ERSITE DE NANCY FACULTE DES SCIENCES

LES LANGAGES A.T.F. - L.M.U.

APPLICATIONS AUX PROBLEMES LINGUISTIQUES

MISE EN OEUVRE SUR C.I.I. 10 070

THESE

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 4 septembre 1969

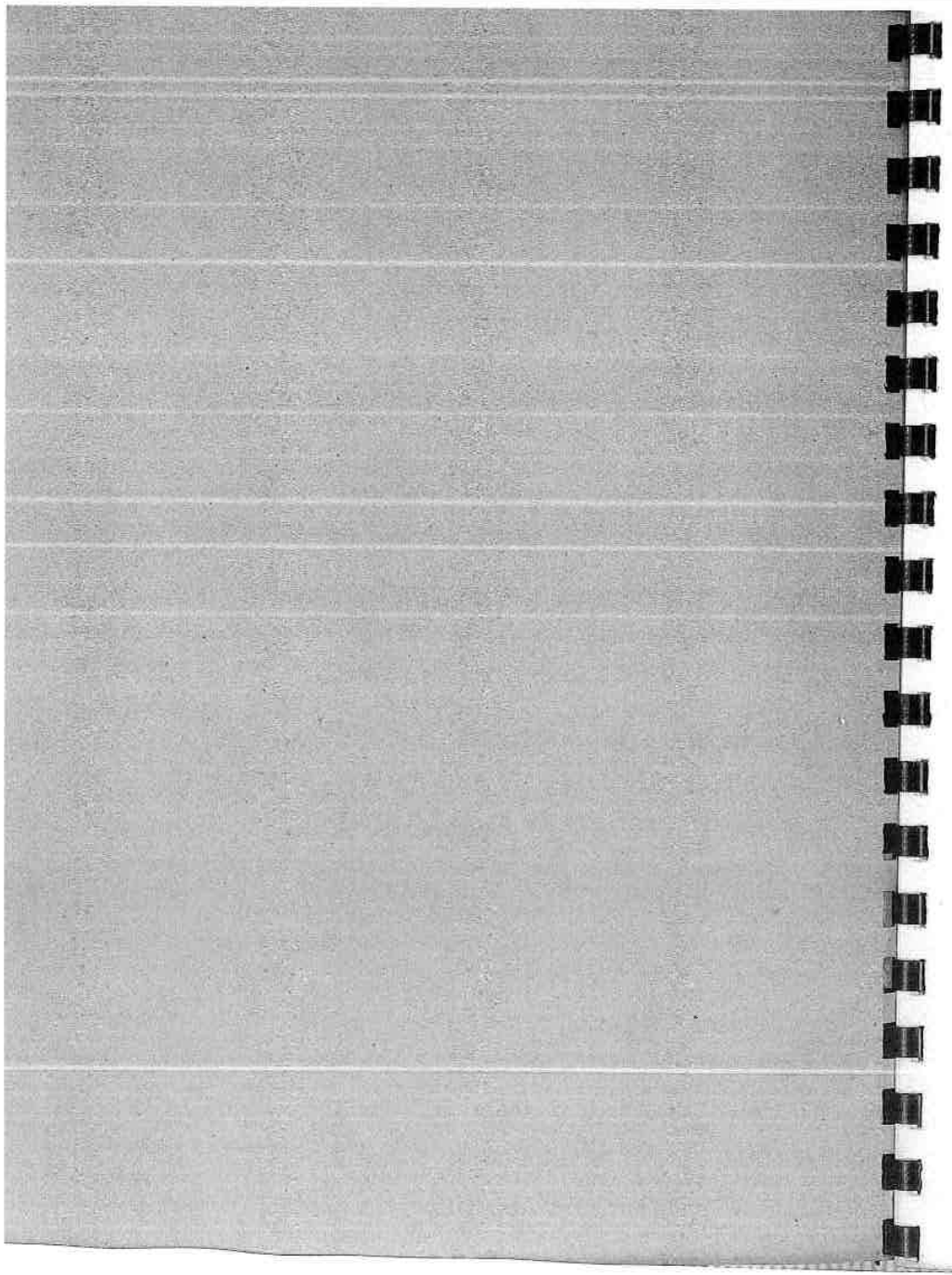
par

JEAN-MARIE DIRAND

Jury : M. J. LEGRAS Président

M. C. PAIR Examinateur

M. J.C. DERNIAME "



UNIVERSITE DE NANCY FACULTE DES SCIENCES

LES LANGAGES A.T.F. - L.M.U.

APPLICATIONS AUX PROBLEMES LINGUISTIQUES

MISE EN OEUVRE SUR C.I.I. 10 070

Par JEAN-MARIE DIRAND

Université de Nancy — Faculté des Sciences

ANNEE SCOLAIRE 1968/69

DOYEN : M. AUBRY

ASSESSEUR : M. GAY

Doyens honoraires : MM. CORNUBERT - ROUBAULT.

Professeurs honoraires : MM. RAYBAUD - LAFFITTE - LERAY - JOLY - LAPORTE - EICHBORN - CAPELLE - GODEMENT - L. SCHWARTZ - DIEUDONNE - DE MALLEMAN - LONGCHAMBON - LETORT - DODE - GAUTHIER - GOUDET - OLMER - CORNUBERT - CHAPELLE - GUERIN - WAHL.

Maîtres de Conférences honoraires : MM. LIENHART - PIERRET - Mle MATHIEU.

PROFESSEURS

MM.	ROUBAULT	Géologie		GAYET	Physiologie
	VEILLET	Biologie animale		HADNI	Physique
	BARRIOL	Chimie théorique	*	BASTICK	Chimie
	BIZETTE	Physique		DUCHAUFOUR	Pédologie
	GUILLIEN	Electronique		GARNIER	Agronomie
	LEGRAS	Mécanique rationnelle		NEEL	Chimie organique industrielle
	BOLFA	Minéralogie		BERNARD	Géologie appliquée
	NICLAUSE	Chimie	*	CHAMPIER	Physique
	FAIVRE	Physique appliquée	*	GAY	Chimie biologique
	AUBRY	Chimie minérale		STEPHAN	Zoologie
	COPPENS	Radiogéologie	*	CONDE	Zoologie
	DUVAL	Chimie	*	WERNER	Botanique
	FRUHLING	Physique		EYMARD	Calcul différentiel et intégral
	HILLY	Géologie			
	LE GOFF	Génie chimique		LEVISALLES	Chimie organique
	SUHNER	Physique expérimentale		FELDEN	Physique
	CHAPON	Chimie biologique	*	GOSSE	Mécanique physique
	HEROLD	Chimie minérale industrielle	*	DAVOINE	Physique (ENSMIN)
	SCHWARTZ B.	Exploitation minière		HORN	Physique (1° cycle)
	MALAPRADE	Chimie	*	ROCCI	Géologie
*	MANGENOT	Botanique	*	Mme LUMER	Mathématiques
				DELPUECH	Chimie physique

N...	Chimie biologique
N...	Mécanique appliquée
N...	Analyse supérieure
N...	Méthodes mathématiques de la physique
N...	Mécanique rationnelle

(*) Professeur titulaire à titre personnel

PROFESSEURS SANS CHAIRE

Mme BASTICK	Chimie P. C. Epinal	MM. FLECHON	Physique P. C.
MM. GUDEFIN	Physique	Mle HUET	Mathématiques C. B. G.
VUILLAUME	Psychophysiologie	VIGNES	Métallurgie
FRENTZ	Biologie animale	BALESDENT	Thermodynamique, Chimie appliquée (ENSIC
MARI	Chimie (ISIN)	BLAZY	Minéralogie appliquée (ENSG)
AUROUZE	Géologie	JANOT	Physique P. C. Epinal
DEVIOT	Physique du solide	CACHAN	Entomologie appliquée (ENSA)

MAITRES DE CONFERENCES

MM JACQUIN	Pédologie et chimie agricoles	MM. BAVEREZ	Chimie (ENSIC)
MAINARD	Physique M. P.	CHAMBON	Exploitation minière (Mines)
MARTIN	Chimie P. C.	HUSSON	Physique (ENSEM)
PAULMIER	Mécanique expérimentale	GERL	Physique
PROTAS	Minéralogie	WEISSLINGER	Physique
JOZEFOWICZ	Physico-chimie	ROQUES	Chimie minérale
JURAIN	Géologie C. B. G.	PERRIER	Mathématiques
RIVAIL	Chimie appliquée (CUCES)	N...	Mécanique des fluides (ISIN)
VILLERMAUX	Génie chimique	N...	Mécanique (ISIN)
METCHE	Biochimie appliquée (Brasserie)	N...	Mathématiques
PAIR	Mathématiques appliquées	N...	Mathématiques P. C.
BAUMANN	Physique 1° cycle	N...	Mathématiques C. B. G.
DURAND	Physique	N...	Physiologie animale
GRANGE	Physique (ISIN)	N...	Mathématiques M. P.
DEPAIX	Probabilités et statistiques		

CHARGES D'ENSEIGNEMENTS

MM. AMARIGLIO, COEURE, DAVRAINVILLE, GILORMINI, GIRARDEAU, HILY, MAURIN, NOVERRAZ, OVAERT, RUYER, WEBER.

Que Monsieur LEGRAS, Directeur de l'Institut Universitaire de Calcul Automatique trouve ici l'expression de ma gratitude pour l'enseignement qui m'a été dispensé à l'Institut et l'honneur qu'il a bien voulu me faire en présidant le Jury.

Ce travail a été effectué sous la direction de Monsieur PAIR à qui j'exprime mes vifs remerciements pour la formation qu'il m'a donnée et pour la bienveillante attention qu'il m'a témoignée.

J'assure de ma reconnaissance Monsieur DERNIAME qui a accepté de participer au Jury et je le remercie pour les conseils qu'il m'a prodigués durant ce travail.

Que toute l'équipe de l'I.U.C.A., en particulier Mademoiselle COLIN, trouve ici l'expression de mes remerciements pour l'aide qu'elle m'a apportée.

SOMMAIRE

INTRODUCTION

Ce travail est consacré d'une part à l'étude des langages de programmation A.T.F. (A Tout Faire) et L.M.U. (Langage Minimum Universel) et une application d'A.T.F. pour les problèmes linguistiques, et d'autre part d'une mise en oeuvre d'une implémentation de ces langages sur l'ordinateur CII 10 070.

Après une caractérisation des problèmes littéraires et des qualités d'un langage de programmation destiné à traiter ces problèmes (chapitre I), le chapitre II signifie le choix du langage A.T.F. - en particulier son extension A.T.F.-GESTION.

L'implémentation d'A.T.F. se fait par l'intermédiaire du langage de traduction L.M.U. En effet, un compilateur A.T.F.-L.M.U. écrit dans le langage L.M.U. est disponible et doit permettre une implémentation rapide sur tout ordinateur destiné à recevoir A.T.F. Le choix de l'ordinateur s'est porté sur le CII 10 070, disponible dans un proche avenir à Nancy.

Le travail pose alors la conception d'un traducteur LMU (chapitre IV) permettant d'une part l'obtention d'un compilateur ATF-LMU en langage-machine 10 070 et d'autre part la traduction des programmes produits par une compilation ATF-LMU. Le chapitre III traite de la machine LMU et de son langage.

Dans le chapitre V, est élaboré un programme chargé du contrôle de l'exécution des programmes ATF traduits, en particulier chargé de l'enchainement des programmes et d'une gestion dynamique de la mémoire disponible.

Le chapitre VI est réservé au compilateur ATF - sa constitution et son exécution.

Une annexe comporte quelques programmes ATF, traitant de problèmes littéraires.

CHAPITRE I

LES PROBLEMES LITTERAIRES ET LINGUISTIQUES

Que ce soit dans l'étude de textes, dans la documentation ou dans la traduction, un outil s'avère indispensable : l'ordinateur. Disposer d'un langage de programmation, simple et efficace, permettant une résolution automatique des problèmes posés, est nécessaire.

I. CARACTERISTIQUES GENERALES DES PROBLEMES LITTERAIRES ET LINGUISTIQUES

Un système destiné, soit à l'étude de textes, soit à la documentation ou à la traduction automatique, est caractérisé tout d'abord par la "masse" considérable d'information à traiter et ensuite par la manière de traiter cette information.

* Masse d'information à traiter

Un texte ou un document ou un dictionnaire représente le plus souvent un volume d'information non négligeable ; quand un système est destiné à traiter automatiquement des textes ou des ensembles de documents ou à effectuer des traductions à l'aide, entre autre de dictionnaires, alors l'information à traiter tient une place importante. Le problème que l'on rencontre alors est celui de la représentation de l'information.

** Définir des opérations pour le traitement de cette information devient ensuite nécessaire. Ces opérations devront permettre l'enregistrement et la recherche de l'information.

Nous allons aborder les critères d'un langage satisfaisant à ces conditions générales.

2. CRITERES D'UN "BON" LANGAGE

Reprenons le problème de la représentation de l'information et celui du traitement de cette information.

2.I. Représentation de l'information

- Le langage devra permettre la représentation d'objets simples :

  . Ainsi le mot est utilisé constamment dans les problèmes littéraires. Il est l'unité dans la phrase d'un texte, dans le résumé d'un document ou dans un dictionnaire.

  . Les entiers relatifs et les nombres réels sont utilisés dans les comptages statistiques.

  . Les booléens apparaissent dans l'écriture des programmes et dans les données.

- La notion de groupement d'objets simples de même nature apparaît alors :

  . Il est nécessaire d'avoir accès dans un mot au caractère ou à plusieurs caractères. Le langage devra donc prévoir la représentation des chaînes de caractères.

  . L'analyse statistique d'un texte fait appel aux tableaux d'entiers ou de réels.

- Une notion plus générale de représentation de l'information est celle d'article (séquence). Un article est un groupement d'objets simples de natures différentes. L'article possède alors une structure composée.

Ainsi par exemple, à un mot d'un texte correspond :

. Le mot lui-même représenté par une chaîne de caractères

. Le nombre de caractères (la longueur de la chaîne représentant le mot) qui est donc un naturel.

. Des informations, sous forme de naturels, caractérisent le mot :

  - Le rang du mot dans la phrase
  - Le rang de la phrase dans le chapitre
  - Le numéro du chapitre.

. Deux mots correspondent au titre et à l'auteur.

- La notion d'article aboutit naturellement à celle de groupement d'articles de même structure. Cette notion est celle de "file".
  Les files les plus simples sont les fichiers tel le fichier des mots d'un texte.

- La constitution de dictionnaire suppose le groupement des informations dont on peut changer l'ordre sans pour autant déplacer l'information. La structure de liste permet une telle technique.

- Les structures arborescentes devront pouvoir être utilisées. L'étude ou l'emploi de grammaires, que ce soit en documentation, en traduction ou en analyse de textes, demande l'usage d'outils tels que les arborescences et ramifications.

Nous voyons donc que les problèmes littéraires ou linguistiques nécessitent une géométrie de l'information particulièrement développée. Pour traiter cette information, le langage devra permettre de désigner efficacement l'élément des structures envisagées. La "prise" d'information dans une collection d'objets (fichier, liste, arborescence) doit pouvoir se faire suivant les cas de façon absolue (aléatoire) ou de façon relative (séquentielle) - après qu'on ait atteint l'information voisine - . Et ceci de manière aussi efficace et simple que possible, c'est à dire en utilisant un langage qui reste très proche des langages-machine.

2.2. Opérations sur les objets

Nous avons vu que, pour traiter les problèmes littéraires et linguistiques, la gamme de structures de l'information est particulièrement étendue. Il reste à préciser les opérations sur les objets ayant de telles structures.

Deux classes d'opérations sont à envisager :

. les opérations internes qui s'effectuent sur les objets placés en mémoire centrale.

. les opérations d'entrée-sortie.

a) Les opérations d'entrée-sortie

Nous avons mis en évidence le volume d'information à traiter. D'autre part, la capacité d'une mémoire centrale d'un ordinateur est limitée (coût très élevé des composants). En conséquence, il est illusoire de pouvoir disposer de toute l'information soumise à un traitement, dans la mémoire centrale - (le volume d'un dictionnaire peut déborder largement la capacité d'une mémoire interne).

En général l'information à traiter est placée sur support externe sous forme de fichiers. Afin d'avoir une souplesse et une efficacité dans le traitement, il est nécessaire de disposer d'opérations d'entrée-sortie particulièrement développées :

Un texte ou un document peut être représenté efficacement par un fichier dit "consécutif" sur support externe ; les opération d'entrée ou de sortie s'effectuant séquentiellement (enregistrement par enregistrement) sur un tel fichier, devront pouvoir être faites en simultanéïté avec les opérations internes (traitement séquentiel des fichiers).

La consultation d'un dictionnaire (placé sur disque), devra pouvoir se faire de manière aléatoire, pour être efficace.

b) Les opérations internes

- Opérations classiques.

  Les opérations arithmétiques portant sur des objets de type entier ou réel sont utilisées dans des études statistiques de texte.

  Les opérations d'affection sont effectuées sur des objets de toute nature.

  Les opérations de saut, de comparaison avec saut, permettent une architecture souple dans l'écriture des programmes.

- Opération sur les chaînes.

  Nous avons vu l'importance des chaînes dans les problèmes littéraires et linguistiques, particulièrement celles de caractères. Des opérations devront pouvoir s'effectuer :

  . la concaténation entre chaînes et expressions de sous-chaînes.

. la reconnaissance et l'enregistrement de groupes de caractères.

. la conversion (chaîne d'entiers ⟷ valeur entière).

- Opération sur les files, tableaux,

Les opérations sont celles de reconnaissance et d'enregistrement .

c) Les procédures

Les opérations que nous venons de voir appartiennent à la gamme des instructions (simples) que doit offrir le langage.

On constate dans le traitement automatique des problèmes linguistiques et plus généralement dans le traitement de l'information, l'intérêt porté aux procédures :

A l'aide d'opérations simples dont on dispose, on construit des opérations complexes (des procédures), qui portent sur des objets non identifiés dont on ne connaît que la structure.

Ainsi une procédure qui est la recherche dans un fichier de tous les articles suivant un indicatif, permet la recherche dans le texte X ou dans la phrase Y, des mots se terminant par ‚ent‚ ou bien es mots de valeur ‚que‚.

Les opérations complexes sur les listes ou les ramifications, par exemple, seraient des procédures-standard.

Au fur et à mesure de l'évolution d'un système destiné au traitement automatique des problèmes littéraires, des opérations nouvelles sont construites et viennent s'ajouter aux précédentes - tout en restant à l'entière disposition du programmeur.

CHAPITRE II

CHOIX DU LANGAGE A.T.F.

Le langage de programmation A.T.F. (A tout faire) a été défini par Monsieur L. NOLIN ([1] ch.III) ; une extension, l'A.T.F.-GESTION ([2]) a été créée pour les besoins de la gestio

L'A.T.F. fait partie des langages modernes dits "symboliques" ; il se veut universel.

Nous allons présenter les caractéristiques essentielles d'A.T.F. qui permettent de justifier le choix de ce langage pour la programmation des problèmes littéraires et linguistiques.

I - CARACTERISTIQUES D'A.T.F.

I.I. Les objets traités

A.T.F. permet de traiter des objets simples, isolés et des objets groupés.

a) Les objets simples fondamentaux sont :

- . les mots
- . les nombres naturels
- . les nombres relatifs
- . les décimaux
- . les éléments booléens ("vrai" ou "faux").

A ces objets fondamentaux s'ajoutent les objets simples

- . index de files
- . étiquettes
- . procédures.

dont nous parlerons plus loin.

b) Notion d'article (ou de séquence)

L'article est un groupement d'objets simples de nature, en général, différente.

Par exemple, les renseignements sur le mot d'un texte, sont réunis dans un article constitué de :

. le mot lui-même (un mot de I0 caractères)
. le rang du mot dans la phrase (un entier de 3 chiffres)
. le numéro du chapitre (un entier de 3 chiffres)
. le titre du texte (un mot de I5 caractères)
. le nom de l'auteur (un mot de I5 caractères)

Cet article est structuré par la partition suivante, que l'on appellera ≪struc mot≫ :

. un mot de I0 caractères puis
. un entier de 3 chiffres puis
. un auteur de 3 chiffres puis
. un mot de I5 caractères puis
. un mot de I5 caractères

Dans cet exemple, la longueur de la représentation du mot du texte est de I0 caractères même si le mot occupe moins de I0 caractères. A.T.F.-Gestion permet l'emploi des longueurs variables : cette partie du nouvel article est structuré par la sous-partition ≪variable≫ ainsi définie :

. un nombre naturel de 2 chiffres puis
. un caractère répété autant de fois que l'indique la valeur du nombre naturel précédent.

Nous voyons donc qu'un article est structuré par une partition. Une partition se décompose en ses éléments. Cette décomposition introduit éventuellement des sous-partitions qui sont décomposées de la même manière et ainsi de suite. En dernière analyse, on aboutit à des objets simples.

A.T.F.-Gestion permet :

- la répétition des éléments d'une partition
- l'emploi de partitions séquentielles
- la répétition de partition
- les partitions conditionnelles (au moins deux éléments conditionnels).

c) Les files

Des articles structurés par une même partition peuvent être groupés dans une file.
Une file est à une ou plusieurs entrées. A chaque entrée est attaché un certain nombre d'index. Chaque index relatif à une entrée d'une file peut être plac dans une position déterminée et déplacée à volonté a cours d'un traitement. Ainsi pour chaque entrée on distingue :

- le premier élément (PREMIER)
- le dernier élément (DERNIER)
- pour chaque élément, sauf le premier, celui qui le précède.
- pour chaque élément, sauf le dernier, celui qui le suit.

Ainsi l'élément (l'article) d'une file est repéré par une suite ordonnée d'index dont le premier est attaché à la première entrée, le second à la deuxième entrée, etc...
L'emploi des index correspond au calcul d'adresses dans les langages-machine ; on peut donc juger ici de l'efficacité d'une telle technique.

d) Une extension de la notion de séquence (article) permet de considérer une séquence dite homogène - les objets regroupés en une séquence sont de même nature τ - comme une file dont l'élément est de nature τ .
Cette extension permet une grande souplesse dans les opérations entre les files et les séquences homogènes (traitement des chaînes de caractères considérées soit comme files, soit comme des mots).

I.2. Désignation des objets

Les objets simples sont désignés par des constantes ou des variables qui sont des identificateurs. L'emploi des expressions qualifiées permet de désigner un élément ou une partie :

- . d'article
- . de file
- . de sous-file
- . de file partielle.

I.3. Instructions

a) Instructions d'attribution

On trouve parmi les instructions d'attribution :

- les instructions d'affectation simple,
- les instructions à opérateur unaire (+,-,ABS,-ABS, SIGNE, $\neg$ (négation)),
- les instructions à opérateur binaire (+,-,x,/, //, MOD (reste), $\wedge$, $\vee$).

On trouve des instructions permettant d'affecter à une variable entière :

- la précision d'un réel (PRE)
- l'ordre d'un réel ou d'un relatif (ORD)
- la longueur d'une file ou d'une sous-file à une entrée (LONG).

Des instructions de conversions (en particulier entier $\longleftrightarrow$ mot) A.T.F.-Gestion permet la concaténation sur les objets de nature "mot" (opérateur noté PUIS) ; il offre des instructions d'attribution dont l'opérateur arrondit les variables entières ou décimales.

b) Les instructions de saut

Les instructions de saut permettent le branchement soit à une étiquette, soit à la procédure-standard ERREUR. Elles sont soit inconditionnelles, soit conditionnelles.

c) Les instructions qui portent sur les index :

* les instructions de progression permettent d'avancer (AV) ou de reculer (AR) un index d'un nombre déterminé de "pas".

** les instructions d'attribution permettent d'affecter à un index une valeur relative à PREMIER ou DERNIER ou à un autre index (relatif à la même entrée d'une même file).

d) Les instructions d'entrée-sortie

A.T.F.-Gestion permet des entrées-sorties particulièrement évoluées, sous la forme d'instruction d'attribution.
Par exemple, soit un fichier placé sur support externe et représenté par une file f à une entrée munie d'un index α, soit un article en mémoire a, dont la structure est la même que tout article de la file f ; alors la lecture d'un article du fichier (repéré par α), vers l'article a s'écrit :

a:= f(α) ;

e) Les instructions d'appel de procédure

Un identificateur de procédure, qui est soit une constante de procédure VIDE ou ERREUR, soit une référence soit une variable de procédure, et qui est suivi de ses paramètres constitue un appel de procédure.

I.4. Les procédures

Deux sortes de procédures : les procédures complètes et les procédures incomplètes.

Structure d'une procédure complète en A.T.F. de base :

```
PROCEDURE     pc(d ; m ; r) ;
REFERENCE     r ;
ETIQUETTE     e ;
INTERNE       i ;
une suite - peut être vide - de descriptions
une suite -       "        - de définitions
DEBUT
une suite - non vide - d'instructions
FIN
```

- L'identificateur pc est le nom de la procédure ;
- d,m,r sont les suites d'identificateurs - peuvent être vides - respectives des arguments donnés, des arguments mixtes, des arguments résultats.
- La suite des identificateurs r avec ERREUR,VIDE,pc, constituent l'ensemble des références. e et i sont respectivement la suite des étiquettes, celle des identificateurs internes ou locaux.
- Les descriptions déterminent la nature des variables et leur domaine de variation.

- Les définitions attribuent des valeurs à certaines variables locales qui deviennent de ce fait des constantes dans le contexte de la procédure.

Structure d'une procédure incomplète :

```
PROCEDURE     pi(d ; m ; r) ;
REFERENCE     r
DANS   pe : ie
une suite - peut être vide - de descriptions
DEBUT
un appel de procédure (une instruction)
FIN
```

- L'identificateur pi est le nom de la procédure
- d,m,r, sont les suites des arguments
- la suite des identificateurs r avec ERREUR,VIDE, pe et pi constitue l'ensemble des références.
- ie constitue la suite des identificateurs externes à la procédure et appartenant à pe.

