjà dit.

b) La dimension de cette cellule, que nous avons appelée « format des caractères » peut autoriser plus de combinaisons que la machine ne reconnaît de caractères. Certaines configurations à l'intérieur d'une cellule élémentaire ne correspondent dans certaines machines à aucun caractère. Elles ne peuvent être utilisées comme des caractères dans une opération d'impression par exemple. Elles donnent lieu lorsqu'elles le sont par inadvertance, à une interruption ou à la substitution au code erroné d'un caractère arbitraire de remplacement prévu pour ces cas, par l'unité de contrôle de l'imprimante.

c) Les représentations internes des caractères obéissent à certaines règles (voir ci-dessus). La connaissance de ces règles est nécessaire pour la conversion des représentations des chiffres sous forme de caractères, en valeurs binaires. Le caractère 2 est représenté dans le système EBCDIC à huit moments ainsi :

1 1 1 1 0 0 1 0 .

Cela signifie que le programmeur qui perfore le caractère 2 sur une carte puis fait lire cette carte à partir du lecteur de cartes obtient la configura-