Une procédure écrite en langage A.T.F.-Gestion utilise des feuilles de programmation pour chaque partie (références, paramètres et internes avec leur description, instructions étiquetées ou non, définitions, description des partitions).

I.5. Traitement immédiat

Le traitement de l'information se fait par les programmes appelés TRAITEMENT IMMEDIAT en A.T.F.-Gestion ou PROGRAMME en ATF.

Un tel programme est sensiblement une procédure complète sans paramètre et sans instruction de saut.

Conclusion

Nous avons souligné dans le chapitre I, l'emploi dans le traitement automatique des problèmes littéraires et linguistiques, d'une information le plus souvent groupée.

L'A.T.F. ne permet en principe de grouper les informations que de 2 manières :

- en article s'il s'agit d'objets de natures différentes,
- en file s'il s'agit d'objets de même nature.

Toutes les structures nécessaires au traitement de l'information dans les problèmes littéraires s'y ramènent :

* une chaîne de caractères est soit une file à une entrée dont l'élément est le caractère, soit un article possédant une structure répétée.

* un tableau à n dimensions est une file à n entrées d'index $\alpha_1, \ldots, \alpha_n$.

* une liste est une file à une entrée dont tout article se compose :
  - d'une partie qui contient l'information proprement dite.
  - d'un index relatif à la file qui désigne l'élément suivant dans l'ordre de la file.

* un aiguillage est une file à une entrée dont l'élément est une étiquette.

* un fichier est une file à une entrée dont les articles sont structurés par une même partition éventuellement conditionnelle.

* un automate est une file à 2 entrées dont la première est appelée "entrées", la seconde "états". L'élément est essentiellement une procédure.

La désignation des objets traités se fait de manière naturelle . Le traitement des files ou des séquences, placées sur support externe se fait à l'aide d'opérations d'entrée-sortie particulièrement simple.

Le traitement d'objets groupés en file en mémoire centrale s'effectue de manière efficace et élégante par l'emploi d'index en nombre suffisant.

La notion de procédure, permettant de traiter des problèmes de même type, est vigoureusement dégagée. On a tout loisir de créer des outils disponibles à tout instant en bibliothèque des procédures.

D'autre part, un programme écrit en langage A.T.F. (procédure ou traitement immédiat) possède les qualités qu'on est en droit d'exiger :

- clarté et simplicité
  Un programme A.T.F. est compréhensible sans difficulté par un spécialiste autre que son auteur, et par quiconque connaissant le problème traité et des notions de programmation (la programmation en A.T.F. s'apprend sans trop de difficultés).
- la logique d'un programme A.T.F. permet toute rectifications et transformation nécessaires. Les procédures permettent une simplification dans l'écriture.
- efficacité
  A.T.F. est un langage proche d'un langage machine universel (L.M.U.). Il est donc permis de penser que l'exécution d'un programme sera aussi rapide que possible.
- l'indépendance de la machine, pour les procédures, permet une réutilisation de celles-ci sur un nouvel ordinateur et cela au moindre coût.

L'A.T.F., tel que le définit Monsieur NOLIN ([1]) s'adresse plutôt à des spécialistes de programmation. Le choix s'est porté vers son extension, l'A.T.F.-GESTION. L'A.T.F.-Gestion possède, en plus, l'opération de concaténation et surtout un éventail plus large de partitions (partitions séquentielles - partitions conditionnelles - partitions répétées) et une clarté dans l'écriture des expressions qualifiées (emploi d'identificateurs de partition). Toutefois, il est regrettable que la définition actuelle de l'A.T.F.Gestion manque de précision : - une partie de sa grammaire (G.9) ne figure pas, ainsi que l'emploi des procédures incomplètes - dans la description des partitions, une confusion est faite entre les identificateurs de mode (type) et les identificateurs de structure.

CHAPITRE III

LA MACHINE L.M.U. ET SON LANGAGE

STRUCTURE D'UN PROGRAMME L.M.U.

I. STRUCTURE DE LA MACHINE LMU

La machine LMU dispose d'une part d'une mémoire principale de N cases adressables numérotées de 0 à N-1 et d'autre part de 7 registres D,I,R0,R1,R2,R3,R4.

Le contenu d'un registre ρ est noté $C\rho$,

Le registre D permet la relativisation des variables de procédure tandis que le registre I permet d'effectuer une translation automatique des procédures placées en mémoire. Le registre R0 contient toujours la valeur zéro (CR0=0) ; les autres sont des registres auxiliaires.

La case LMU contient fondamentalement l'un des entiers naturels 0 1 ... K- avec $K=2^H$ où H désigne le nombre de bits par case. L'octet du IO 070 (H=8) correspondra à la case LMU.

2. LE LANGAGE LMU

2.I. Les expressions d'adresse

Une adresse est désignée par l'expression :

$(\alpha \pm C\rho I + C\rho 2)$

. α est une constante décimale désignant un entier naturel.

. $\rho I \in \{R0,R1,R2,R3,R4\}$

$\rho 2 \in \{R0,R1,R2,R3,R4,I,D\}$

Une telle expression désigne la case dont l'adresse est le reste modulo N de l'entier naturel représenté par $\alpha \pm C\rho 1 + C\rho 2$.

L'expression

$C\rho \qquad (\rho \in \{R0,R1,R2,R3,R4,ID\})$

est aussi une expression d'adresse.

2.2. Les expressions de longueur

La longueur d'un opérande, c'est à dire le nombre de cases qu'il occupe, est désigné par une expression de longueur :

$(\alpha$ + CRO) ou C(α + CD)

où α est une constante désignant un entier naturel.

2.3. Les expressions de désignation

On désigne un opérande d'une instruction L.M.U. par une expression de l'une des formes suivantes :

- une constante. Elle peut être de type entier (N), caractère (C), mot (M) ou booléen (B).
- ($\alpha \pm$ CPI + CP2), qui désigne l'adresse de la case ($\alpha \pm$ CPI + CP2)
- CP qui désigne le contenu du registre
- C(α + CPI + CP2), (β + CRO) C(α + CPI + CP2), C(β + CD) qui désigne le contenu de β (ou de C(β + CD)) cases dont la première est α + CPI + CP2.

2.4. Les instructions

L'instruction L.M.U. la plus générale est du type à trois opérandes :

C(α + CPI + CP2),(β + CRO) := C(α' + CP'I + CP'2),

(β' + CRO) "op" "t" C(α'' + CP"I + CP"2),(β'' + CRO) ;

où "op" désigne un opérateur et "t" désigne la lettre N,M ou B suivant le type des opérandes :

N : entier
M : mot
B : booléen

2.5. Les définitions

La codification des constantes s'effectue à l'aide des définitions.

Si Δ_θ est une constante de type θ, si β et γ sont des constantes désignant un entier naturel alors on note les définitions fondamentales de la façon suivante :

(I)
- N, β : Δ_N ; pour un entier naturel
- Z, β : Δ_Z ; pour un entier relatif
- M, β : Δ_M ; pour une constante de mot
- B, β : Δ_B ; pour une constante booléenne
- R, β,γ : Δ_R ; pour une constante réelle

Ainsi le processus de codification de la définition

$\theta, \beta : \Delta_\theta$

conduit à placer la représentation de l'objet désigné par Δ_θ dans les β cases qui suivent les précédentes codifications.
Il est évident que le nombre entier , représenté par la constante β dépend de K (nombre d'entiers naturels représentables dans une case). Aux définitions (I) sont adjointes les définitions simples :

(2)
- A : Δ_A ; pour une constante d'adresse (ceci équivaut à N,2 : Δ_A ;)
- C : Δ_M ; pour une constante de caractère (ceci équivaut à M,I : Δ_M ;)
- V : Δ_B ; pour une constante booléenne (Δ_B est VRAI ou FAUX)

ensuite les files de définitions :

(3) $S, \kappa, \nu : t_I \mid \ldots \mid t_i : e_I, \ldots, e_j$;

. κ, ν sont des constantes d'entiers naturels qui désignent respectivement le nombre, j, d'éléments d'une "file" à une entrée et ν, le nombre de termes, i, par élément.

. $t_I, \ldots, t_i$ sont des définitions partielles de chacun des termes :

R, α, β ou M, β ou B, β ou N, β

ou Z, β ou A ou C ou V.

Ils précisent le type et la longueur des termes.

. Les e_K sont des suites de i constantes séparées par des barres (|) :

$$\Delta_{\theta_I} \mid \Delta_{\theta_2} \mid \ldots \mid \Delta_{\theta_i}$$

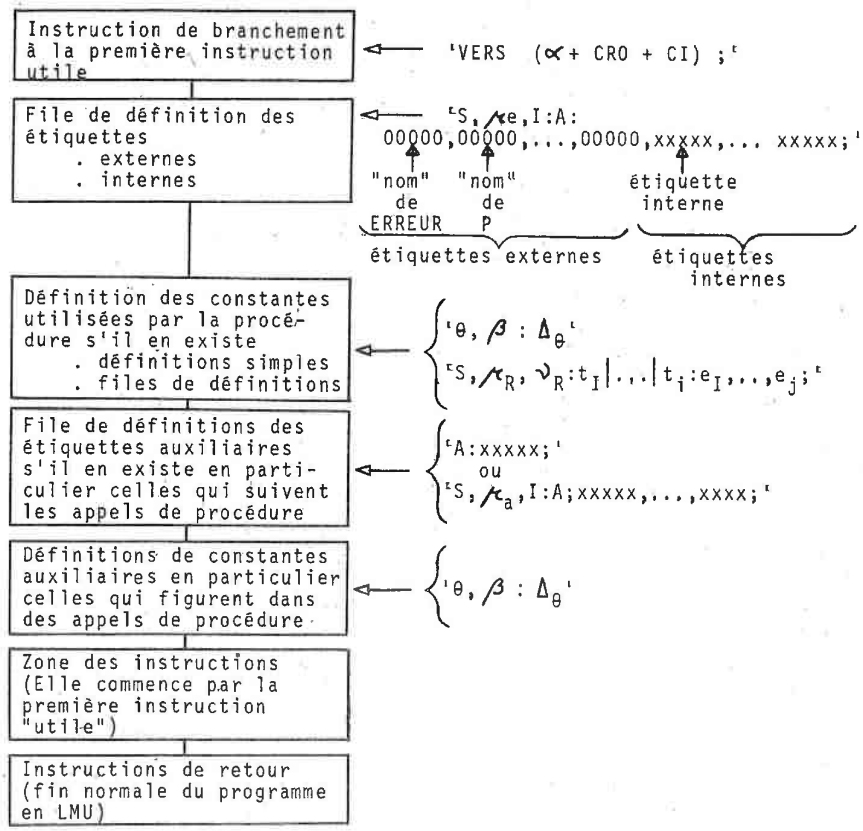
où Δ_e est du type défini par t_e.

3. STRUCTURE D'UN PROGRAMME LMU

3.I. Structure d'un programme "écrit" en L.M.U.

Un programme "écrit" en L.M.U. est produit par une compilation A.T.F.-L.M.U. . Il constitue une suite de définitions et d'instructions séparées par des points-virgules. Chaque définition ou instruction est représentée par une chaîne de caractères.

Schéma général d'un programme P écrit en L.M.U.



3.2. Structure d'un programme P, codé dans la machine LMU

Les emplacements occupés dans la mémoire principale, au moment de son exécution, par P et par les données qu'il traite sont disjoints. On les désigne respectivement sous les noms de "file des instructions" et de "file des données" relatives à P ; ce sont les contenus de cases consécutives commençant aux adresses CI et CD (respectivement contenus des registres I et D).

Les instructions LMU sont codées de façon telle que chacune d'elles occupe exactement 6 cases.

3.2.I. Structure de la file des instructions

Adresses	Contenus
0+CI	'VERS(∝+CR0+CI);'
6+CI	"nom" de la procédure ERREUR
8+CI	"nom" de P
I0+CI] .....]	"nom" des autres références utilisées par P (étiquettes externes)
2m+CI] .....]	Valeurs des étiquettes
2n+CI] .....]	Constantes définies dans P
p+CI] .....]	Valeurs des étiquettes auxiliaires
q+CI] .....]	Constantes auxiliaires
∝+CI] .....]	Instructions
S+CI] .....]	"Instructions de retour"
t+CI	première case libre.

Les entiers naturels 0,6,8,I0,2m,2n...,t sont des adresses relatives dans la file des instructions.

3.2.2. Structure de la file des données

Adresses	Contenus
0+CD] ....]	Informations nécessaires aux instructions de retour et d'appel de procédures
8+CD] ....]	"caractéristiques" des arguments
a+CD] ....]	"caractéristiques" des internes de longueur variable et valeur des internes de longueur fixe
b+CD] ....]	Constantes définies
c+CD] ....]	Constantes auxiliaires
e+CD] ....]	Valeurs des internes de longueur variable
f+CD	Première case libre

Les entiers naturels 0,8,a,b,...,f sont des adresses relatives dans la file des données.

4. EXECUTION D'UN PROGRAMME EN MACHINE LMU

Soit $\mathcal{P}_0$ une procédure qui contient un appel θ à la procédure $\mathcal{P}_I$. En supposant que les traductions de $\mathcal{P}_0$, $\mathcal{P}_I$ sont en mémoire principale et que celle-ci est suffisante pour contenir toutes les files de données nécessaires au déroulement de $\mathcal{P}_0$ alors l'opération de substitution de $\mathcal{P}_I$ à θ dans $\mathcal{P}_0$ est réalisée par l'enchainement des programmes et par la substitution des opérandes, qui figurent dans l'appel d'une procédure, aux arguments de celle-ci.

4.I. Enchainement automatique des programmes

Les informations nécessaires aux instructions de retour et aux appels de procédure se trouvent dans les huit premières cases de la file des données d'une procédure $\mathcal{P}_j$, appelée par une procédure $\mathcal{P}_i$:

Adresse	Longueur	Contenus
0+CD	2	CD pour $\mathcal{P}_i$ moins CD pour $\mathcal{P}_j$ Valeur négative s'exprimant en cases (utilisé lors du retour)
2+CD	2	Son contenu ajouté à celui du registre D donne l'adresse de la première case libre après la file des données de $\mathcal{P}_j$ (utilisé lors d'un appel)
4+CD	2	Adresse relative de retour (en cases) dans la procédure $\mathcal{P}_i$ (adresse de la première instruction de $\mathcal{P}_i$ qui suit l'appel de $\mathcal{P}_j$) (utilisé lors du retour)
6+CD	2	"nom" de la procédure appelant $\mathcal{P}_i$ (utilisé lors du retour)

* Appel de procédure. $\mathcal{P}_j$ dans $\mathcal{P}_i$ se présente de la manière suivante

Instructions	Effets des instructions
CR2:=NC(2+CRO+CD),(2+CRO);	L'instruction charge l'adresse de la première case libre dans le registre R2
CRI:=N(0-CR2+CRO);	charge RI de l'opposé de R2
Suite d'instructions qui place les caractéristiques des opérandes substituées aux arguments de $\mathcal{P}_j$ à partir de la huitième case qui suit la file des données de $\mathcal{P}_i$;	
C(0+CR2+CD),(2+CRO):=NCRI;	range la différence de CD pour $\mathcal{P}_i$ et CD pour $\mathcal{P}_j$ dans les deux premières cases de la file des données de $\mathcal{P}_j$ (ces deux cases sont aussi celles qui suivent la file des données de $\mathcal{P}_i$).
C(4+CR2+CD),(2+CRO):=NC(∝+CRO+CI),(2+CRO);	La valeur de l'étiquette de l'instruction suivant l'appel de $\mathcal{P}_j$ dans $\mathcal{P}_i$ est chargée dans les cases 5 et 6 de la file des données de $\mathcal{P}_j$
C(6+CR2+CD),(2+CRO):=NC(8+CRO+CI),(2+CRO);	Le "nom" de $\mathcal{P}_i$ (procédure appelante) est placé dans les cases 7 et 8 de la file des données de $\mathcal{P}_j$
CRI:=C(β+CRO+CI),(2+CRO);	Le "nom" de $\mathcal{P}_j$ (procédure appelée) est placé dans le registre RI
CD:=N(0+CR2+CD);	Le contenu du registre D prend la valeur de l'adresse de la première case qui suit la file des données de $\mathcal{P}_i$ - case qui est aussi la première de la file des données de $\mathcal{P}_j$
CI:=NC(0+CRI+CRO),(2+CRO)VERS(0+CRO+CI)	Le contenu du registre I prend la valeur de l'adresse d'implantation de la procédure $\mathcal{P}_j$. Puis il y a branchement à cette adresse.

** Retour à une procédure $\mathcal{P}_i$ dans $\mathcal{P}_j$.

Les "instructions de retour" se composent :

Instructions	Effets des instructions
CRI:=NC(O+CRO+CD),(2+CRO);	charge le registre RI de la différence des CD de $\mathcal{P}_i$ et $\mathcal{P}_j$
CR2:=NC(4+CRO+CD),(2+CRO);	charge le registre R2 de l'adresse relative de retour dans la procédure $\mathcal{P}_i$
CR3:=NC(6+CRO+CD);	charge le registre R3 du "nom" de la procédure appelante $\mathcal{P}_i$
CD:=N(O+CRI+CD);	Le registre D prend la valeur de l'adresse de la première case de la file des données de $\mathcal{P}_i$. (CD:=CD pour $\mathcal{P}_i$)
CI:=NC(O+CR3+CRO),(2+CRO)VERS(O+CR2+CI);	Le contenu du registre I prend la valeur de l'adresse d'implantation de la procédure $\mathcal{P}_j$. Puis il y a branchement à l'instrustion de $\mathcal{P}_i$ qui suit l'appel de $\mathcal{P}_j$

4.2. Substitution des opérandes aux arguments

La substitution des opérandes qui figurent dans un appel de $\mathcal{P}_j$ aux arguments de cette procédure est assurée par la structure des caractéristiques des opérandes.

Dans la procédure appelante $\mathcal{P}_i$ figure une suite d'instructions qui place les caractéristiques des opérandes substitués aux arguments de $\mathcal{P}_j$ à partir de la huitième case qui suit la file des données de $\mathcal{P}_i$.

Exemple :

Un booléen a comme caractéristiques :

Adresse relative à CD	Longueur	Contenu
X	I	'B'
X+I	2	Adresse de la case qui contient l'opérande.

Supposons que la procédure $\mathcal{P}_i$ ait à transmettre lors d'un appel à $\mathcal{P}_j$, un booléen. L'opérande de nature booléenne occupe la case d'adresse relative α, dans la file des données de $\mathcal{P}_i$.

Supposons que le paramètre correspondant de $\mathcal{P}_j$ soit le premier.

La substitution au premier paramètre de $\mathcal{P}_j$ place les caractéristiques de l'opérande booléen à partir de la huitième case qui suit la file des données de $\mathcal{P}_i$.

La séquence d'instructions dans $\mathcal{P}_i$ opérant la substitution est :

```
C(8+CR2+CD),(I+CRO):=M'B';
C(9+CR2+CD),(2+CRO):=N(α+CRO+CRI);
```

Le registre R2 (respectivement le registre RI) a été chargé préalablement de l'adresse relative de la première case suivant la file des données de $\mathcal{P}_i$, respectivement pour RI de l'opposé de cette valeur.

Il est à remarquer que α+CRO+CRI correspond bien à l'adresse relative de l'opérande, pour le CD de $\mathcal{P}_i$.

CHAPITRE IV

UN TRADUCTEUR L.M.U.

Introduction

Le rôle du traducteur L,M,U. est de traduire un programme source écrit dans le langage L.M.U. en un programme objet, sémantiquement équivalent, sous forme binaire, directement assimilable par l'ordinateur CII-IO 070.

Le programme source constitue un fichier-données, DLMU, du traducteur. L'article est soit une définition, soit une instruction LMU, constituant une chaîne alphanumérique, sans blanc non significatif, qui se termine par un point-virgule.

Le programme cible (objet) est généré dans la zone de travail d'adresse OBJET. Il est assemblé à partir de l'adresse zéro, car toute adresse de la partie programme est indexée par le registre-relativiseur I. Lors d'une exécution, ce registre contient l'adresse d'implantation du programme . Le programme généré occupe un nombre entier de pages (5I2 mots) ; ceci dans un but de simplification de la gestion de la mémoire disponible lors d'une exécution. En conséquence, la zone OBJET occupe un nombre entier de pages. Cette zone est libérée de son contenu soit après une traduction, soit lorsqu'elle est entièrement remplie. Pour cela une écriture de son contenu s'effectue sur disque dans le fichier résultat constituant la bibliothèque des traductions de procédures. (cf. ch,V - bibliothèque des procédures).

Le traducteur s'applique tout d'abord sur la première instruction d'un programme LMU qui est un saut de la zone des définitions vers la première instruction "utile" de la zone des instructions. (cf. ch.III - Structure d'un programme LMU).

Le traducteur s'applique ensuite sur la zone des définitions. Les adresses relatives portant sur cette zone sont entièrement respectées ce qui entraine une codification des définitions identique à celle qui serait faite dans la machine LMU théorique.

La zone des instructions, proprement dite, est ensuite traitée par le traducteur. Les adresses relatives portant sur cette zone ne sont pas respectées puisque d'une instruction LMU (codée sur 6 cases en machine théorique) va correspondre une séquence d'instructions-machine IO 070 en nombre variable ; ces adresses relatives apparaissant dans le programme LMU "écrit" ne correspondant en fait qu'à des étiquettes symboliques et sont alors traitées comme telles.

ORGANISATION GENERALE

L'architecture fixe d'un programme LMU permet d'avoir pour la zone instruction du traducteur une structure modulaire et linéaire :

- . module de traitement de la première instruction ("non utile")
- . module de traitement de la file des étiquettes
- . module de traitement de la zone "des définitions"
- . module de traitement de la zone des instructions.

Pour chaque module les différentes phases sont semblables :

1ère phase :

Garnissage de la zone de travail ANALYSE par une instruction ou une définition, obtenue à partir du fichier séquentiel DLMU. Cette opération s'effectue à l'aide du sous-programme GARNIR.

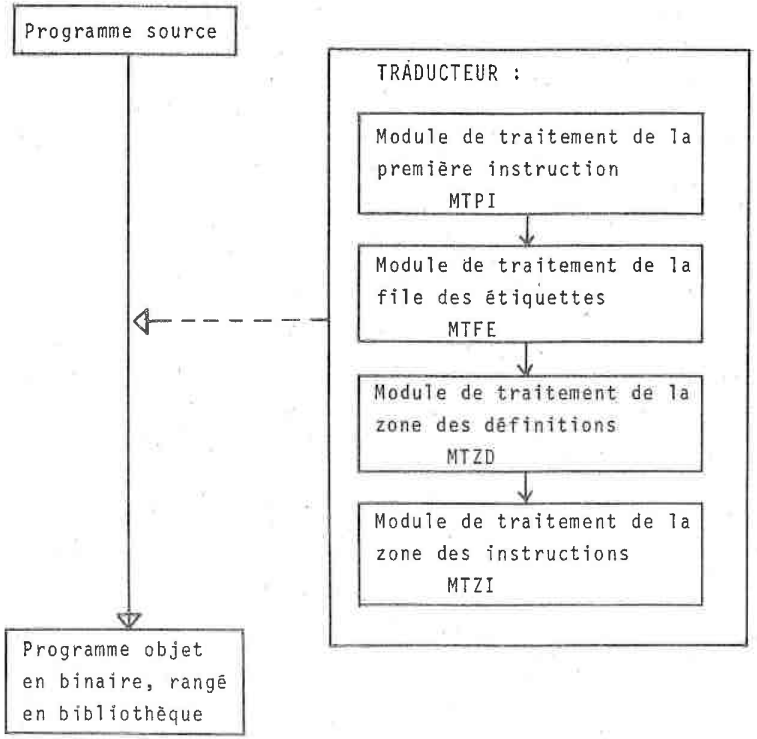
2ème phase :

Reconnaissance et analyse de la zone ANALYSE munie d'un index S permettant une recherche systématique caractère par caractère. Garnissage de tables s'il y a lieu.

3ème phase :

Exploitation des résultats et génération dans la zone OBJET munie de l'index Ø permettant le remplissage et un test de débordement de cette zone.

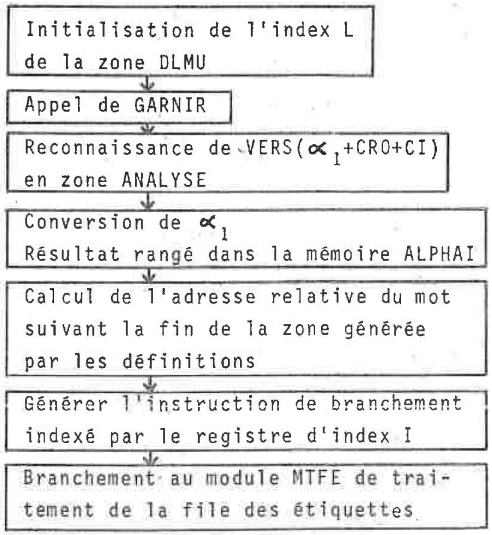
ORGANIGRAMME GENERAL DU TRADUCTEUR



MODULE DE TRAITEMENT DE PREMIERE INSTRUCTION MTPI

La première instruction LMU du type VERS(α_1+CRO+CI) permettant le saut de la zone des définitions, est un branchement à la première instruction LMU "utile". La valeur de α_1 représentée par une chaîne de 5 chiffres décimaux, indexée par I, désigne dans la machine LMU la première case libre suivant la zone des définitions correspondant à l'adresse de la première instruction "utile". Comme la machine IO 070 n'accepte que des instructions commençant en début de mot-mémoire, l'instruction à générer est alors une instruction de branchement à l'adresse du mot suivant la fin de la zone générée par la définition. Dans la machine LMU où une instruction occupant 6 cases, il s'ensuit que la codification suivante (file des étiquettes) s'effectue à partir de l'adresse relative 6 (en octets).

ORGANIGRAMME GENERAL



Constantes et mémoires de travail propres à MTPI

* Le mot d'adresse BRANCH contient une constante correspondant au profil binaire de l'instruction IO 070 `BCR,0 ,RI` qui est un branchement direct, inconditionnel indexé par RI (un registre d'index de la IO 070 sera choisi pour RI) sans opérande.

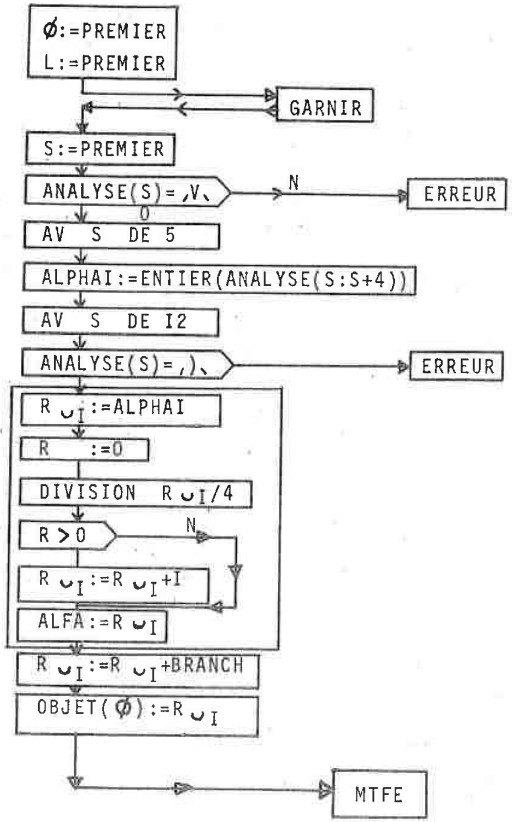
* La mémoire de travail d'adresse ALPHAI contient la valeur numérique binaire de la chaîne α_I convertie. Cette mémoire sera utilisée ensuite pour tester la fin de la zone des définitions.

* La mémoire de travail d'adresse ALPHA contiendra l'adresse relative du premier mot mémoire suivant la zone générée des définitions. Cette mémoire sera utilisée pour donner au pointeur Ø l'adresse relative en mot de la première instruction IO 070 "utile".

* R est un registre pair de la machine IO 070

  $R_{\cup I}$ le registre suivant.

ORGANIGRAMME



MODULE DE TRAITEMENT DE LA FILE DES ETIQUETTES - MTFE

La structure de la file des étiquettes est celle des files de définitions en LMU.

La file contient deux suites séparées par une virgule.

I. La suite des ne étiquettes externes séparées par des virgules et apparaissant sous la forme de mot '00000' ; à la traduction, il ne s'agit que de laisser la place nécessaire à leur codification. Chaque valeur d'étiquette doit être codifiée sur 2 cases, c'est à dire 2 octets. La codification de la première suite commençant à l'adresse relative 6, celle de la deuxième suite commence donc à l'adresse 6+2xne, ce n'est que lors de la mise à jour de la table CATALOGUE des procédures en bibliothèque que cette zone sera remplie par les "noms" des procédures correspondant aux étiquettes externes.

2. La suite des étiquettes internes, celles du programme A.T.F., séparées par des virgules. Chaque étiquette est représentée en LMU par une chaîne de 5 chiffres décimaux dont la valeur numérique représente une adresse relative en cases dans le programme codé dans la machine LMU théorique. Rappelons qu'une instruction LMU codée occupe 6 cases ; une instruction LMU une fois traduite, va être représentée par une séquence d'instructions en nombre variable. Il s'en suit que les étiquettes sont symboliques.

Le rôle du module MTFE est, d'une part, de laisser la place nécessaire à la codification des deux suites, d'autre part, de garnir la table des étiquettes internes appelée ETIQUETTE. Pour chaque étiquette symbolique rencontrée (de la deuxième suite), l'article suivant dans la table est garni. L'article est constitué :

. d'un demi-mot, ETI dans lequel est rangé la valeur numérique de la chaîne décimale représentant l'étiquette.

. d'un demi-mot AD dans lequel est rangé l'adresse relative du premier octet de la codification de l'étiquette dans le programme objet.

Cette table sera gérée lors du traitement de la partie instructions :

Pour chaque article, le demi-mot ETI sera remplacé l'adresse relative "vraie" de l'étiquette dans le programme objet. Après traduction de la zone instruction, il y aura codification dans le programme objet de ces adresses ; l'adresse de début de codification (pour chaque étiquette) est fournie par la partie AD.

ORDINOGRAMME GENERAL DE MTFE

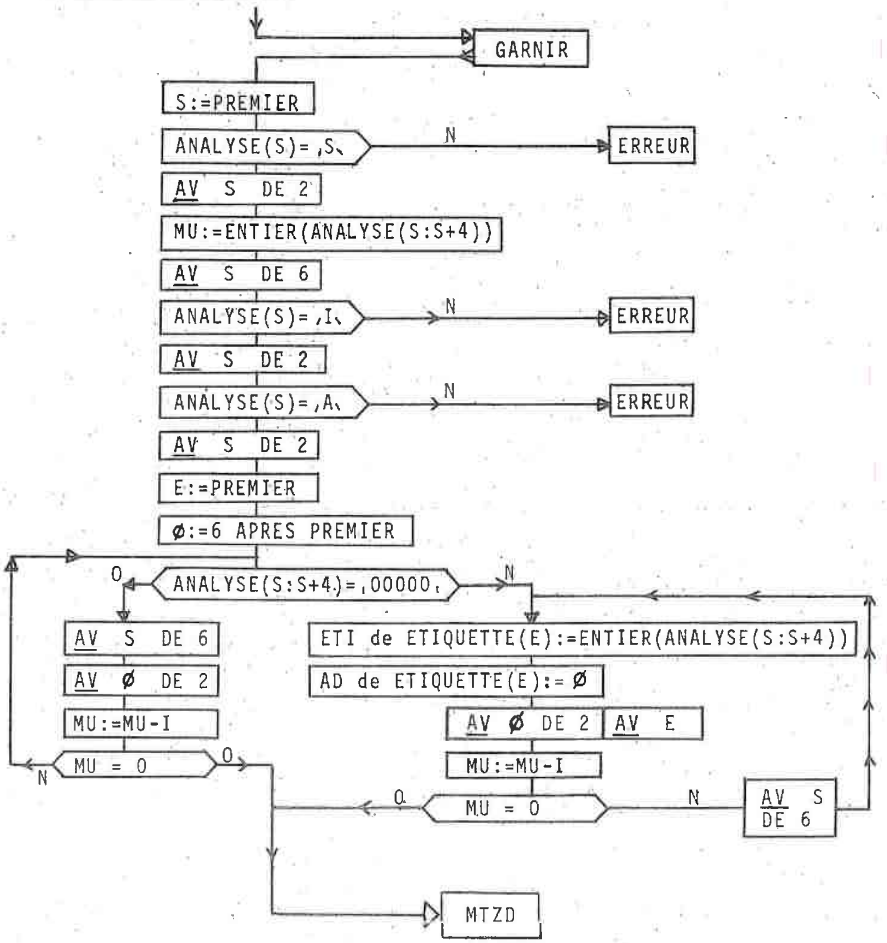
Mémoires de travail et pointeurs

* MU désigne un mot-mémoire contenant le nombre d'articles d'une file à traiter en particulier ici la file des étiquettes. Après chaque article traité, MU est décrémenté de I.

* E est un pointeur de la table ETIQUETTE.

* Ø désigne une adresse relative en octets dans le programme généré, c'est un paramètre mixte du module.

ORGANIGRAMME DE MTFE



MODULE DE TRAITEMENT DE LA ZONE DES DEFINITIONS | MTZD

L'introduction dans un programme LMU de constantes, qu'elles soient de mode réel (R), entier relatif (Z) ou naturel (N), booléen (B), caractère (C), mot (M), booléen de longueur I (V) ou adresse (A), se fait par l'intermédiaire de la zone des définitions.

Cette zone est constituée par une suite de définitions séparées par des points-virgules ; chacune peut être simple dans le cas d'une seule constante ou "composée" , dans le cas d'une "file" de définitions regroupant par articles de même structure une ou plusieurs constantes LMU ; chaque constante apparaît comme une chaîne de caractères constituant un terme d'un article d'une file (cf. ch.III - Les définitions).

Une définition simple est un cas particulier de file de définitions où le nombre d'articles et le nombre de termes par article valent implicitement tous les deux I ; seul le cas d'une définition composée sera donc envisagé.

Le module MTZD effectue la codification des constantes en opérant sur une définition après l'autre. La codification d'une constante LMU (un terme d'un article) s'effectue à partir de la première case libre qui suit les précédentes codifications en fonction de la longueur et du mode de la constante. Pour chaque mode possible, MTZD dispose d'un sous-programme codifieur. Chaque codifieur conduit à générer dans le programme objet la constante LMU, de mode correspondant, sur laquelle il est appliqué.

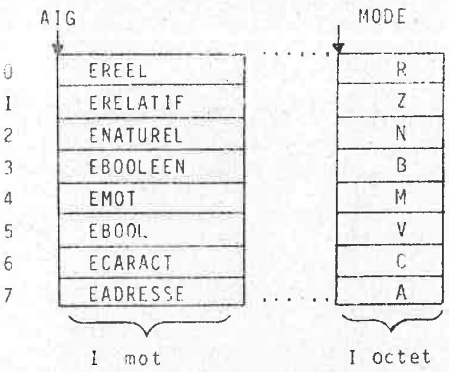
Dans une file de définitions, les articles ont tous la même structure, c'est à dire, que les $i^{èmes}$ termes des articles sont de même mode et de même longueur ; donc le même codifieur leur est appliqué.

L'analyse des définitions "partielles" d'une file renseigne sur les modes et les longueurs des termes de l'article de la file; ce qui permet pour tout article, de déterminer quelle suite d'appels de codifieurs faut-il exécuter.

res de travail et tables

- mémoires d'adresse MU et NU serviront à placer pour chaque définition, respectivement son nombre d'articles et son nombre de termes par article.

* MODE et AIG sont les adresses origines de deux tables en parall Le $i^{ème}$ enregistrement de la table MODE (respectivement le $i^{ème}$ de la table AIG) contient le $i^{ème}$ caractère de la chaîne des modes possibles 'R Z N B M V C A' (respectivement l'adresse relative du codifieur correspondant au mode).

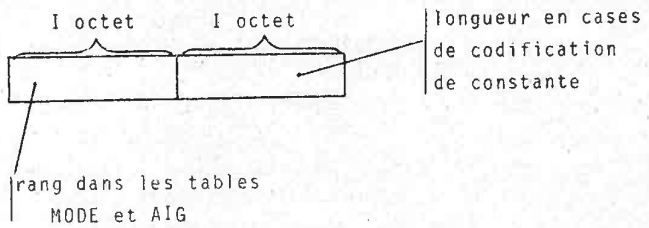


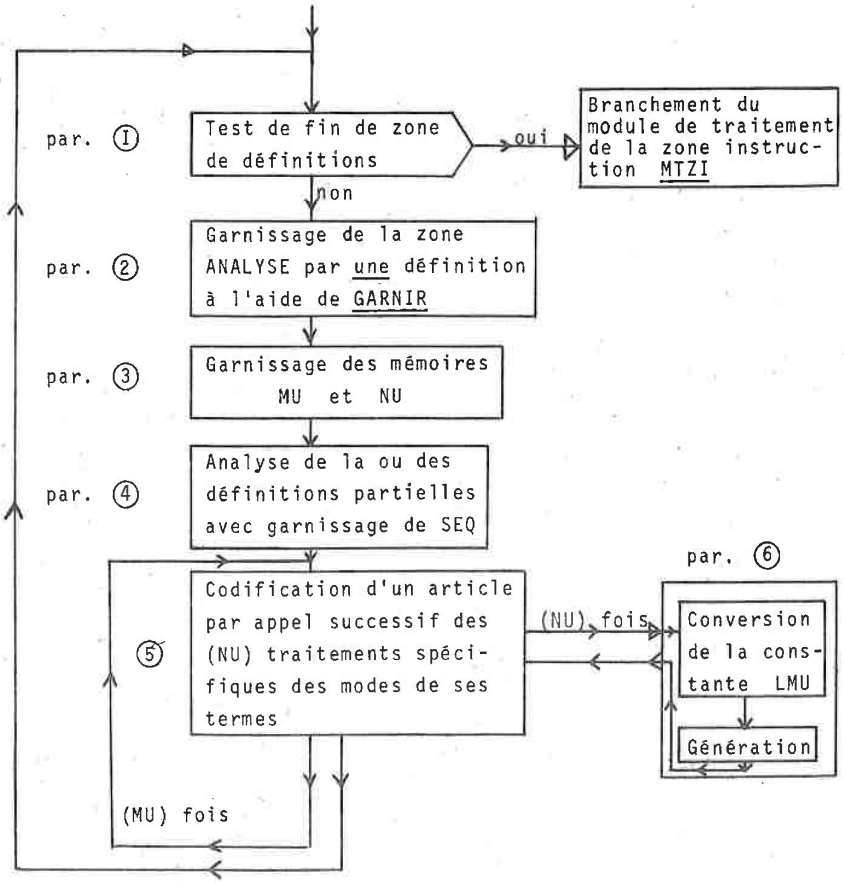
* SEQ désigne l'adresse origine de la table construite à l'analys des (NU) définitions partielles. Le $j^{ème}$ enregistrement placé su un demi-mot, comporte dans le premier octet un rang dans la table AIG (ou valeur d'index de cette table). Ce rang est le rang dans la table MODE, du caractère constituant la première partie de la $j^{ème}$ définition partielle. Dans le deuxième octet se trouve la longueur en case LMU (octet) de codification d'une constante. Cette longueur obtenue par conversion de la chaîne de caractères constituant la deuxième partie de la $j^{ème}$ définition partielle.

Ainsi la table, SEQ, construite un seule fois par analyse d'une file de définitions, va permettre d'effectuer, pour chaque article de la file, la suite des (NU) appels des codifieurs.

Le $i^{ème}$ terme d'un article de la file va être codifié par l'appel du sous-programme codifieur dont l'adresse se trouve dans AIG. Le rang dans la table AIG, de cette adresse, est celui situé dans le $i^{ème}$ enregistrement de SEQ. La codification se fait sur la longueur en case, contenue aussi dans le $i^{ème}$ enregistrement de SEQ.

Structure de l'enregistrement de la table SEQ




par. ①
Test de fin de zone de définitions
oui
Branchement du module de traitement de la zone instruction MTZI
non
par. ②
Garnissage de la zone ANALYSE par une définition à l'aide de GARNIR
par. ③
Garnissage des mémoires MU et NU
par. ④
Analyse de la ou des définitions partielles avec garnissage de SEQ
par. ⑥
⑤
Codification d'un article par appel successif des (NU) traitements spécifiques des modes de ses termes
(NU) fois
Conversion de la constante LMU
Génération
(MU) fois

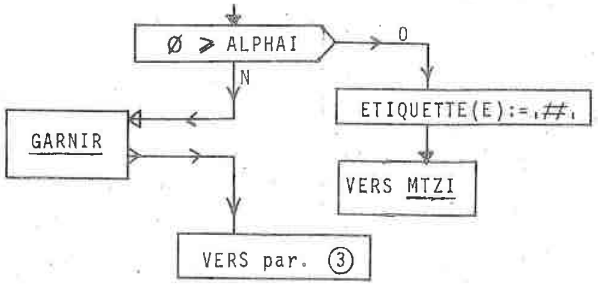
par. ①

Le test sur la fin de la zone des définitions se fait avec les contenus des mémoires ALPHAI et Ø (cf.MTPI).
ALPHAI contient l'adresse relative de la première case libre suivant la zone codifiée des définitions.
Ø contient l'adresse courante de la première case libre.
Tant que Ø < ALPHAI, il reste des définitions. Dans le cas contraire, un marqueur (,#,) est inséré dans la table étiquette et un branchement au module de traitement de la zone instruction (MTZI) est effectué.

par. ②

Le garnissage de la zone ANALYSE commune à tous les modules, se fait par l'appel du sous-programme GARNIR.

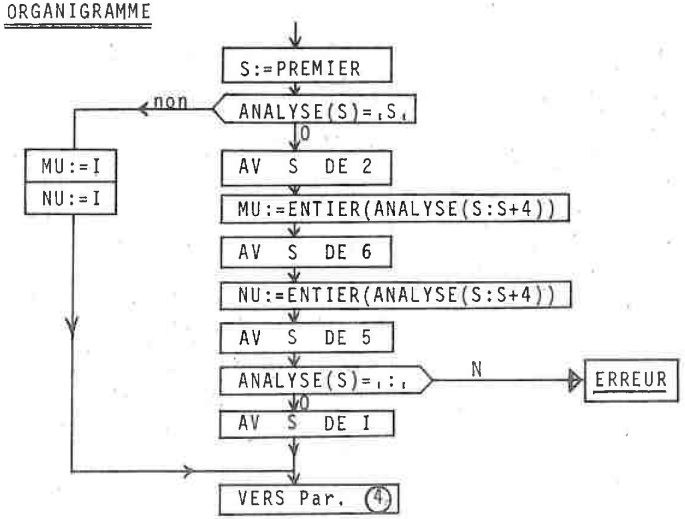
ORGANIGRAMME DE par. ① et par. ②



par. ③

Le test sur le premier caractère de la zone ANALYSE permet de savoir si la définition est simple (respectivement composée) lorsqu'il est différent de ,S, (respectivement égal à ,S,). Dans le premier cas, les mémoires MU et NU sont initialisées à I, dans l'autre le nombre d'articles (respectivement le nombre de termes par article) est obtenu par conversion de la chaîne de cinq chiffres décimaux ⌐ (respectivement ⌙).
(cf. description des définitions LMU - Ch.III).

ORGANIGRAMME



Remarque :

ENTIER(ANALYSE(S:S+4)) est le résultat de la conversion de la chaîne de chiffres décimaux commençant à l'adresse ANALYSE indexé par S (S contient une valeur en octet) et se terminant en S+4.

par. ④

Ce paragraphe, par l'analyse de la zone des définitions "partielles", garnit la table SEQ. La zone des définitions "partielles" est constituée de (NU) définitions partielles séparées par le caractère ,|,. A chacune correspond un enregistrement dans la table SEQ. La $i^{ème}$ définition partielle rend compte du mode et de la longueur en cases LMU, en fonction desquels le $i^{ème}$ terme (constituant une constante LMU) de chaque article de la file, devra être codifié dans le programme objet. Le mode s'exprime sous la forme d'un caractère appartenant à la chaîne 'R Z N B M V C A'. Le rang du caractère dans cette chaîne est chargé dans le $i^{ème}$ enregistrement de SEQ.

Lorsque le caractère identifiant le mode appartient à la chaîne 'Z N B M', la longueur de la codification s'exprime sous forme d'une chaîne de 5 caractères décimaux. Lorsque le caractère identifiant le mode est ,R, une première chaîne de 5 caractères décimaux représente la longueur en cases suivant laquelle l'exposant d'une constante réelle doit être codifié ; une deuxième chaîne de 5 caractères décimaux représente la longueur en cases suivant laquelle la mantisse d'une constante réelle doit être codifiée.

Lorsque le caractère identifiant le mode est ,V, ou ,C, ou ,A, la longueur de la codification est implicite et vaut respectivement I,I,2. Une table SUITE, jouant le rôle d'un aiguillage, permet un branchement au traitement qui garnit la partie longueur du $i^{ème}$ enregistrement de SEQ.

SUITE

0	SI
I	S2
2	S2
3	S2
4	S2
5	S3
6	S3
7	S3

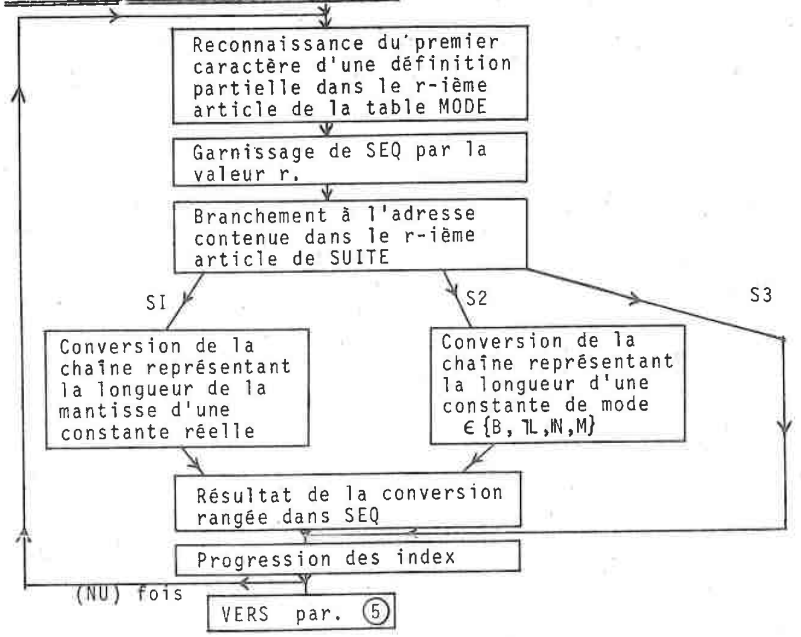
I mot

SI est l'étiquette du traitement des longueurs de codification d'une constante réelle.

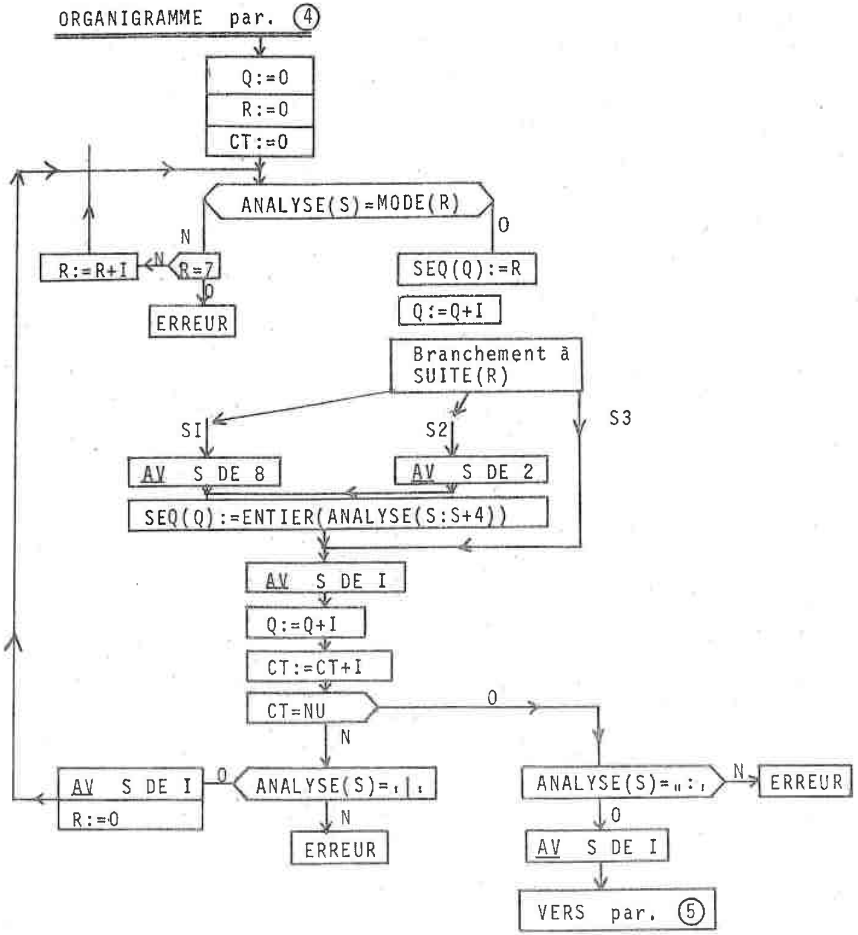
S2 est l'étiquette du traitement des longueurs de codification d'une constante de mode relatif ou naturel ou booléen ou mot.

S3 est l'étiquette des traitements des longueurs implicites.

ORGANIGRAMME GENERAL DE par. ④



ORGANIGRAMME par. ④

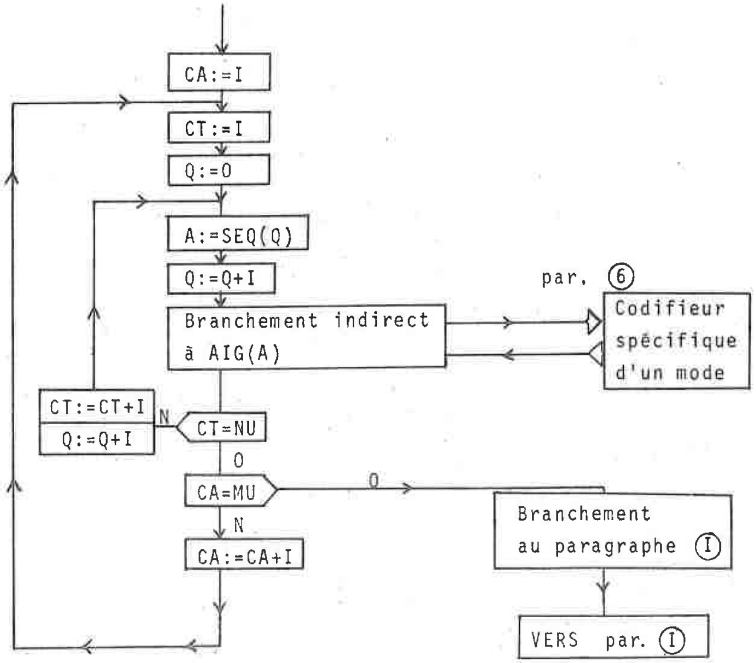


Q,R,CT sont des mémoires de travail.

par. ⑤

Ce paragraphe effectue par article, une séquence d'appels des sous-programmes codifieurs permettant la codification des (NU) constantes constituant l'article ; ceci répété (MU) fois pour les (MU) articles de la file de définitions.

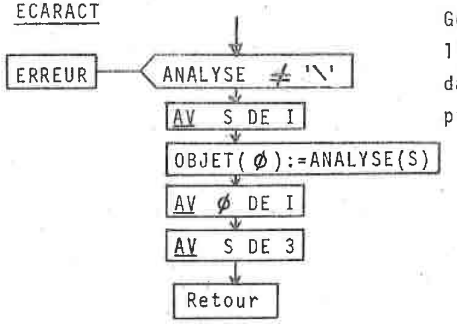
ORGANIGRAMME



par. ⑥

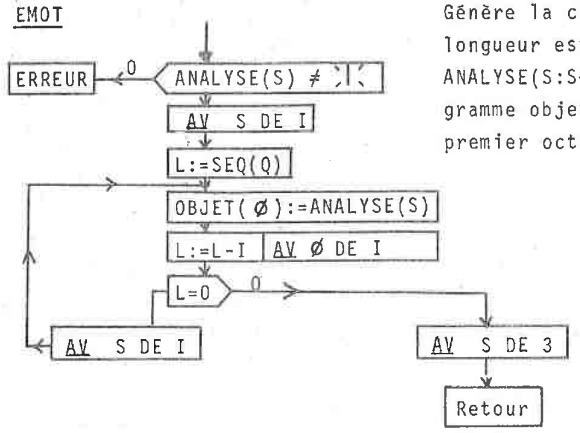
Ce paragraphe est constitué des différents traitements spécifiques des modes possibles ; chacun étant étiqueté dans MTZD.

ECARACT



Génère le caractère repéré par l'index S dans la zone ANALYSE, dans le premier octet libre du programme objet.

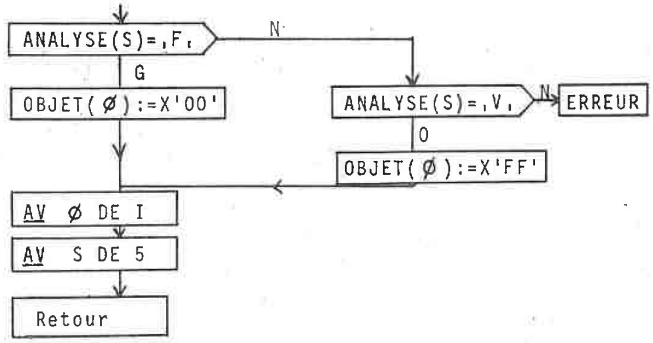
EMOT



Génère la chaîne, dont la longueur est placée dans L, ANALYSE(S:S+L-I) dans le programme objet à partir du premier octet libre.

EBOOL

Génère la constante booléen de valeur LMU VRAI ou FAUX dans le premier octet libre en y plaçant respectivement la valeur hexadécimal X'FF' ou X'00' (tous les bits à I ou sous les bits à 0).



Cas des vecteurs booléens dont le traitement est étiqueté par EBOOLEEN.

Le traitement reste à définir car d'une part une extension d'ATF ne fait pas de distinction entre vecteur de booléen et vecteur de figures binaires et d'autre part, la représentation LMU de vecteur de figures binaires n'est pas très bien définie.

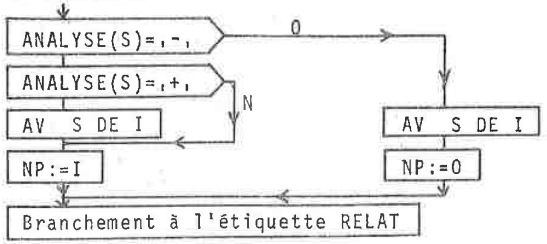
Pour l'instant le compilateur ne produit que des constantes booléennes VRAI ou FAUX et des files de telles constantes que le traducteur codifie par l'intermédiaire de EBOOL.

ENATUREL - ERELATIF

Les codifieurs étiquetés par ENATUREL et ERELATIF permettent la codification en binaire d'une constante entière positive ou négative exprimée sous la forme d'une chaîne de chiffres décimaux de longueur $\leqslant 8$, précédée d'un signe ou pas.

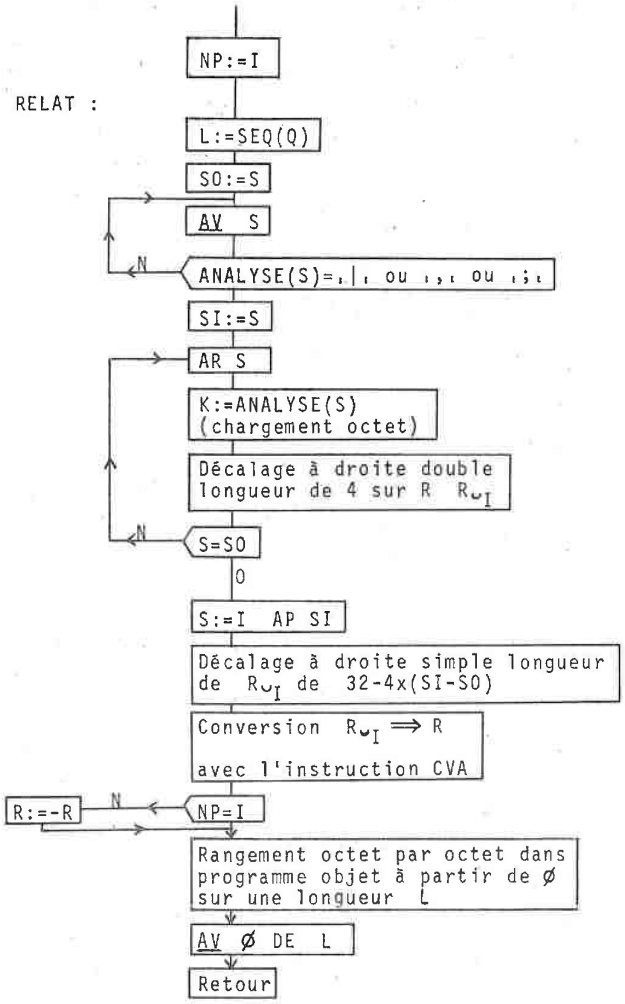
ERELATIF — positionne la mémoire d'ordre NP à I ou 0 suivant que la constante est $\geqslant 0$ ou < 0 et effectue un branchement à l'étiquette RELAT dans ENATUREL.

ORGANIGRAMME DE ERELATIF



ENATUREL
- détermine la chaîne décimale (chaque chiffre occupe I caractère sur I octet)
- compactifie la chaîne dans un registre $R_{\cup I}$ (R pair) (chaque chiffre sur I/2 octet).
- effectue la conversion
- génère la constante dans le programme objet avec complémentation à 2 si elle est < 0.

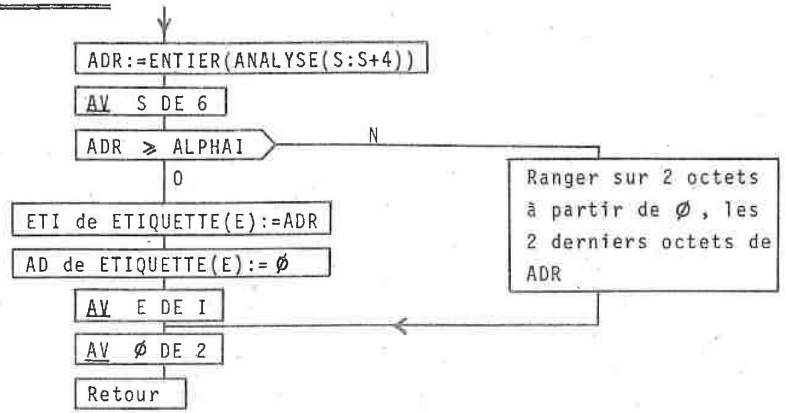
ORGANIGRAMME DE ENATUREL



EADRESSE

Ce traitement permet la codification sur 2 octets d'une constante d'adresse représentée par une chaîne de 5 chiffres décimaux ; suivant que sa valeur binaire est ⩾(ALPHAI) (respectivement comprise entre 0 et (ALPHAI)) elle représente une étiquette symbolique (respectivement une adresse relative dans le programme objet).

ORGANIGRAMME



. ADR est une mémoire (ou un registre) dans laquelle est placée la valeur binaire de la chaîne de 5 chiffres décimaux, convertie.

. Le traitement se réfère à la table des étiquettes insatisfaites, ETIQUETTE munie de son index E.

EREEL

Le compilateur ATF ne produit pas jusqu'alors des représentations de nombres réels, en conséquence nous n'aborderons pas dans les détails le codifieur EREEL.

Le IO 070 possède une arithmétique virgule flottante qui offre des instructions opérant sur des nombres virgule flottante à format court (I mot) ou long (2 mots). Le signe et l'exposant occupe toujours les 8 bits de poids fort ; le reste est réservé à la mantisse.

LMU, théoriquement, permet l'emploi de nombres réels - plus exactement des approximations de nombres réels par des expressions de nombres décimaux suivis de la lettre e, précédant une constante qui désigne un entier relatif (exposant). La codification, en machine théorique, d'une telle expression se fait sur des longueurs variables d'une part pour la mantisse et d'autre part pour l'exposant - ce qui pour ce dernier cas semble déjà trop général.

En conséquence, pour utiliser au mieux l'arithmétique virgule flottante du IO 070, il est nécessaire que les longueurs de codification, d'une part, de l'exposant soit de une case (sur octet), d'autre part, de la mantisse soit de 3 cases (format court) ou de 7 cases (format long) ; ce qui entraine une restriction au niveau d'A.T.F..

Occupation des registres de l'ordinateur 10 070

Un programme L.M.U. utilise fondamentalement les registres R0,R1,R2,R3,R4,I,D. Lors d'une exécution, ces registres sont représentés par des registres de l'ordinateur 10 070. A la traduction, il est nécessaire de connaître les adresses (absolues) de ces registres, afin de pouvoir générer des instructions machine. Ces registres n'ont pas été choisis parmi les registres d'index (registres d'adresse 1,2,...,7) car lors d'un calcul d'adresse ils contiennent des adresses en octet, alors que l'adresse d'un opérande d'une instruction 10 070 s'effectue à partir d'une adresse de mots, indexée éventuellement par un registre d'index, qui contient alors un rang exprimé sous forme d'octets ou de demi-mots ou de mots ou de double-mots, suivant le type de l'instruction.

Le registre I, L.M.U., se dédouble en un registre I 10 070 contenant une adresse en octets et en un registre RI 10 070 contenant la même adresse en mots. Ce dernier est utilisé particulièrement dans les instructions de branchement comme registre d'index.

La correspondance des noms de registre avec leurs adresses, se trouve dans la table TREG.

TREG

R 0	8
R 1	9
R 2	10
R 3	11
R 4	12
I	13
D	15

1 octet 1 octet 1 demi-mot

Le registre RI contenant une adresse en mots, est situé à l'adresse ④ .

MODULE DE TRAITEMENT DE LA ZONE INSTRUCTION | MTZI

Introduction

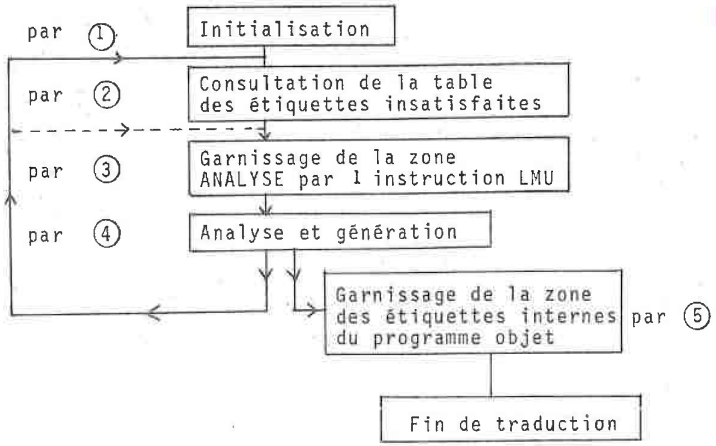
Le module de traitement de la zone des instructions LMU, MTZI, opère sur une instruction LMU après l'autre.

Son rôle est de traduire chaque instruction LMU en une séquence d'instructions machine IO 070.

L'instruction LMU se présente sous la forme d'une chaîne de caractères se terminant au premier point-virgule rencontré.

La séquence résultante est placée à la suite des précédentes en mémoire centrale dans la zone OBJET.

Organigramme général



Les différentes parties de MTZI

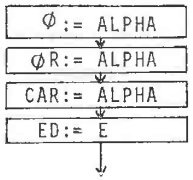
par ①

Le module MTZI prend le relai du module MTZD lorsque celui-ci a épuisé la zone des définitions du programme LMU. Comme le module MTZI ne génère que des instructions IO 070 chacune sur un mot, l'index Ø de la zone OBJET va prendre comme valeur, l'adresse relative du premier mot libre dans cette zone.

Un deuxième index ØR va contenir l'adresse relative du premier mot libre par rapport à l'origine du programme Objet ; car il se peut que la zone OBJET, limitée, ne puisse contenir toute la traduction du programme LMU ; lors d'un débordement, cette zone est vidée sur disque et la génération continue alors à partir du premier mot libre de la zone.

Un compteur d'adresses relatives CAR est initialisé à l'adresse de la première case libre suivant la zone codifiée des données. Il contient l'adresse théorique de l'instruction LMU en cours d'analyse. Il progresse de 6 pour chaque instruction LMU analysée. La valeur 6 correspond à la longueur (en cases) de codification des instructions LMU dans la machine théorique LMU. L'index ED de la table ETIQUETTE pointe initialement vers le dernier enregistrement rempli par MTZD.

Organigramme



par. ②

Tant qu'il reste au moins une étiquette non satisfaite dans la table ETIQUETTE, le module MTZI fait exécuter cette partie à chaque nouvelle instruction LMU à analyser ; sinon il se branche directement à la partie ③ . L'étiquette implicite de l'instruction LMU est dans CAR. L'adresse relative du premier mot libre est dans ØR.

Une consultation de la table ETIQUETTE permet de déterminer, si à l'étiquette implicite de l'instruction LMU, correspond une étiquette insatisfaite de la table ; dans ce cas la valeur de l'étiquette dans la table prend la valeur du compteur ØR.

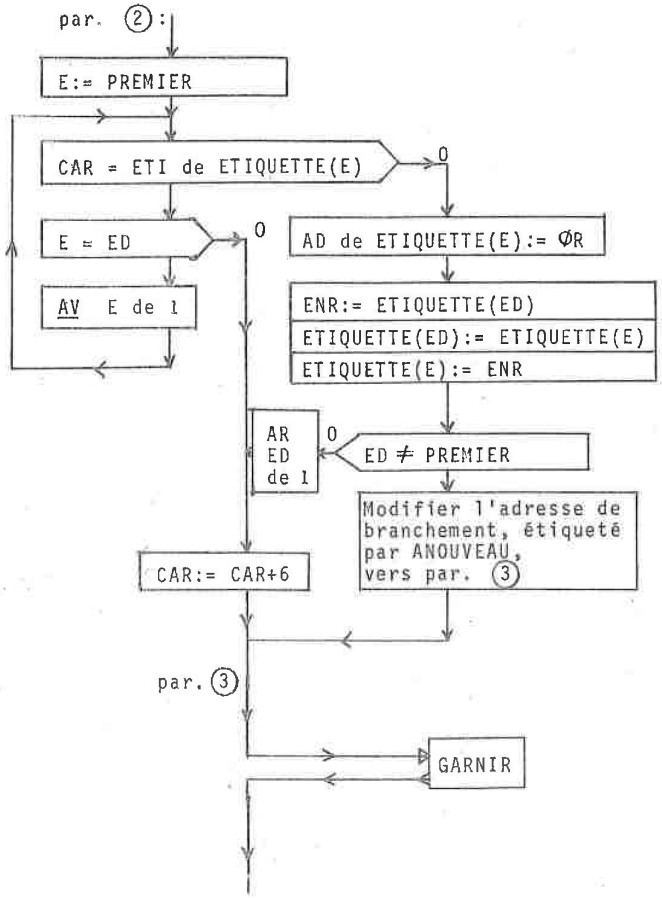
par. ③

Le garnissage de la zone ANALYSE par une instruction LMU, se fait par le sous-programme GARNIR.

* ENR est une mémoire de travail qui permet de charger un enregistrement de la table ETIQUETTE.

* ANOUVEAU étiquette une instruction de branchement pour une réitération de MTZI. Tant qu'une étiquette au moins reste insatisfaite, le branchement s'effectue vers la partie ② ; dans l'autre cas le branchement a lieu vers la partie ③ .

Organigramme de par. ② et par. ③



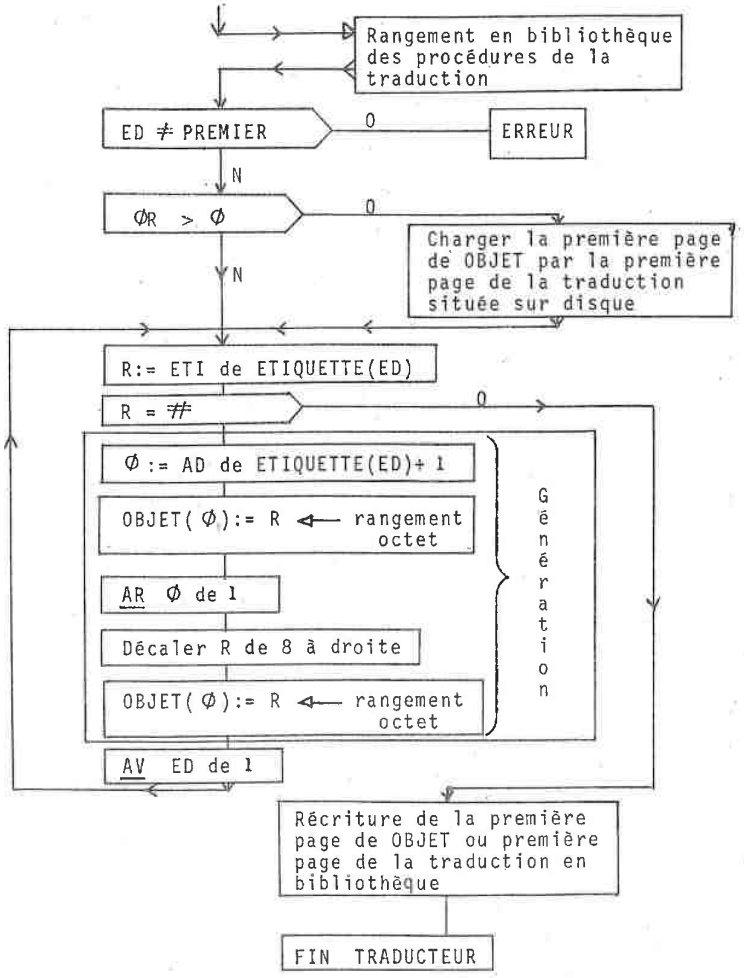
par. ⑤

Une fois traduite la zone des instructions, par la partie ④ (cf. plus loin), la partie ⑤ est exécutée.

Ce paragraphe range la ou le reste de la traduction située dans la zone OBJET, dans la bibliothèque des traductions de procédures située sur disque. Puis il effectue la codification des étiquettes internes dans le programme objet. Pour cela, il dispose de la première page de la traduction, située ou chargée dans la première page de la zone OBJET. Il dispose d'autre part de la table ETIQUETTE. Cette table contient pour chaque étiquette sa valeur d'adresse (relative) en mots (ETI) et son adresse relative de début d'implantation dans la traduction, adresse exprimée en octets (AD). La valeur d'adresse d'une étiquette est codifiée sur 2 octets.

L'exécution du paragraphe ⑤ , permet de tester si toutes les étiquettes ont été satisfaites puis de garnir la zone des étiquettes dans la traduction. La première page de la zone OBJET est ensuite réécrite en bibliothèque.

Organigramme



par. ④

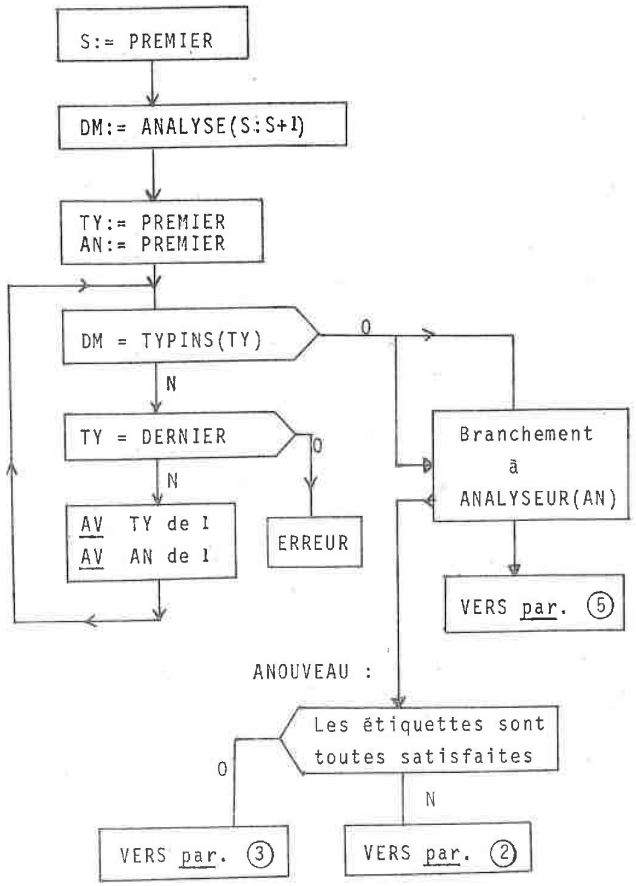
I. Reconnaissance du type de l'instruction

L'analyse d'une instruction LMU placée en zone ANALYSE commence par une reconnaissance du type de l'instruction. Cette reconnaissance s'effectue sur les 2 premiers caractères de l'instruction et permet d'aiguiller l'analyse vers l'exécution d'un traitement spécifique du type de l'instruction.

Pour cela, la table TYPINS contient tous les couples de caractères susceptibles de se trouver en début d'une instruction LMU, avec en parallèle dans la table ANALYSEUR toutes les adresses des traitements correspondants (analyseurs). La recherche se faisant par le haut de la table les couples de caractères sont rangés de haut en bas par ordre (a priori) de fréquences décroissantes.

TYPINS	ANALYSEUR
C (	AN . C(
C R	AN . CR
C I	AN . CI
C D	AN . CD
S I	AN . SI
S T	AN . ST
V E	AN . VE
V I	AN . VI
E R	AN . ER

Organigramme de reconnaissance du type d'instruction



2. Les analyseurs d'instructions LMU

Deux classes d'analyseur sont disponibles, suivant la complexité des instructions LMU.

Les analyseurs AN:CI, AN:CD, AN:VE, AN:ER et AN:ST constituent la première classe ; chacun détermine si l'instruction LMU placée dans la zone ANALYSE répond au profil respectif imposé puis effectue la traduction de l'instruction en générant une séquence d'instruction IO 070.

La deuxième classe est constituée des analyseurs AN:SI, AN:CR et AN:C(. Chacun effectue l'analyse et la traduction de l'instruction LMU qui peut alors avoir plusieurs profils possibles.

Remarque :

La rencontre dans les organigrammes de `AV Ø de 1` signifie un appel de sous-programme :
Si Ø = DERNIER alors vider la zone OBJET en bibliothèque des procédures et Ø:= PREMIER ; sinon avancer Ø de 1 .

2.I. La première classe d'analyseurs

`AN:CI`

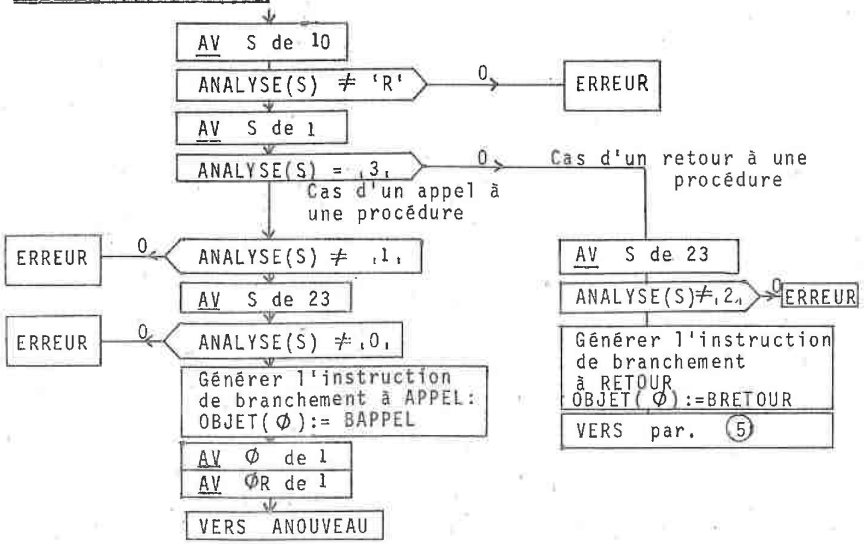
Une instruction LMU commençant par 'CI' a le profil suivant :

```
'CI:=NC(0+CRX+CRO),(2+CRO)VERS(0+CRX+CI)'
        ↑                      ↑
        X1                     X2
```

suivant la valeur de X1 , AN-CI détermine si l'instruction correspond à un appel de procédure (X1='1',X2='0'), ou bien à un retour de procédure (X1=3,X2=2) ; dans le dernier cas l'instruction LMU est la dernière du programme LMU.

AN:CI, traduit l'instruction LMU en générant dans le premier mot libre de la zone OBJET une instruction de branchement inconditionnel, indirect soit vers MAPPEL lors d'un appel, soit vers MRETOUR lors d'un retour. MAPPEL et MRETOUR sont des adresses fixes, qui à l'exécution d'une procédure désignent des mots mémoires contenant respectivement : l'adresse des entrées APPEL et RETOUR du programme SYST.

Organigramme de AN:CI



BAPPEL et BRETOUR sont des mots-mémoire de AN:CI qui contiennent une instruction de branchement inconditionnel indirect avec respectivement comme opérande l'adresse de MAPPEL et de MRETOUR.

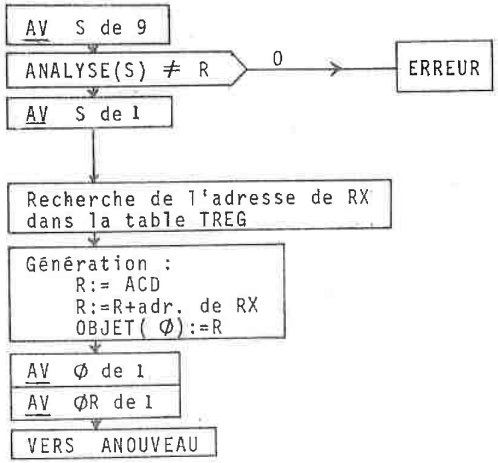
AN:CD

Une instruction LMU commençant par 'CD' a le profil suivant :

'CD:=N(0+CRX+CD)'
↑
X

CD désigne le contenu du registre D du IO 070 auquel on additionne le contenu du registre RX. Une consultation de la table des registres TREG permet de déterminer l'adresse de RX.
AN-CD génère alors l'instruction d'addition dans le premier mot libre de la zone OBJET.

Organigramme



* R est un registre auxiliaire de la IO 070
* ACD est un mot de AN-CD, contenant le profil d'une instruction d'addition au registre D (sans opérande)

AN:VI

Cet analyseur traitant l'instruction LMU : VIDE, se résume à l'instruction de branchement vers ANOUVEAU.

VERS ANOUVEAU

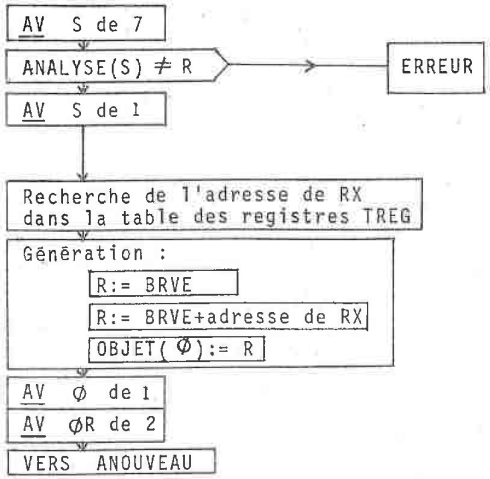
AN:VE

Le profil de l'instruction de branchement est :

'VERS(O+CRX+CI)'
X

Une consultation de la table des registres TREG permet de déterminer l'adresse du registre RX.
AN:VE génère alors une instruction de branchement inconditionnel indexé par RI.

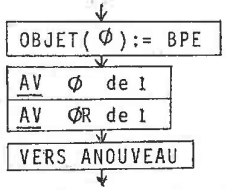
Organigramme



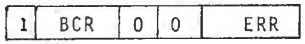
* BRVE contient dans un mot :

AN:ER

L'instruction LMU, ERREUR est reconnue et traduite par l'analyseur qui génère une instruction de branchement indirect inconditionnel vers un sous-programme SPERREUR.



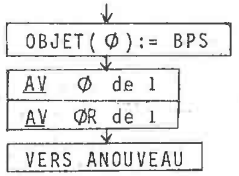
* BPE est une mémoire contenant l'instruction de branchement indirect vers SPERREUR.



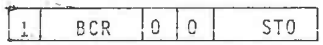
* ERR est l'adresse absolue de la mémoire contenant l'adresse du sous-programme SPERREUR.

AN:ST

L'instruction LMU, STOP est reconnue et traduite par l'analyseur qui génère une instruction de branchement indirect inconditionnel vers un sous-programme SPSTOP.



* BPS est une mémoire contenant l'instruction de branchement indirect vers SPSTOP.



* STO est l'adresse absolue de la mémoire contenant l'adresse du sous-programme SPSTOP.

2.2. La deuxième classe d'analyseurs

La structure d'une instruction LMU traitée par les analyseurs de cette classe dépend :

- du type de l'instruction,
- du nombre d'opérandes, de leur nature, de l'opérateur s'il existe,
- de la forme des opérandes.

Durant la première phase, celle de l'analyse, chaque analyseur effectue :

- L'analyse de chaque opérande par l'appel du sous-programme commun, PTO. Ce sous-programme génère, en particulier, les instructions de calcul d'adresse de l'opérande ou de sa valeur lorsqu'il est une constante et garnit la table OPER précisant la forme des opérandes et leur longueur.
- Le garnissage de mémoires auxiliaires relatives à la nature des opérandes, à l'opérateur.

La deuxième phase est celle du traitement effectif de l'instruction :

- Exploitation des résultats et génération des instructions de traitement des opérandes.

Le programme de traitement d'un opérande PTO

Le programme PTO, effectue le traitement de l'opérande en fonction de la forme de celui-ci.

Pour cela, il dispose de la table des registres TREG - donnant la correspondance : registres LMU ⟶ registres IO 070 - de l'adresse des registres AI,A2,A3 et de la table OPER.

Rôle de AI,A2,A3

Les registres AI,A2,A3 correspondent aux trois opérandes possibles d'une instruction LMU. Le registre Ai reçoit à l'exécution soit l'adresse du $i^{ème}$ opérande de l'instruction LMU, soit sa valeur lorsque celui-ci est une constante ou une expression d'adresse.

Rôle de la table OPER

La table OPER comporte 3 enregistrements correspondant aux trois opérandes possibles d'une instruction LMU.

Pour le $i^{ème}$ opérande, l'enregistrement comporte :

- un octet BETA qui contient, lorsque l'opérande a une expression de longueur (β+CRO) ou C(β+CD), la valeur de β.
- un octet INDIC. Les bits 0-1 sont positionnés en fonctio du contenu de Ai.

00	constante ou expression d'adresse Le registre Ai contient une valeur.
01	expression de registre Le registre Ai contient une adresse en mot.
10	expression de contenu avec expression de longueur Le registre contient une adresse en octet.

Le bit-7 est positionné à 1 lorsque l'expression de longueur de l'opérande (si elle existe) est de la forme C(β+CD).

PTO opère sur le $i^{ème}$ opérande de l'instruction LMU d'une des manières suivantes :

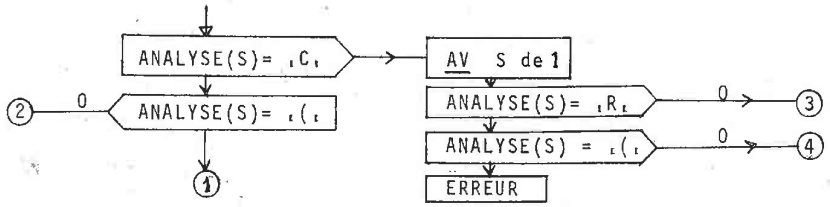
I) Si l'opérande est une constante (forme F1).
- Après conversion de la constante en fonction de son type, PTO génère une instruction de chargement immédiat plaçant la valeur de la constante dans le registre Ai.
- Positionne les bits 0-1 de INDIC(i).

2) Si l'opérande est une expression d'adresse (de la forme F2 : $(\alpha \pm C\rho 1 + C\rho 2)$)
- PTO génère les instructions de calcul d'adresse dont le résultat se trouve placé à l'exécution dans le registre Ai.
- Positionne INDIC(i).

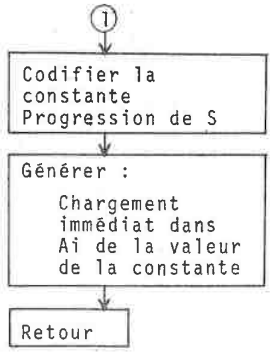
3) Si l'opérande est une expression de registre (de la forme F3 : $C\rho$)
- PTO génère une instruction de chargement immédiat plaçant l'adresse du registre (en mot) dans le registre Ai.

4) Si l'opérande est une expression de contenu (de la forme F4 : $C(\alpha + C\rho 1 + C\rho 2)$, suivi de l'expression de longueur (β+CRO) ou $C(\beta + CD)$).
- PTO - génère les instructions de calcul de l'adresse de l'opérande,
- positionne les bits 0-1-7 de INDIC(i),
- range la valeur de β dans BETA(i).

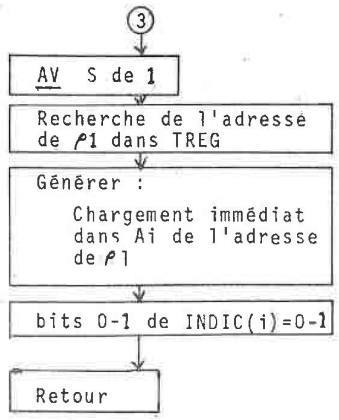
ORGANIGRAMME DE PTO



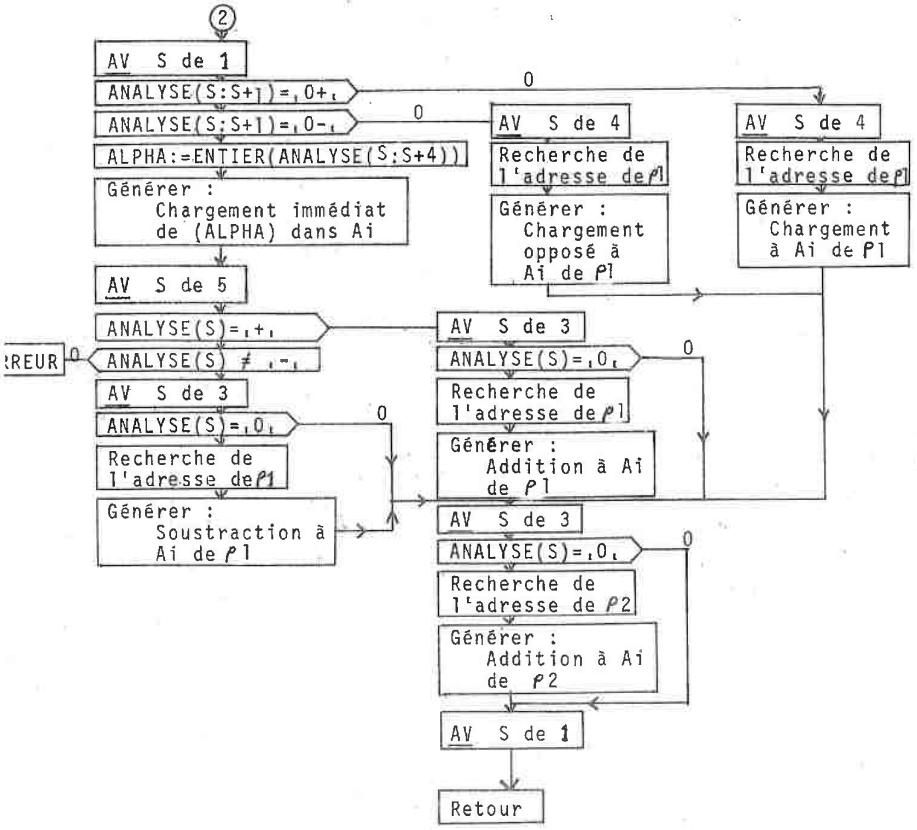
Constante



Cρ1

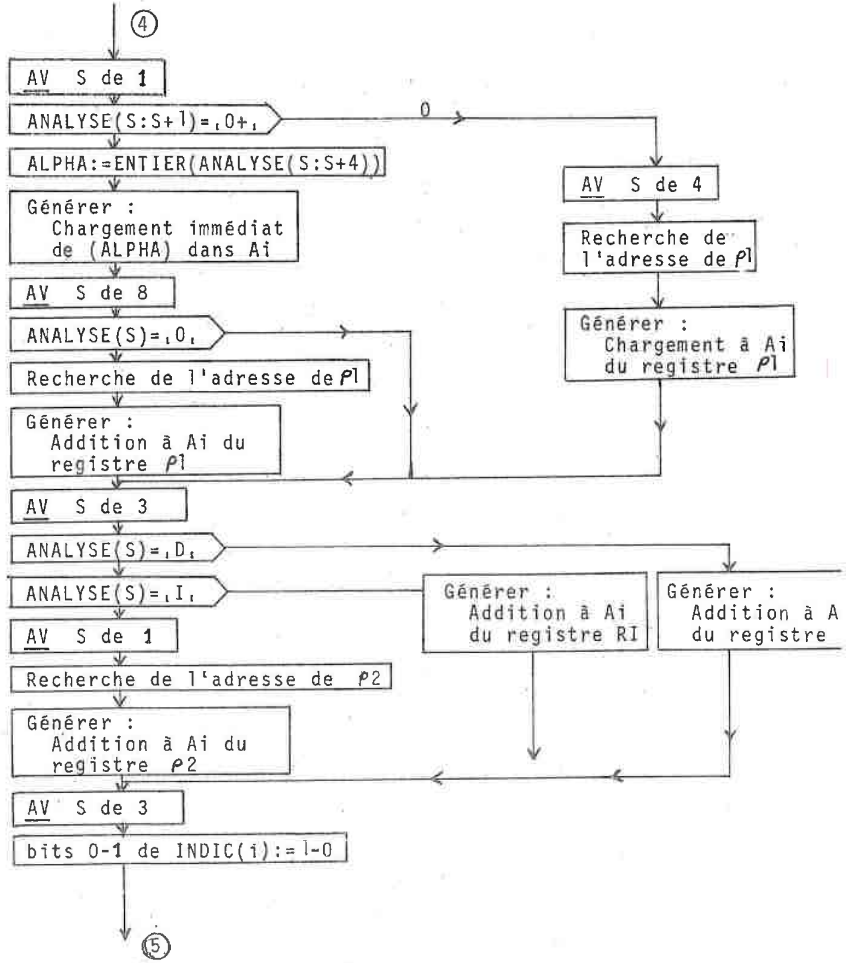


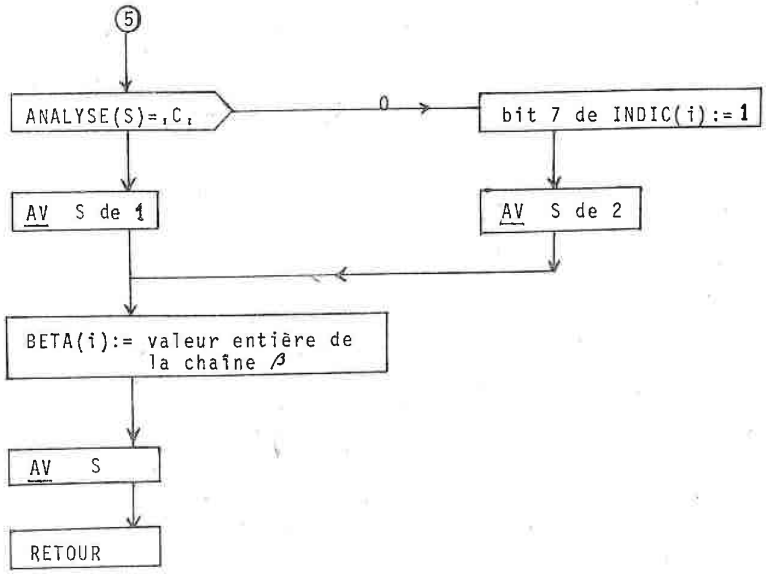
- $(\alpha \pm C\rho 1 + C\rho 2)$



- $C(\alpha + C\rho 1 + C\rho 2)$, $(\beta + CRO)$

ou $C(\alpha + C\rho 1 + C\rho 2)$, $\mathcal{E}(\beta + CD)$





L'analyseur AN:SI

L'analyseur AN:SI opère sur l'instruction L.M.U. de branchement conditionnel qui est de la forme :

SI expression booléenne VERS ε

- Expression booléenne.

Soit Δ , Δ' désignant chacun un opérande de l'une des formes F1,F3,F4
(cf. programme de traitement d'un opérande).

Soit $\perp$,θ désignant respectivement un opérateur et la nature des opérandes :

mot : $\theta = M \longrightarrow \perp \in \{=, \neq\}$

booléen : $\theta = B \longrightarrow \perp \in \{\neg, \wedge, \vee\}$

booléen, naturel : $\theta \in B,N,R \longrightarrow \perp \in \{=, <, >, \leq, \geq, \neq\}$

Alors une expression booléenne est de l'une des formes

- Δ_B
- $\neg_B \Delta_B$
- $\Delta_\theta \perp_\theta \Delta'_\theta$

- VERS ε

ε = (0+Cρ1+CI)

VERS ε peut être remplacé par ERREUR.

L'analyse de l'expression booléenne garnit les mémoires OPERATEUR et NATURE et appelle une ou deux fois le programme PTO, suivant que l'expression possède un ou deux opérandes.
La rencontre dans l'instruction du premier caractère de ,VERS, ou ,ERREUR, détermine la fin de l'expression booléenne. L'analyseur appelle alors un des sous-programmes générateur, chargé de pro-

duire les instructions de calcul de l'expression booléenne, positionnant le code condition CC du 10 070. Le choix du sous-programme se fait en fonction de la forme de l'expression booléenne à l'aide de l'aiguillage AGSI ; les renseignements sur la forme de l'expression sont dans les bits 0-1 de INDIC(1) et de INDIC(2).

Le sous-programme rend le contrôle au programme appelant. AN:SI poursuit alors l'analyse de la partie adresse de branchement (VERS £). Il génère une instruction de branchement conditionnel indirect ainsi construite :

- la partie code opération et test code condition est construite grâce à la table T-OPERATEUR en fonction de l'opérateur de l'expression booléenne.
- la partie adresse est déterminée
  - soit à partir de l'adresse du registre ℘1 indexé par le registre I lorsque l'opérande L.M.U. est (0+C℘1+CI),
  - soit à partir de l'adresse du sous-programme SPERREUR, lorsque l'opérande est ERREUR.

L'aiguillage AGSI

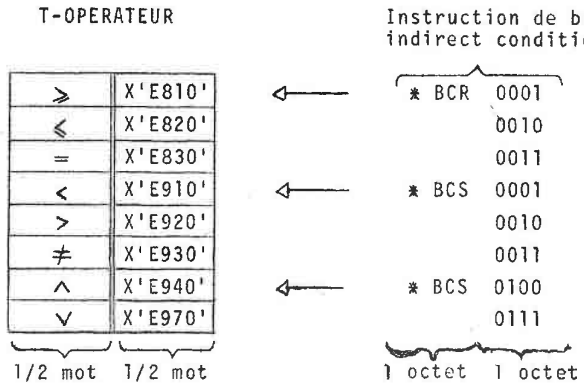
AGSI est une table qui contient sur chaque enregistrement (1 mot), l'adresse d'un des sous-programmes codifieurs ou zéro. Chaque codifieur correspond à une forme possible d'expression booléenne. La valeur zéro correspond à une forme inexistante.

Lorsque l'expression booléenne comporte deux opérandes l'index dans l'aiguillage vaut la valeur binaire constituée des bits 0-1 de INDIC(1) réunis aux bits 0-1 de INDIC(2), la valeur zéro n'étant pas possible. Lorsque l'expression booléenne comporte un seul opérande l'index vaut zéro (la forme F4 étant seule possible).

l'opérateur d'une expression booléenne,

. le deuxième contient le code opération et le test code condition d'une instruction de branchement.

La recherche dans cette table du représentant de l'opérateur d'une expression booléenne, permet de construire l'instruction de branchement indirect conditionnel.



Rappelons que les formes F1-F3-F4 correspondent respectivement à des expressions - de constante - de registre - de contenu - positionnant des bits 0-1 de INDIC respectivement à -00-01-10.

Structure de l'expression booléenne	AGSI	Valeur d'index
F4	A F4:	00 00
F1 F3	A F1:F3	00 01
F1 F4	A F1:F4	00 10
inexistante	0	00 11
F3 F1	A F3:F1	01 00
F3 F3	A F3:F3	01 01
F3 F4	A F3:F4	01 10
inexistante	0	01 11
F4 F1	A F4:F1	10 00
F4 F3	A F4:F3	10 01
F4 F4	A F4:F4	10 10
	1 mot adresse des codifieurs AFi:Fj	bit 0-1 de INDIC(1) bit 0-1 de INDIC(2)

FG est un index de la table.

La table T-OPERATEUR

Chaque article de la table se décompose en deux demi-mots :

. le premier contient l'un des caractères indiquant

Organigramme de AN:SI

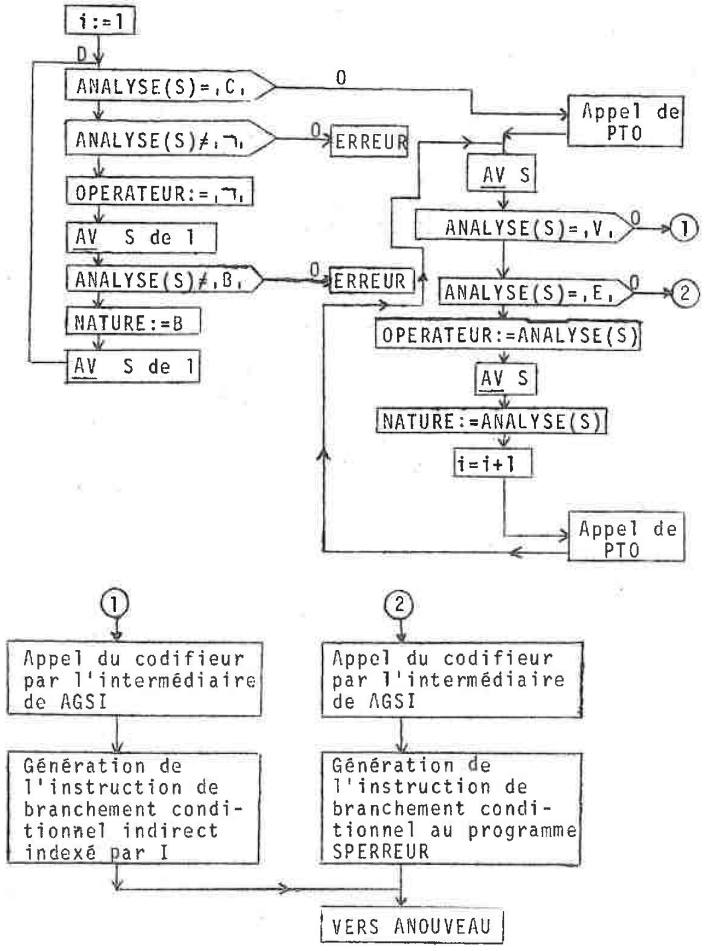
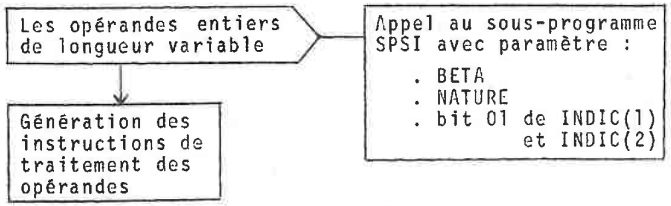


Schéma très général d'un générateur de AN:SI



Remarque importante sur la suite du traducteur

La rédaction des générateurs de AN:SI, des analyseurs AN:CR et AN:C(n'a pu se terminer faute de temps.

Un exposé général de SPSI et de AN:CR et de AN:C(termine la fin de ce chapitre.

Le sous-programme SPSI

Le sous-programme SPSI permet de traiter à l'exécution les expressions booléennes (à 2 opérandes) d'une instruction SI, dont la séquence de traduction s'avère trop longue ; cette séquence est remplacée par une séquence d'appel à SPSI. Deux générateurs de l'analyseur AN:SI génèrent dans certains cas une telle séquence : le générateur F4:F1 lorsqu'il traite un opérande avec une longueur variable, le générateur F4:F4 dans tous les cas sauf celui où il opère sur 2 chaînes de caractères de même longueur connue à la traduction. Le sous-programme SPSI, après avoir calculé la longueur du premier opérande à partir des paramètres situés dans la séquence d'appel teste la forme de l'expression booléenne.

- Lorsque la forme est F4 F1, SPSI range le premier opérande dans le registre 0, il le cadre à droite par un décalage arithmétique ou logique suivant que l'opérande est de nature entière ou pas. Le deuxième opérande (la constante K) se trouvant dans la séquence d'appels, SPSI effectue alors la comparaison avant de rendre le contrôle au programme appelant.

- Lorsque la forme est F4 F4, SPSI calcule la longueur du deuxième opérande. Lorsque la nature des opérandes est entière, ils sont rangés et cadrés à droite par décalage arithmétique , le premier dans le registre 0, le second dans la mémoire MOT, SPSI effectue alors la comparaison. Lorsque la nature des opérandes n'est pas entière, une comparaison sur les longueurs des opérandes est effectuée. Si les longueurs sont identiques, une comparaison sur les chaînes d'octets des 2 opérandes est effectuée avant le renvoi au programme appelant. Dans le cas contraire, il y a renvoi au programme appelant (le code condition CC, étant positionné).

AN.CR et AN.C(

Les deux analyseurs AN.CR et AN:C(, traitant les instructions d'affectation L.M.U. à deux et trois opérandes.

Les instructions d'affectation à deux opérandes sont constituées par :

- les instructions de transfert
- les instructions comportant un opérateur unaire s'appliquant sur le deuxième opérande.

Les instructions d'affectation à trois opérandes comportent un opérateur binaire s'appliquant sur le deuxième et le troisième opérande.

Elles comprennent :

- les instructions à opérateur booléen
- les instructions à opérateur arithmétique.

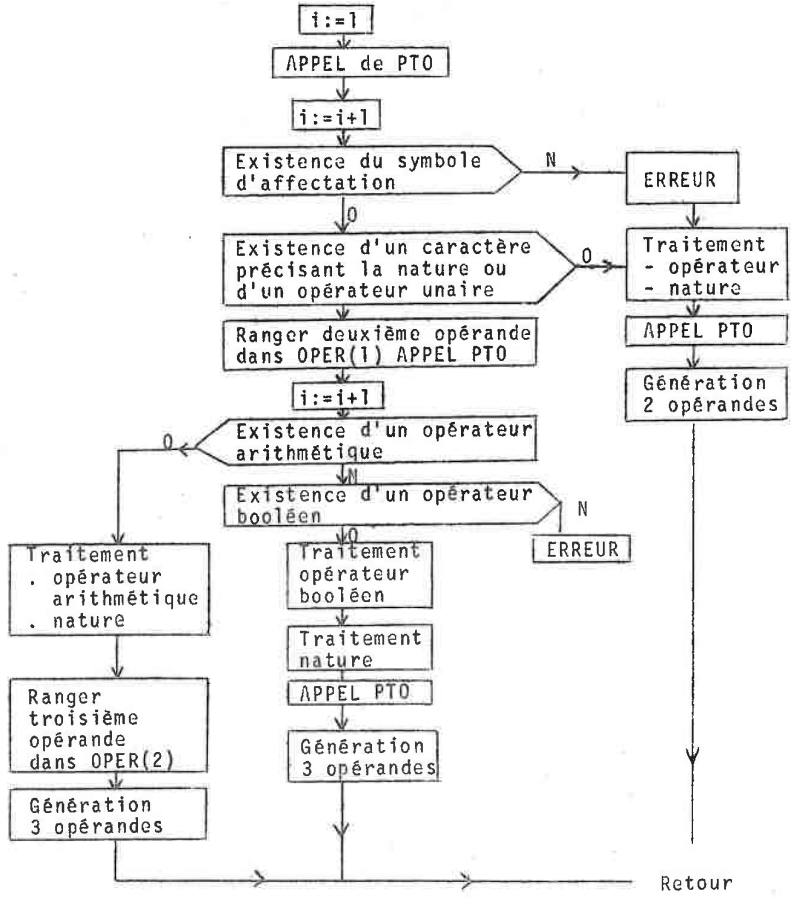
Les deux analyseurs ne se différencient que lors de l'analyse du premier opérande jusqu'à la rencontre du symbole d'affectation ':='. La partie commune analyse le reste de l'instruction L.M.U.

Deux cas se présentent :

1) La rencontre d'un caractère (ou deux suivant le cas), précisant un opérateur ou la nature des opérandes, permet d'aiguiller l'analyseur vers le traitement des instructions à deux opérandes. Après codification du restant de l'instruction L.M.U., l'analyseur appelle un sous-programme générateur en fonction de la forme des opérandes.

2) Le deuxième cas est celui des instructions à trois opérandes. Le deuxième opérande est analysé et codifié. Puis l'analyse donne la connaissance de l'opérateur binaire et de la nature des opérandes. Le troisième opérande est codifié. Alors suivant que l'opérateur est booléen (respectivement arithmétique) l'analyseur appelle les générateurs correspondants.

Organigramme général de AN:CR et AN:C(



CHAPITRE V

PROGRAMME DE CONTROLE
DE L'EXECUTION DES PROCEDURES

Introduction

L'exécution d'un traitement immédiat suppose théoriquement la présence en mémoire centrale, d'une part des traductions de toutes les procédures susceptibles d'être appelées, et d'autre part de toute la file des données nécessaire au déroulement. Cette file est constituée des files de données respectives des procédures en cours. Elle évolue de façon dynamique à l'exécution au fur et à mesure des appels des procédures. Elle a donc une structure de pile. Lors d'un appel d'une procédure, l'enchainement des programmes et la substitution des opérandes aux arguments se fait théoriquement de manière automatique.

En réalité, l'espace disponible en mémoire centrale étant limité, il est hors de question d'espérer pouvoir y mettre dans tous les cas, toutes les traductions et la file des données nécessaires au traitement.

Nous ferons donc intervenir un programme de gestion de l'espace disponible en mémoire centrale. Ce programme,appelé SYST, sera le seul (à l'exception de IMPLAN présenté plus loin) connu du système IO 070. Son rôle est, d'une part de gérer l'espace mémoire disponible divisé en deux zones ; la première, zone des données, est réservée à la file des données, la seconde, zone des instructions, est réservée aux procédures à exécuter. D'autre part, SYST doit charger et enchainer les procédures et les faire exécuter ; pour cela il dispose d'un fichier sur disque contenant toutes les procédures existantes.

Le partage de la mémoire en blocs ou pages permet une simplification dans la gestion. Les blocs de 512 mots ont été retenus de telle sorte qu'ils s'identifient aux pages de la mémoire du IO 070 et permettent ainsi d'utiliser le système de verrouillage des pages.

Le programme SYST va devoir tenir compte de deux contraintes :

I. La présence en mémoire centrale de la totalité de la file des données, nécessaire à un instant quelconque de l'exécution. Cette condition est nécessaire parce que l'appel des paramètres de procédures s'effectue par "nom" ; c'est à dire que figure dans la file de données d'une procédure, l'adresse des paramètres et non pas leurs valeurs, celles-ci se trouvant à l'extérieur dans la file globale.

2. La présence en mémoire centrale de toute la procédure en cours d'exécution. Cette contrainte pourrait être affaiblie en ne demandant en mémoire centrale que la présence de la page de procédure en cours d'exécution. Cette option n'a pas été retenue afin de simplifier à la fois la gestion de la mémoire et les programmes-objets des procédures.

Ces deux contraintes peuvent entrainer des conflits dans l'allocation de la mémoire au moment du chargement d'une procédure ou au moment d'une extension de la file des données. En effet, la taille de la file des données peut augmenter à tout moment de l'exécution ; la file peut donc être amenée à déborder dans la zone des instructions, ce qui nécessite l'intervention de SYST. Pour déceler automatiquement cette situation, nous utilisons le système IO 070 de protection contre l'écriture. Nous avons décomposé le programme de contrôle de l'exécution d'un traitement en deux parties :

- SYST autorise à écrire dans tout l'espace utilisateur sauf dans la première page de la zone des instructions.

- IMPLAN programme s'exécutant en mode maître, est chargé de tenir à jour la protection de la mémoire.

La procédure en cours d'exécution, seule susceptible de provoquer un débordement de la file des données, s'exécute avec le "double mot d'état" de SYST : ainsi une tentative d'écriture en page protégée, due à un débordement de la file des données, provoquera un "déroutement" vers IMPLAN. Le programme IMPLAN déverrouille alors la page protégée, verrouille la suivante et rend ensuite le contrôle à SYST. SYST alloue la page déverrouillée à la zone des données et effectue un branchement à l'instruction ayant provoquée le déroutement.

Lors d'un appel ou d'un retour d'une procédure, un branchement est effectué dans SYST. Il consulte une table appelée table d'implantation TI. Cette table contient pour chaque procédure susceptible d'être appelée, le numéro de la procédure, son adresse en bibliothèque, son volume d'occupation en pages, son adresse en mémoire centrale si elle y est présente et son nombre d'appels non terminés. Si la procédure est en mémoire, SYST attribue au registre I (registre indexant les adresses de la partie instruction d'une procédure), l'adresse de début d'implantation de la traduction de la procédure et effectue un branchement pour exécution.

Si elle n'est pas en mémoire centrale, SYST doit la charger à partir de sa copie située sur disque. L'implantation ne peut se faire que s'il se trouve de la "place" vide en mémoire centrale, c'est à dire s'il existe des pages consécutives, libres, en nombre suffisant pour recevoir la totalité de la procédure. Dans le cas où aucune place vide suffisante n'est disponible, SYST doit libérer des pages de la zone instruction en éliminant certaines procédures; on déterminera ces procédures à l'aide d'un critère fondé sur le nombre d'appels non terminés des procédures en mémoire. Cette recherche de place s'opère par consultation d'une table appelée la table d'occupation TO. Pour chaque page de la mémoire utilisatrice, cette table indique la procédure qui peut s'y trouver et

le nombre d'appels non terminés de cette dernière
(≥ 0 si la page est occupée, =-I si la page est libre).

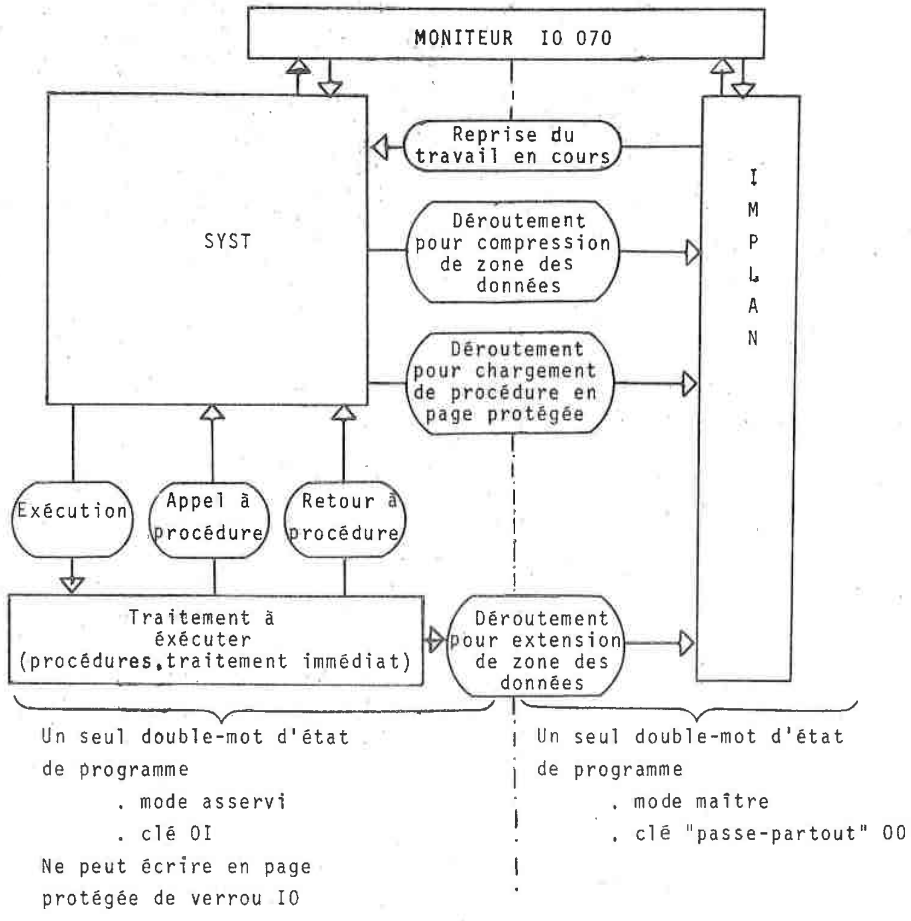
La recherche dans la table d'occupation ne s'effectue que dans la partie correspondant aux pages de la zone des instructions. SYST prévoit afin de "grossir" cette zone, de diminuer la zone des données dans le cas où le remplissage de cette dernière, par la file des données, est trop faible. SYST tente alors une écriture en page protégée ce qui provoque un déroutement vers IMPLAN. Comme dans le cas de l'augmentation de la zone des données, IMPLAN modifie les verrous, puis rend le contrôle à SYST,

Afin de pouvoir effectuer un chargement de procédure en page protégée, il est nécessaire que ce rôle ne soit pas réservé au programme SYST qui ne peut écrire dans cette page ; c'est le programme IMPLAN muni d'une clé "passe-partout" qui aura cette fois la charge d'implanter la procédure. Là encore, SYST provoque un déroutement vers IMPLAN. IMPLAN analyse l'adresse de l'instruction ayant provoquée le déroutement et détermine sa cause. Selon le cas, il charge une procédure en page protégée où il modifie les verrous pour augmenter ou diminuer la zone des données.

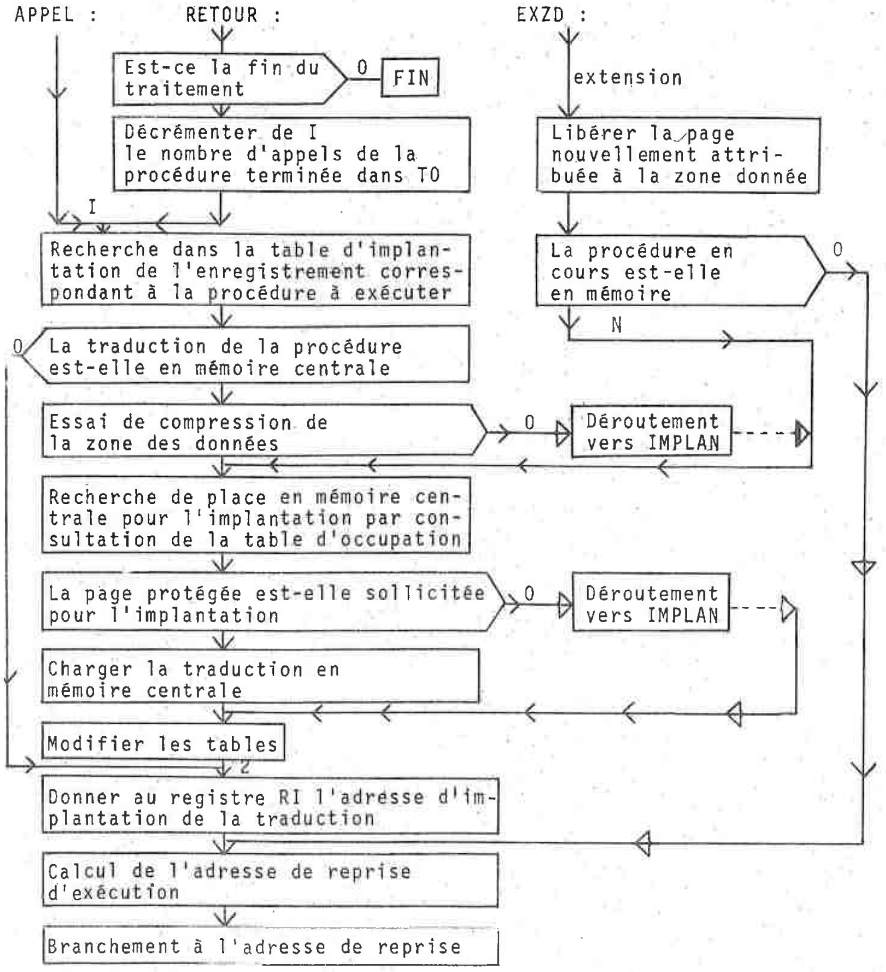
En résumé, le programme SYST est un intermédiaire entre le système de la IO 070 et les procédures traduites de l'ATF. SYST gère l'espace mémoire, charge les procédures et les fait exécuter. La frontière entre les zones des données et d'instructions est contrôlée par le système de protection par clés et verrous de la IO 070. Toute modification de cette frontière (ou le chargement d'une procédure en page protégée) est effectuée par le programme maître IMPLAN. Le contrôle est donné à IMPLAN par déroutement pour tentative d'écriture en zone protégée. IMPLAN modifie l'allocation de la mémoire ou charge une procédure en page protégée et rend le contrôle à SYST.

On peut schématiser le dispositif comme suit :

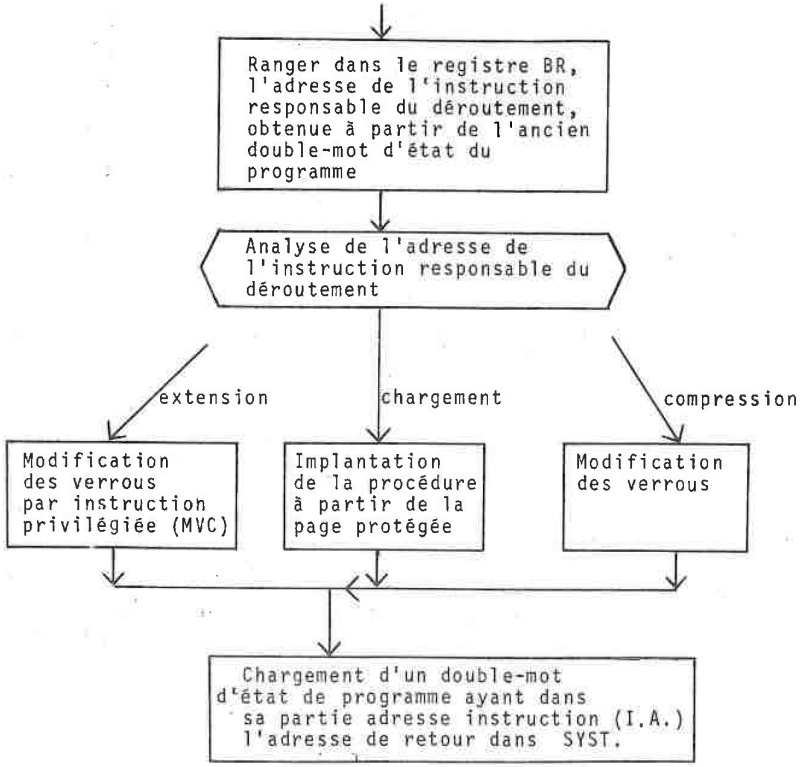
SCHEMA DES TRANSFERTS



ORGANIGRAMME GENERAL DE SYST



ORGANIGRAMME GENERAL DU PROGRAMME IMPLAN



Nous allons examiner les "ressources" mises à la disposition de SYST et IMPLAN.

BIBLIOTHEQUE DES TRADUCTIONS DE PROCEDURES - SON CATALOGUE

La bibliothèque des traductions constitue un fichier - donnée des programmes IMPLAN et SYST.

Chaque traduction de procédure se trouve sous forme binaire. Il est inutile d'effectuer des translations lors d'un chargement en mémoire en vue d'une exécution : en LMU, toutes les adresses des opérandes situés dans la file des données sont indexées par un registre D et toutes les adresses de branchement dans la file des instructions sont indexées par un registre I contenant l'adresse d'implantation de la procédure. Chaque traduction occupe un nombre entier n de blocs de 512 mots, permettant une implantation en début de page de la mémoire centrale sur n pages consécutives.

La bibliothèque comprend :

1. La traduction des procédures du compilateur ATF-LMU.
2. La traduction des procédures des utilisateurs
3. S'il y a lieu, la traduction d'un traitement immédiat en vue d'une exécution immédiate : un traitement immédiat est considéré comme une procédure sans paramètre.

Toutes les traductions des procédures sont répertoriées dans une table appelée CATALOGUE placée sur support externe. Cette table, mise à jour à chaque création ou effacement de procédure permet :

1. La construction de la "zone référence". d'une nouvelle procédure ou du traitement immédiat.
2. La construction de la table d'implantation TI précédant l'exécution d'un traitement immédiat .

STRUCTURE DE CATALOGUE

A chaque enregistrement de CATALOGUE correspond une procédure dont le numéro est égal au rang de l'enregistrement dans la table.

Chaque enregistrement est constitué de :

Zone fixe de l'enregistrement (3 mots)

1. La partie identificateur (5 mots)

   Cette zone contient l'identificateur de la procédure ATF (chaîne d'au plus 8 caractères).

2. La partie adresse disque (I/2 mot)

   Cette zone contient l'adresse (en pages) de la première page de la traduction de la procédure dans la bibliothèque sur disque.

3. La partie volume (I/4 mot)

   Cette zone contient le nombre de pages occupées par la traduction de la procédure.

4. La partie référence qui se décompose en

   a) nombre N de procédures référencées (I/4 mot)

Zone variable

   b) la zone référence
   C'est une suite (qui peut être vide) de I/2 mot , chacun contenant le numéro de la procédure dont l'identificateur apparaissait dans la zone référence du programme ATF.

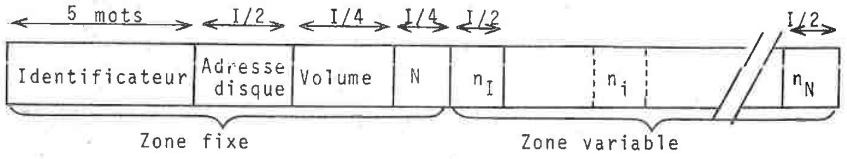


TABLE D'OCCUPATION TO

La table d'occupation est située dans le programme SYST à partir de l'adresse TO.

A chaque page d'adresse i (en pages) de la mémoire disponible va correspondre le (i+I)ème enregistrement de la table qui va rendre compte du contenu de la page (comme la mémoire disponible commence en une adresse fixe, les enregistrements correspondant aux pages précédentes pourront être utilisés à d'autres fins).

La consultation et la modification de cette table se fait par le programme SYST, lors de la recherche de place, en vue d'implanter la traduction d'une procédure dans la zone variable des instructions,

Pour cela on utilise deux mots :

- NDP qui contient le numéro de l'enregistrement correspondant à la page protégée (première page de la zone des instructions),
- PPI qui correspond à la dernière page de la zone des instructions (adresse fixe).

STRUCTURE DE L'ENREGISTREMENT

Pour chaque page, l'enregistrement est constitué :

I. d'une partie RTI contenant le rang dans la table d'implantation, de l'enregistrement correspondant à la procédure si celle-ci est implantée dans la page, zéro sinon.

2. d'une partie NA (I/4 mot) contenant soit la valeur du nombre d'appels non terminés de la procédure, si sa traduction occupe la page, soit moins un. Il est

à remarquer qu'un nombre d'appels nul correspond à une procédure terminée dans tous ses appels, mais dont la traduction est encore en mémoire centrale.

TABLE D'IMPLANTATION TI

Avant l'exécution d'un traitement immédiat on établit une table des procédures susceptibles d'être appelées lors de cette exécution. Cette table est une donnée "mixte" de SYST en ce sens qu'elle est misę à jour par SYST à l'exécution d'un traitement immédiat.

Elle est construite une fois pour toute dans le cas d'une compilation (cas particulier de traitement immédiat) ou dans le cas d'un traitement immédiat mis en bibliothèque. Elle est créée à partir de la zone référence du programme ATF par consultation de la table CATALOGUE.

STRUCTURE DE LA TABLE TI

Elle occupe une zone, dont l'adresse origine sera appelée TI, comprise entre I et 2 pages et située dans la "zone fixe" de la mémoire disponible. La zone des données qui la suit en mémoire commencera au début de la page suivante.

A chaque procédure susceptible d'être appelée correspond un enregistrement de la table. Cet enregistrement est composé de :

I. numéro (NO) de la procédure dans CATALOGUE. (I/2 mot)

2. Adresse mémoire centrale (AMC) de la première page de la traduction de la procédure si elle s'y trouve (1/2 mot) sinon zéro. Cette adresse est exprimée en pages.

3. Adresse disque (AD) de la première page de la copie de la traduction dans la bibliothèque. Cette adresse, en nombre de pages, est celle obtenue à partir de l'enregistrement dans la table CATALOGUE, dont le rang dans cette dernière est le numéro de la procédure (I/2 mot).

4. Nombre de pages (NP) occupées par la traduction de la procédure. Ce nombre est obtenue comme l'adresse disque à partir de CATALOGUE (I/4 mot).

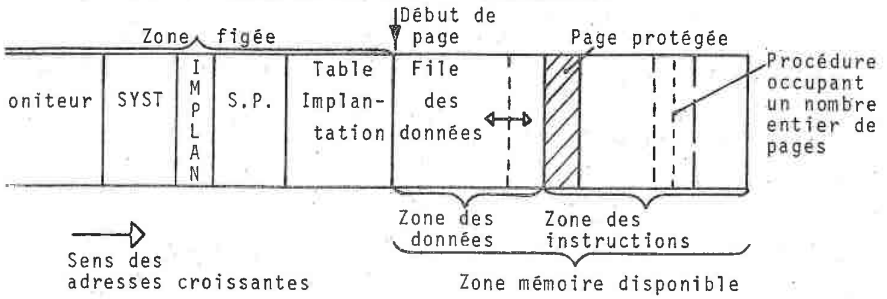
5. Nombre d'appels non terminés (NA)
On a vu que la table d'occupation contient pour chaque procédure en mémoire centrale le nombre d'appels non terminés de cette procédure. Ce nombre intervient dans le critère de choix des pages dans lesquelles on chargera une nouvelle procédure.
Lorsqu'une procédure est éliminée de la mémoire centrale, il est nécessaire de conserver ce nombre. On prévoit donc un emplacement dans la table d'implantation pour y ranger ces nombres.
Lors d'une première - ou d'une nouvelle - implantation d'une procédure, la valeur NA de la table TI, augmentée de 1, est chargée dans la table d'occupation. Tant que cette procédure reste en mémoire centrale, c'est dans la table d'occupation que NA est mis à jour. Si cette procédure vient à être éliminée de la mémoire centrale, NA est alors recopié de la table TO dans la table TI.

Pour un traitement immédiat et pour certaines procédures du compilateur qui sont sollicitées plus souvent que d'autres il est préférable que la valeur initiale de leur nombre d'appels non terminés soit positive, ce qui permet de considérer ces procédures comme "presque" résidentes en mémoire centrale - ("presque" en ce sens qu'il peut quand même, afin de poursuivre le déroulement d'une exécution, être nécessaire de les éliminer de la mémoire centrale).

Le premier enregistrement de la table TI contient le nombre d'enregistrements qui la constituent.

NO	AMC	AD	NP	NA
n°	Adresse M.C.	Adresse disque	Nb. de pages	Nb. d'appels
I/2 mot	I/2 mot	I/2 mot	I/4 mot	I/4 mot

Organisation générale de la mémoire centrale



I. ZONE FIGEE

La zone figée comprend :

- Le système moniteur sous contrôle duquel s'effectue le travail.
- Les programmes SYST et IMPLAN, seuls connus du moniteur.
- La zone S.P. qui contient des sous-programmes standard utilisés par les procédures.
- La table d'implantation qui est une donnée de SYST.

II. ZONE MEMOIRE DISPONIBLE

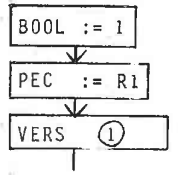
Elle est partagée en deux zones dont la frontière commune est variable pendant l'exécution.

- Zone des données contenant la file des données
- Zone des instructions contenant certaines procédures déjà appelées.

ORGANIGRAMMES DE SYST

Organigramme à partir du point d'entrée APPEL

APPEL :



Lors d'un appel d'une procédure, son numéro est chargé par la procédure appelante dans le registre R1, registre de la machine LMU et de la IO 070.

BOOL est une mémoire permettant d'aiguiller l'exécution de SYST lors du calcul de l'adresse de reprise.

PEC est une mémoire contenant le numéro de la procédure en cours.

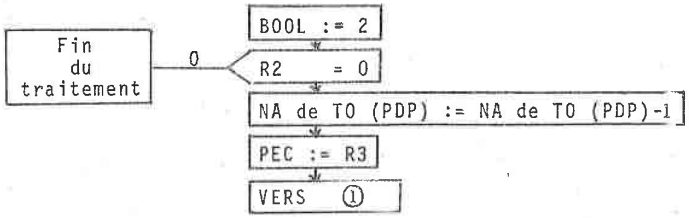
Organigramme à partir du point d'entrée RETOUR

Lors d'un retour à une procédure, la procédure en cours charge - le registre LMU-IO 070, R3, du numéro de la procédure dans laquelle il faut poursuivre l'exécution - le registre R2, de l'adresse relative de retour dans cette procédure, avant d'effectuer un branchement à RETOUR.(cf. ch.III - Retour à une procédure)

* Le contenu de R2, nul indique une fin d'exécution. La procédure appelante ayant un appel terminé et sa traduction étant en mémoire centrale, SYST met à jour la table d'occupation.

** La mémoire PDP pointe vers l'enregistrement de la table d'occupation correspondant à la dernière page de la procédure en cours.

RETOUR :



Remarque :

Il n'est pas nécessaire d'avoir le nombre d'appels non terminés d'une procédure, dans tous les enregistrements de la table d'occupation TO correspondant aux pages occupées par la traduction de la procédure, celui correspondant à la dernière page suffit.

C.2

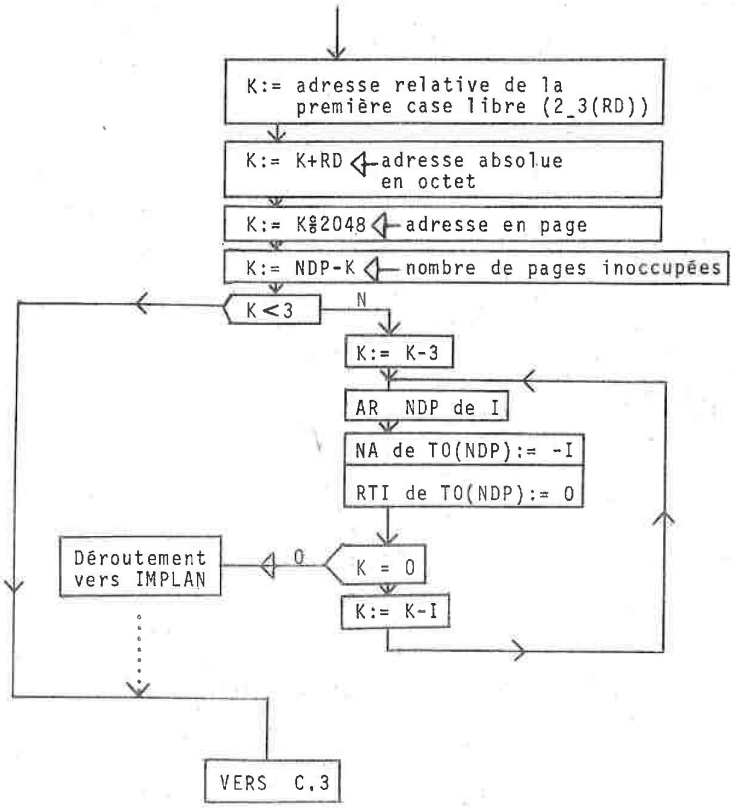
On a vu (cf. introduction) que la taille de la file des données varie à l'exécution. Il est donc nécessaire de récupérer la place qui pourrait être libre entre la file des données et la zone des instructions. On a fixé à 2 le nombre de pages ainsi libérables à partir duquel on fait effectivement une diminution de la zone des données au profit de la zone des instructions.

La différence entre le numéro de la première page de la zone d'instructions (contenu dans NDP) et celui de la page contenant le premier octet libre suivant la file des données, donne le nombre de pages inoccupées plus I.

L'adresse, relativement au contenu du registre RD, de cet octet est située dans la file des données de la procédure en cours, sur les deux octets rangés aux adresses 2 et 3. (Rappelons que RD est le registre LMU-IO 070, contenant l'adresse en octet du début de la file des données de la procédure en cours).

Dans le cas où le nombre de page inoccupées est ≥ 3, la zone donnée est compressée de telle manière qu'une marge de 2 pages est laissée à la file des données ; la mémoire NDP est modifiée ; les verrous de la nouvelle page protégée et de l'ancienne sont mis respectivement à OI et OO par un appel du programme IMPLAN. Cet appel s'effectue par un déroutement pour tentative d'écriture en zone protégée.

ORGANIGRAMME DE C.2



C.3 Recherche de place en mémoire pour implantation

La recherche de place pour l'implantation d'une procédure à exécuter s'effectue en consultant la partie de la table d'occupation correspondant à la zone des instructions dans le sens des adresses de pages décroissantes, car le remplissage de la zone des instructions s'effectue par le "bas". On espère ainsi garder le plus longtemps possible "toute liberté" à la zone des données sans avoir lors d'une extension, à éliminer des traductions déjà en mémoire. Le pointeur de début de recherche, PDR, contient le numéro de la page précédant la dernière implantation, ce qui assure une continuité dans le remplissage de la zone instruction tant qu'il reste des pages libres en partie haute.

A quel critère doit obéir la place choisie pour l'implantation ? Divers critères sont possibles, voici celui qui a été retenu :

Critère de choix de place

Appelons, bloc d'implantation de la procédure à exécuter toute partie de la zone instruction satisfaisant à :

1. sa longueur en page est égale à la taille de la traduction de la procédure.
2. Elle est connexe.
3. La page de numéro le plus haut correspond soit à la page de numéro le plus élevé de l'implantation d'une procédure, soit à une page libre.
   L'existence d'au moins un bloc est nécessaire, sinon il y a blocage et impossibilité d'implanter.
   Appelons taux d'un bloc d'implantation la somme des nombres d'appels non terminés des différentes procédures occupant tout - ou en partie - le bloc ; une

page libre est considérée comme occupée par une procédure de nombre d'appels non terminés égal à -1. Le taux est calculé directement à partir de la table d'occupation. Alors, rechercher la place suffisante pour l'implantation d'une procédure, revient à trouver le premier bloc d'implantation de taux minimum.

AVANTAGES DU CRITERE

. Le critère tient compte du nombre d'appels non terminés d'une procédure. Pour une même occupation en zone instruction, il est plus logique d'éliminer la traduction et la procédure qui a le moins d'appels non terminés : plus le nombre d'appels non terminés est grand, plus on aura besoin de cette procédure dans un avenir proche.

. Une procédure dont tous les appels sont terminés (=0) peut voir sa traduction rester en mémoire, s'il reste d'autres pages libres, et être ainsi utilisée dans de nouveaux appels.

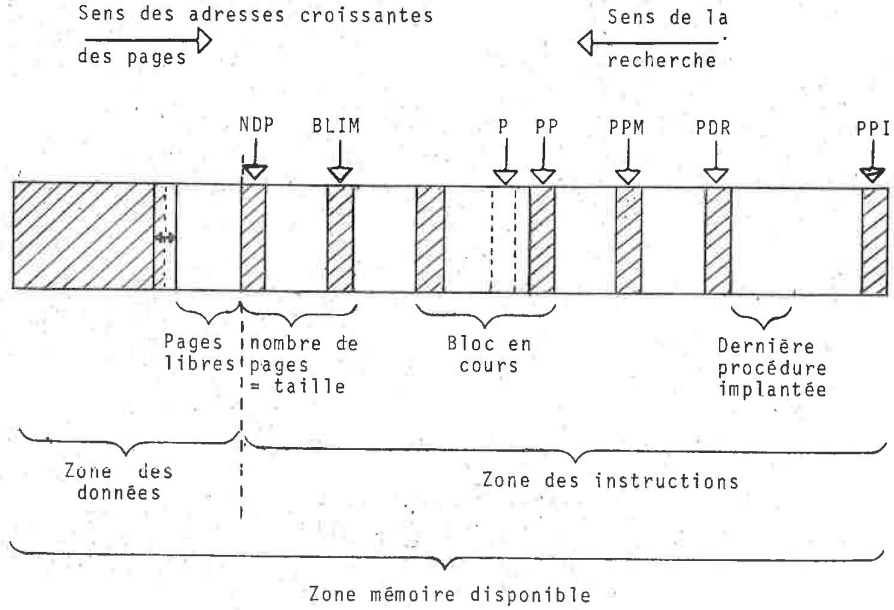
. Une procédure de grande taille, ayant un nombre d'appel non terminés faible, se voit fortement pénalisée, car sa contribution au calcul d'un taux sera faible. Le critère tient compte de l'encombrement des procédures.

. Le calcul d'un taux n'étant pas fait à partir des différentes pages mais à partir des différentes procédures occupant le bloc, il s'ensuit qu'un bloc contenant une seule procédure ayant un nombre d'appels non terminés =n, sera préféré à un autre ayant plusieurs procédures de nombre d'appels =n.

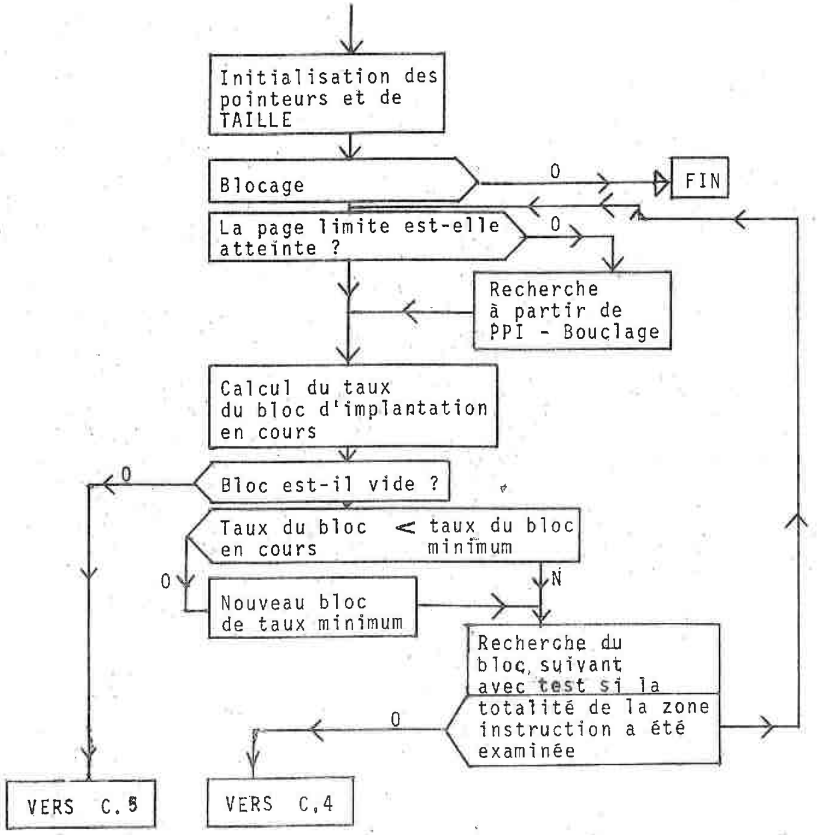
LES POINTEURS DE LA TABLE D'OCCUPATION :

Leurs valeurs sont à la fois des numéros de page et des rangs dans TO.

PPI → pointe vers la page de plus haut numéro
PDR → pointeur de début de recherche
PPM → pointe vers la dernière page du premier bloc de taux minimum rencontré
PP → pointe vers la dernière page du bloc en cours
P → pointe vers la page courante
BLIM → pointe vers la dernière page du bloc limite de longueur = nombre de page de la traduction
NDP → pointe vers la page protégée suivant la zone des données

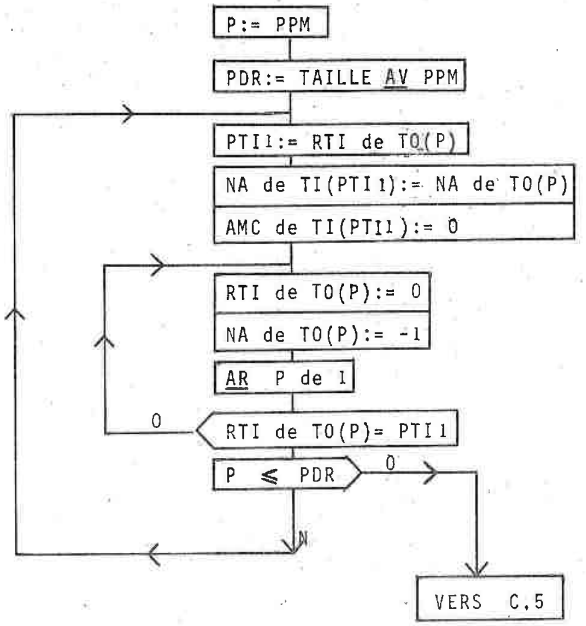


ORDINOGRAMME DE C.3



TAILLE	contient le nombre de pages de la traduction à implanter
X	variable auxiliaire
TMIN	contient le taux minimum
TEC	contient le taux en cours

ORGANIGRAMME DE C.4

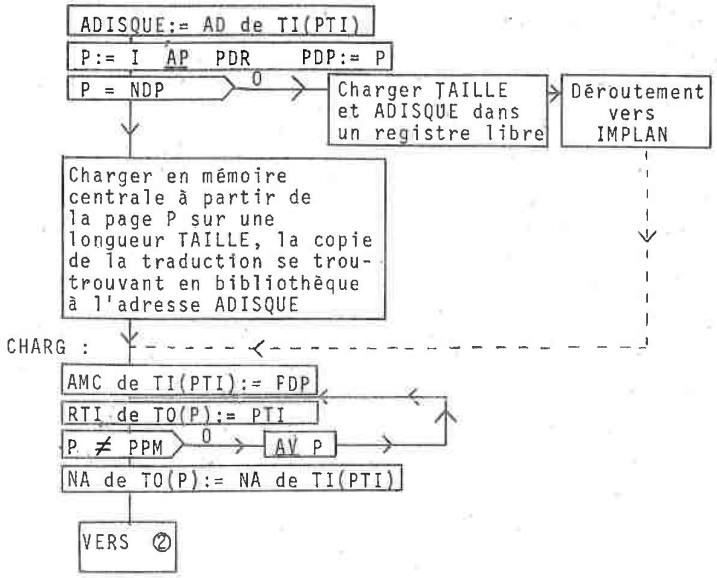


C.5.

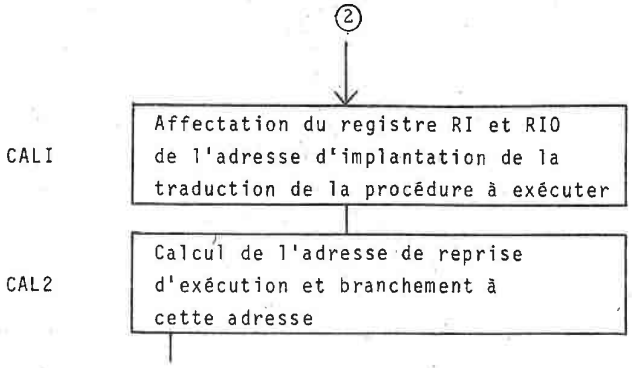
Le chargement s'effectue en général par SYST (sauf dans le cas où la page protégée va contenir le début de la traduction de la procédure ; il est alors nécessaire de faire appel à IMPLAN possédant une clé "passe-partout" qui lui permet d'écrire dans cette page). Une mise à jour des tables est effectuée :

- dans la table d'implantation TI, la partie adresse mémoire centrale AMC, de l'enregistrement correspondant à la procédure est affectée de son adresse (en page) d'implantation
- dans la table d'occupation la partie rang dans la table d'implantation RTI des enregistrements correspondant aux pages chargées, est affectée de la valeur du pointeur PTI. La partie nombre d'appels non terminés NA de l'enregistrement correspondant à la dernière page chargée est affectée de la valeur effective du nombre d'appels NA se trouvant en table d'implantation.

ORGANIGRAMME DE C.5



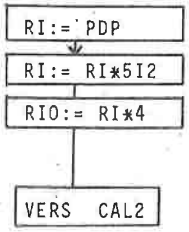
Organigramme à partir de ②



CALI

Les registres RI et RIO contiennent respectivement l'adresse d'implantation de la traduction de la procédure, en mots respectivement en octets.

ORGANIGRAMME DE CALI



CAL2

Suivant que SYST est appelé ou activé pour l'appel à une procédure ou pour le retour à une procédure ou pour une extension de la zone donnée, trois calculs d'adresse de reprise sont à envisager :

I. Lors d'un appel à une procédure, le nombre d'appels non terminés de la procédure est incrémenté de un, et l'adresse de reprise est alors l'adresse d'implantation de la traduction de la procédure.

2. Lors d'un retour à une procédure, le nombre d'appels non terminés a déjà été décrémenté de un ; l'adresse de reprise est alors la somme de l'adresse d'implantation de la traduction et de l'adresse relative de retour dans la traduction. Cette dernière adresse a été chargée dans le registre R2.

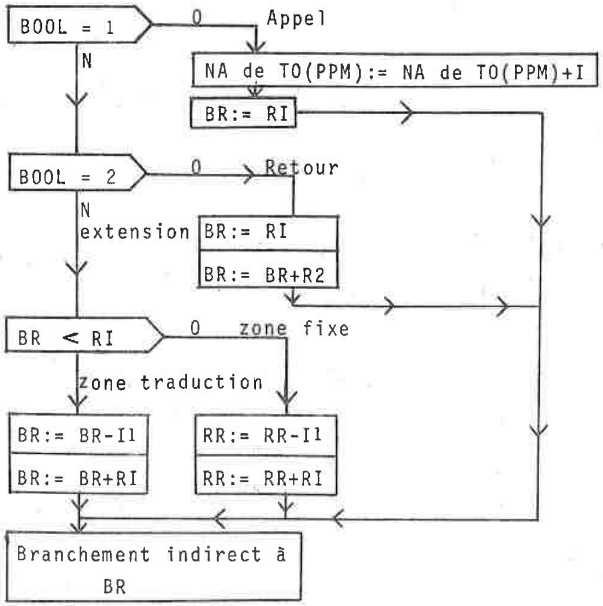
3. Lors d'une extension de zone des données, deux cas sont à envisager :

- la traduction de la procédure est restée au même endroit alors l'adresse de reprise est celle de l'instruction qui a provoqué le déroutement vers IMPLAN.

- la traduction, se trouvant avant le déroutement en zone protégée, n'est plus à la même place :
. si l'instruction responsable du déroutement est dans la traduction, l'adresse de reprise est à calculer en fonction de la nouvelle implantation de la traduction ;
. si l'instruction responsable du déroutement est dans un sous-programme situé en zone fixe SP, l'adresse de reprise est celle de l'instruction qui a provoqué

le déroutement, mais l'adresse de retour (chargée dans le registre RR) du sous-programme vers la traduction est à modifier en fonction de la nouvelle implantation de la traduction.

ORGANIGRAMME DE CAL2



I1 contient la valeur de RI avant sa modification

BR contient - l'adresse de l'instruction responsable du déroutement pour extension de la zone des données
- l'adresse de reprise

RR contient l'adresse de retour d'un sous-programme de la zone SP vers la traduction de la procédure en cours.

Organigramme à partir de EXZD

Lors d'une extension de la zone des données, IMPLAN rend le contrôle au programme SYST grâce à une instruction de chargement de double-mot d'état de programme (LPSD) ; ce double-mot contient comme adresse de la prochaine instruction l'adresse de l'étiquette EXZD. L'adresse de l'instruction responsable du déroutement est transmise dans un registre (B).

SYST charge cette adresse dans la mémoire BR. Par consultation de la table d'occupation, SYST détermine si la page rendue à la zone donnée était vide ou contenait une traduction de procédure. Dans le premier cas, il y a un branchement à l'instruction qui avait provoqué le déroutement. Dans le deuxième cas, la traduction de la procédure est vidée par une mise à jour des tables d'occupation et d'implantation suivant que la procédure en cours voit sa traduction rester en mémoire ou pas, un branchement est effectué vers l'adresse de l'instruction responsable du déroutement ou vers l'étiquette RCHARG pour une nouvelle recherche de place en mémoire.

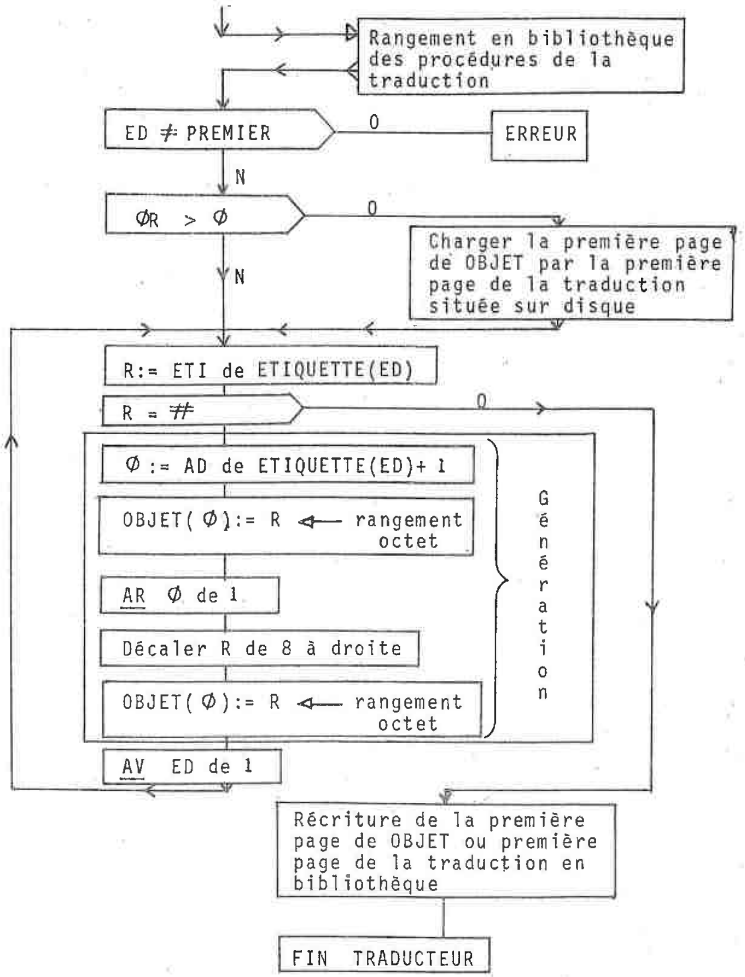
par. ⑤

Une fois traduite la zone des instructions, par la partie ④ (cf. plus loin), la partie ⑤ est exécutée.

Ce paragraphe range la ou le reste de la traduction située dans la zone OBJET, dans la bibliothèque des traductions de procédures située sur disque. Puis il effectue la codification des étiquettes internes dans le programme objet. Pour cela, il dispose de la première page de la traduction, située ou chargée dans la première page de la zone OBJET. Il dispose d'autre part de la table ETIQUETTE. Cette table contient pour chaque étiquette sa valeur d'adresse (relative) en mots (ETI) et son adresse relative de début d'implantation dans la traduction, adresse exprimée en octets (AD). La valeur d'adresse d'une étiquette est codifiée sur 2 octets.

L'exécution du paragraphe ⑤ , permet de tester si toutes les étiquettes ont été satisfaites puis de garnir la zone des étiquettes dans la traduction. La première page de la zone OBJET est ensuite réécrite en bibliothèque.

Organigramme



Passage de SYST à IMPLAN

Toute exécution d'une opération interdite provoque l'exécution de l'instruction située en adresse mémoire X'40', qui est une instruction de changement de double-mot d'état de programme (XPSD). Dans le cas d'une violation de protection mémoire, le bit 3 du nouvel état de programme est égal à I, et la partie adresse instruction du nouvel état de programme est augmentée de I.

Il est alors nécessaire que cette adresse désigne une instruction permettant de donner le contrôle au programme IMPLAN.

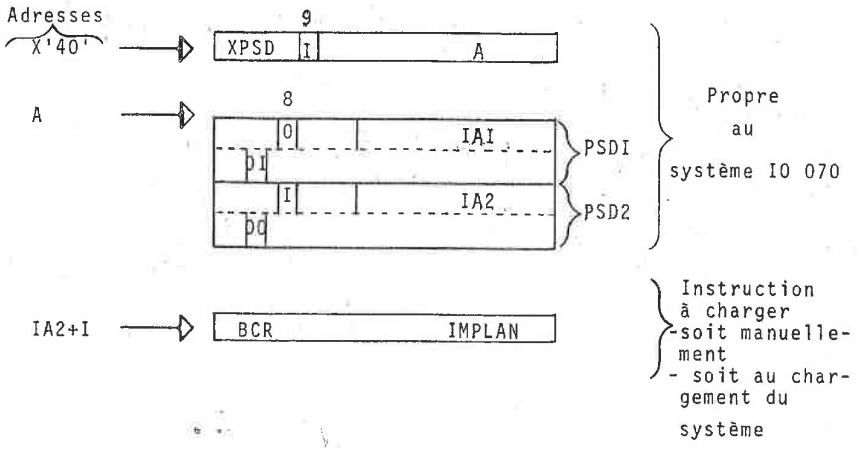
Elle peut être :

- soit un branchement à la première instruction de IMPLAN, dans le cas où le double-mot d'état de programme chargé, répond aux exigences du programme IMPLAN (mode maître, clé 00..).

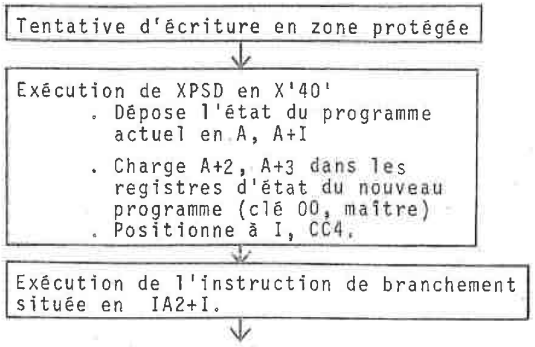
- soit un chargement de double-mot d'état avec comme partie adresse instruction, l'adresse de la première instruction de IMPLAN.

Nous avons choisi la première solution. Cette adresse sera fixée lors de la génération du système utilisé, à l'aide de la directive GEN.

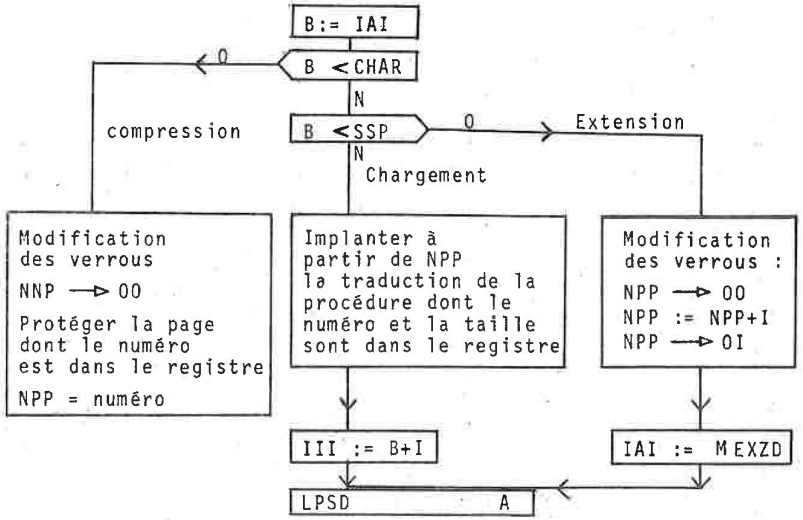
SCHEMA DES OPERATIONS



ORGANIGRAMME



ORGANIGRAMME DE IMPLAN



Mémoires

B — est un registre IO 070 qui contient l'adresse de l'instruction responsable du déroutement vers IMPLAN

SSP, CHAR, MEXZD — sont des mémoires contenant les adresses respectives des débuts des zones :

- zone des sous-programmes
- zone etiquetée dans C.5 de SYST par CHARG.
- zone etiquetée dans SYST par EXZD.

IAI — est la partie adresse du double-mot d'état de programme rangé à l'adresse A lors de l'exécution de l'instructio XPSD se trouvant en X'40'.

NPP — Mémoire de IMPLAN contenant le numéro de la page protégée.

CHAPITRE VI

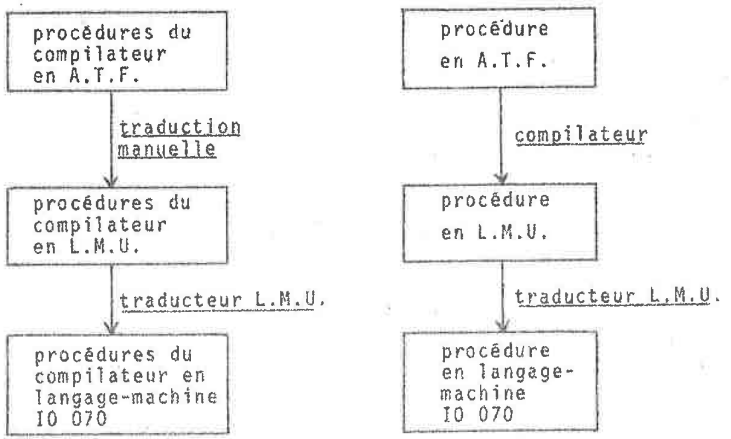
LE COMPILATEUR A.T.F. DE BASE

I. CONSTITUTION DU COMPILATEUR

Le compilateur A.T.F. permet le passage d'un programme A.T.F. en un programme L.M.U., sémantiquement équivalent.

Le compilateur a été d'abord écrit dans le langage A.T.F. lui-même, sous la forme de procédures complètes - Une exécution du compilateur se fait par l'appel effectif de la procédure "procédure complète atf". Le compilateur A.T.F. a ensuite été écrit en L.M.U., en traduisant manuellement l'écriture A.T.F.

On obtient alors le schéma suivant :



Remarquons que seul le passage de L.M.U. au langage-machine IO 070 dépend du matériel utilisé.

Les procédures ainsi traduites sont placées sur disque, dans le fichier bibliothèque des procédures.

La première partie de la table CATALOGUE , répertoriant les procédures du compilateur, est construite une fois pour toute :

les procédures sont classées et numérotées 0 I 2 ..... - le numéro de la procédure "procédure complète atf" correspond à 0, etc... - Les numéros des procédures correspondent aussi aux rangs de leurs enregistrements dans la table CATALOGUE (un enregistrement par procédure). On connait alors les valeurs prises par les différents éléments de chaque enregistrement (les éléments : identificateur, adresse disque, volume et référence).

Exécution du compilateur A.T.F.

Le lancement de la compilation A.T.F.-L,M.U. est effectuée par l'appel de la procédure "procédure complète atf". Cette procédure possédant des paramètres, il est nécessaire de leur substituer les paramètres effectifs c'est à dire de transmettre les caractéristiques des paramètres effectifs à la file des données de la procédure.

Les paramètres effectifs:

. La file texte qui représente le programme A,T.F. chargé en mémoire - l'article constitue un caractère,
. La file identif qui est une file de travail dans laquelle vient se placer à la compilation les identificateurs du programme A.T.F. L'article est un caractère.
. L'index id de la file identif (2 octets).
. La file caract qui est une file de travail.
. L'index c de la file caract (2 octets),
. Index tr de la file traduction (2 octets),
. La file traduction dont l'article est un caractère; elle contient en fin de compilation le programme L.M.U.

La place pour ces paramètres est réservée dans la zone des données de SYST.

Constitution de la file des données de "procédure complète atf"

Les huit premières cases

Adresse	Longueur	Contenu
0+CD	2	0
2+CD	2	adresse de la première case libre après les caractéristiques des paramètres effectifs.
4+CD	2	0 Cette adresse de retour nulle permet de tester la fin de compilation.
6+CD	2	0

Les caractéristiques des paramètres

Nous nous bornerons aux caractéristiques de la file texte.

Adresse	Longueur	Contenu	
8+CD	I	'f'	file
9+CD	2	I	nombre d'entrée
II+CD	2	I	nombre de termes par élément
I3+CD	I	'C'	file complète
I4+CD	2	i=0	borne inférieure
I6+CD	2	s	borne supérieure
I8+CD	2	p=0	pas
20+CD	2	ℓ	longueur de la file $\ell=(s-i)p$
22+CD	I	'C'	élément(caractère)
23+CD	2	adresse relative du premier élément de texte	
25+CD	I	';'	fin des paramètres-données.

Remargue

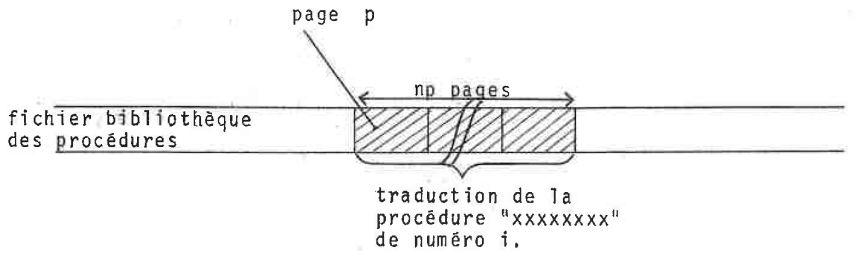
L'A.T.F. de base ne possède pas d'opérations d'entrée-sortie. En conséquence, la sortie du programme L.M.U. se fait par l'appel de la procédure "sortie lmu" écrite manuellement en langage machine IO 070 et fonctionnant comme une autre procédure du compilateur. Son appel figure dans la procédure "lmu" (n° 58).

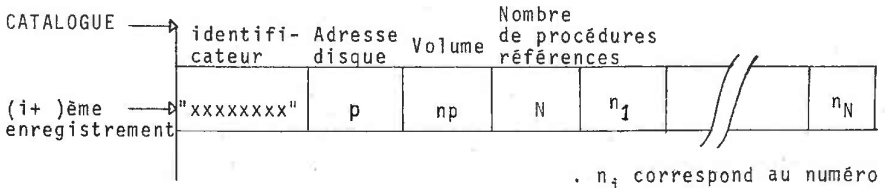
L'exécution d'un traitement immédiat, en particulier celle du compilateur se fait sous contrôle des programmes SYST et IMPLAN.

SYST dispose, entre autre, de la table d'implantation TI qui permet d'enchainer les procédures susceptibles d'être appelées. Dans le cas d'une exécution du compilateur, toutes les procédures le constituant, sont susceptibles d'être appelées. Pour toute compilation, la table TI est initialisée à partir de la table constante C:TI.

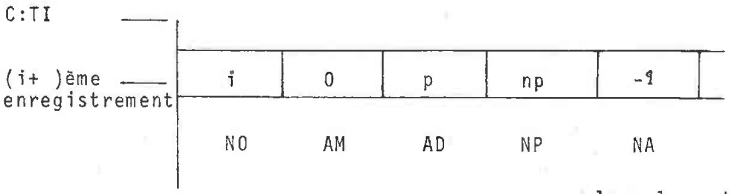
La table C:TI est constituée d'enregistrements ayant la même structure que ceux de la table TI. A la $(i+1)^{ème}$ procédure du compilateur (de numéro i) correspond le $i^{ème}$ enregistrement de C:TI.

Schéma de correspondance des tables CATALOGUE, C:TI et du fichier des procédures.





. n_j correspond au numéro de la $j^{ème}$ procédure référencée.



. la valeur du nombre d'appel NA est rendue ⩾ 0 pour les procédures que l'on veut "presque résidentes".

Après l'initialisation :

- de la file de données de la procédure "procédure complète atf"
- du registre D contenant l'adresse du premier octet de de la file des données
- du registre R1 qui contient le numéro (ici 0) de la procédure à exécuter.

Alors le lancement proprement dit de la compilation s'effectue par un appel au programme SYST vers son étiquette APPEL.

CONCLUSION GENERALE

Nous traiterons ici :

. les tâches qu'il reste à faire pour terminer le travail.
. les points obscurs dans A.T.F. et L.M.U. qui devront être élucider.
. une appréciation sur la traduction de A.T.F. via L.M.U.

I. TACHES A FAIRE

En dehors du travail d'écriture du traducteur L.M.U. et des programmes SYST et IMPLAN, il reste à concevoir :

- un programme d'enchainement chargé de coordonner les différentes tâches d'une compilation, d'une procédure A.T.F. ou d'un traitement immédiat A.T.F. et d'autre part, chargé de la gestion de la bibliothèque des procédures (suppression de procédures).
- un { système / traducteur } permettant de traiter un A.T.F., muni d'entrée-sortie.

I.1. Programme d'enchainement - ENCHAINE

Nous décrirons ici le programme d'enchainement ENCHAINE dans ses grandes lignes.

Différents travaux sont commandés à ENCHAINE :

- la compilation d'une procédure et sa mise en bibliothèque.
- la compilation d'un traitement immédiat et son exécution.
- suppression de procédures en bibliothèque.

L'analyse de la commande (sous forme de carte de commande) indique au programme d'enchainement ENCHAINE, parmi les 3 travaux possibles, lequel lui est demandé.

I.1.1. Compilation d'une procédure

Chargement

Le programme d'enchainement charge le programme source A.T.F., en mémoire centrale, pour constituer la file "texte" (donnée de la procédure "procédure complète atf" du compilateur).

Mise à jour de la table CATALOGUE

La recherche du nom de la procédure (obtenu à partir de la file "texte") dans la table CATALOGUE permet de savoir si la procédure a été référencée dans des procédures déjà compilées ; auquel cas un enregistrement de CATALOGUE lui est déjà réservé. Dans l'autre cas un enregistrement de la table lui est attribué. la partie identificateur est garnie du nom de la procédure.

La partie référence de l'enregistrement est alors garnie des numéros des procédures référencées. Le numéro d'une procédure référencée est obtenu à l'aide du nom de la procédure situé en zone référence dans la file "texte". Une consultation de CATALOGUE détermine alors l'enregistrement contenant dans sa partie identificateur le nom de la procédure référencée. Le rang de l'enregistrement dans la table est alors le numéro de la procédure. Dans le cas où aucun enregistrement ne contient l'identificateur, c'est que la procédure en cause n'a pas été encore compilée. Un enregistrement de CATALOGUE lui est alors réservé ; sa partie identificateur est garnie du nom de la procédure. Son numéro est alors le rang dans CATALOGUE du nouvel enregistrement.

Compilation

La tâche suivante de ENCHAINE est de mettre le programme SYST-IMPLAN en mémoire, d'initialiser la table d'implantation TI, le registre D, le registre R1 et la file des données de la procédure "procédure complète atf".

ENCHAINE donne le contrôle à SYST, à l'étiquette APPEL.

La compilation A.T.F, - L.M.U. s'effectue alors sous contrôle de SYST par l'exécution de la procédure "procédure complète atf" qui appelle les différentes procédures du compilateur.

Traduction

SYST rend le contrôle à ENCHAINE (lorsqu'il a testé une adresse de retour à une procédure qui était nulle). ENCHAINE appelle le traducteur L.M.U. en mémoire. Celui-ci, après avoir pris le contrôle, effectue la traduction du programme L.M.U. Le programme objet est placé en bibliothèque des procédures.

ENCHAINE reprend le contrôle, termine le garnissage de l'enregistrement de CATALOGUE.

. La partie adresse disque reçoit l'adresse (page) de la première page de la procédure placée sur disque.

. La partie volume reçoit le nombre de pages occupées par la procédure.

I.1.2. Compilation et exécution d'un traitement immédiat

Les phases chargement, compilation et traduction, sont les mêmes que précédemment. La phase suivante est la construction de la table d'implantation TI.

Construction de TI

A partir des numéros des procédures référencées, le programme ENCHAINE par consultation de CATALOGUE détermine les numé-

ros des procédures que ces dernières référencent et ainsi de suite. L'ensemble des numéros de procédures, susceptibles d'être utilisés à l'exécution (numéro du traitement immédiat compris) est alors rangé par ordre croissant dans la table TI.

Exécution

Le programme SYST-IMPLAN est chargé en mémoire. Il y a initialisation de :

. file des données du traitement immédiat : les 8 premiers octets sont mis à zéro sauf les 2 octets d'adresse relative 2-3 qui contiennent la valeur 8 ; valeur qui correspond à la première case libre de la file des données,

. registre R1 qui contient le numéro du traitement immédiat.

. registre D qui contient l'adresse absolue de la première case de la file des données. ENCHAINE donne le contrôle à SYST vers l'étiquette APPEL.

La fin d'exécution détermine la fin du travail.

I.2.3. Suppression de procédures en bibliothèque

ENCHAINE charge en mémoire le nom de la procédure à supprimer. Une recherche dans CATALOGUE du nom de cette procédure détermine l'enregistrement qui le contient et aussi le numéro de la procédure. La suppression d'une procédure empêche l'exécution de procédures qui l'appellent. Donc avant de supprimer une procédure, on effectue une recherche dans CATALOGUE des enregistrements contenant, en partie référence, le numéro de la procédure à supprimer. Les procédures correspondantes sont aussi à supprimer. ENCHAINE fournit une liste de procédures ainsi obtenues et ce n'est qu'après un message de l'opérateur qu'elles sont supprimées.

II. UN SYSTEME POUR UN A.T.F. COMPLET

Le compilateur A.T.F. ⟶ L.M.U., dont nous avons connaissance, doit servir "d'amorce" pour une extension ; une auto-compilation prouvant son bon fonctionnement. Les procédures du compilateur ne font appel ni aux réels ni aux mots de longueur supérieure à un et surtout n'utilise aucune instruction d'entrée-sortie (La procédure permettant la sortie du programme objet LMU, doit être écrite dans le langage machine sur lequel s'implante ATF).

Les problèmes des entrées-sorties sont les plus importants :

1) Description des objets placés sur support externe, au niveau d'A.T.F.

2) - Traduction des opérations d'échange en L.M.U.
   - Traduction de ces opérations en langage-machine.

a) Description des objets placés sur support externe

En général, un travail demandé à un ordinateur, consiste - en une suite de cartes de commande qui permettent, entre autre, de donner au "système" sous lequel on travaille, les caractéristiques des fichiers ou autres objets externes - puis vient le programme, proprement dit écrit dans un langage déterminé, précédé par exemple d'une carte de commande de compilation. La liaison entre les fichiers décrits dans les cartes et ceux correspondants dans le programme, se fait à l'aide de tables gérées par le système.

Jusqu'alors en A.T.F. nous avons considéré un traitement immédiat comme une procédure sans paramètre traduite par le compilateur A.T.F. Mais utiliser des entrées-sorties, suppose un traitement des descriptions de files externes à la mémoire centrale, comme le ferait le système, précédemment.

Il est à remarquer que les descriptions de files externes en A.T.F. sont insuffisantes : il ne suffit pas de dire

que la file est sur le disque mais encore est-ce une file à rangement consécutif, aléatoire, séquentiel indexé, le bloc physique est-il constitué de plusieurs articles ? Tous ces renseignements ne figurent pas.

Nous avons considéré le traitement immédiat, mais il se peut aussi qu'une procédure indépendante utilise elle-même, comme variables internes, des files placées sur support externe. Un appel d'une telle procédure nécessitera donc une allocation dynamique de mémoire externe ce qui n'est pas permis pour beaucoup de systèmes actuels. Faut-il que cette réservation se fasse avant l'exécution - ce serait nier l'esprit d'A.T.F.

b) Traduction des opérations d'échange

L.M.U., initialement, ne contenait aucune instruction d'entrée-sortie. C'est dans cette optique que la traduction L.M.U. a été conçu.

Puis L.M.U. s'est vu adjoindre des opérations d'échange: celles d'une machine théorique L.M.U.,machine qui est loin de la réalité ; déjà aux niveaux des autres instructions, nous sommes éloignés d'une machine comme le 10 070. En supposant qu'un compilateur A.T.F. fournisse des instructions d'échange en L.M.U., alors traduire ou interpréter de telles instructions serait simuler une machine peu performante alors qu'on dispose d'un système "hardware-software" d'entrées-sorties très performant sur le 10 070. On s'éloigne ainsi du but d'efficacité, qualité qui était a priori inhérente à l'A.T.F.

Il est donc nécessaire de prévoir un moyen permettant de traduire des ordres d'entrées-sorties A.T.F. ou bien un des appels de procédures écrites en langage-machine et fonctionnant

comme toute autre procédure A.T.F., ou bien en des appels aux procédures d'entrées-sorties du système 10 070 ; celles-ci seraient de toute manière utilisées dans le premier cas.

Conclusion

Nous avons aborder le problème des entrées-sorties (traduction des descriptions de fichier externe en cartes de commande d'un système 10 070 disponible - traduction des opérations d'échange en appel de procédures systèmes).

En réalité il serait beaucoup plus intéressant de disposer d'un système A.T.F. permettant :

- Une compilation à la fois des procédures et des traitements immédiats. Pour cela une unification du langage de commande du système et du langage de programmation serait faite ; A.T.F. l'a presque atteinte. Il reste à enrichir le langage de commande notamment dans la description des files placées sur support externe ; une commande de désallocation dynamique de mémoire serait utile afin d'attribuer de nouvelles "ressources" à la "tâche" suivante dans un traitement immédiat.

Le système A.T.F. disposerait de toute une bibliothèque de procédures A.T.F. indépendante , en particulier celles nécessaires à la compilation, celles nécessaires aux entrées-sorties.

Sur la base du programme SYST, le système permettrait l'allocation et la désallocation dynamique de mémoire interne et externe et des "ressources" en général (passage d'une procédure à une autre).

Ajoutons que le système pourrait prévoir la multi-programmation (chaque travail aurait sa "file de données" ; les procédures étant indépendantes, chacune peut opérer sur l'une ou l'autre des "files de données".

Les points obscurs

Le point obscur le plus important est celui de la compilation des procédures incomplètes. Le compilateur A.T.F. actuel (fait de procédures complètes) ne traduit pas les procédures incomplètes.

Une procédure complète est totalement indépendante - tout programme peut y faire appel.

Une procédure incomplète pi dépend d'une autre procédure p par des variables externes (supposons qu'il n'y en ait qu'une: β) Cela suppose qu'à l'exécution de pi, la file des données de pi contiennent les caractéristiques de β . Par définition, celles-ci n'ont pu être transmises comme celles des paramètres, juste avant l'appel de pi. Donc la compilation doit produire des instructions permettant la construction des caractéristiques de β :

Il s'agit donc de trouver - d'une part la "version actuelle" de la procédure p qui contient β dans sa file de données (p doit être en cours de déroulement) - d'autre part, d'identifier les caractéristiques de β dans la file de données. La première partie se traite relativement facilement en recherchant dans la file globale des données, la première des files de données correspondant à la procédure p (cette définition de la version actuelle n'est d'ailleurs pas pleinement satisfaisante).
La deuxième partie est plus complexe car rien dans la file des données de p ne permet d'identifier le β de pi. Pour rester dans l'esprit d'A.T.F. il est nécessaire de disposer des files "caract" et "identif" qui ont servi lors de la compilation de p ; à partir de l'identificateur β de pi il est alors possible de déterminer relativement les caractéristiques de β en particulier sa valeur. Le fait de pouvoir disposer des files "caract" et "identif" de p oblige que pour toute procédure A.T.F., il en soit ainsi, car a

priori, p n'est pas sensée avoir une variable référencée dans une procédure incomplète.

Une solution plus efficace serait de compiler une procédure incomplète avec la procédure dans laquelle elle référence une ou plusieurs variables ; le lien entre les deux procédures serait plus aisé, la procédure complète gardant toute son indépendance.

Toujours dans les procédures incomplètes on ne sait pourquoi le "corps" de procédure se limite à une seule instruction (d'appel à une procédure).

III. APPRECIATION DE LA TRADUCTION ATF PAR L'INTERMEDIAIRE DE L.M.U.

Le compilateur A.T.F. d'abord écrit en A.T.F. a été traduit en L.M.U. pour une meilleure diffusion ; le L.M.U. se voulant un langage proche de la machine.

Cette technique d'utiliser un langage intermédiaire est souvent utilisé dans les compilateurs actuels avec toutefois ceci : le langage intermédiaire permet une codification simple et compact et il est très proche du langage machine de l'ordinateur.

En ce qui concerne le 10 070, le L.M.U. ne correspond pas à son langage machine : les techniques d'adressage sont totalement différentes, l'adressage du mot est facile en 10 070 mais l'octet (correspondant à la case L.M.U.) l'est déjà beaucoup moins ; travailler sur un groupe d'octets, à cheval sur des mots, n'est plus très aisé. Une instruction L.M.U. est en général à trois adresses alors que celle 10 070 est à une seule adresse.
La structure d'un programme L.M.U. rend compte parfaitement des techniques modernes des appels de procédures (transmission des paramètres, indépendance des instructions et des données, enchaînement de procédures) ; pour cela le langage est très pédagogique.

Mais ces techniques nécessitent une production importante d'instructions. Comme le compilateur A.T.F. de base comporte déjà plus de 80 procédures, il faut s'attendre à un compilateur énorme. Certaines procédures, ayant peu d'instructions mais surtout des appels de procédures, ont un programme objet L.M.U. très important.

. A une exécution lente : le compilateur étant important, il ne peut rester globalement en mémoire centrale.

D'autre part, la traduction A.T.F.-L.M.U. produit un programme L.M.U., très mal codifié : les instructions L.M.U. "écrites" sont des chaînes de caractères de longueur moyenne située entre 20 et 30. Une instruction qui pourrait occuper un mot va en occuper 5. Le programme intermédiaire L.M.U. doit donc pouvoir être sorti de la mémoire puis entrer à nouveau pour traduction en langage-machine. Un compilateur plus performant serait obtenu en traduisant, manuellement les procédures A.T.F. du compilateur dans le langage machine du 10 070. Cela ne résoud pas le problème des autres procédures compilées nécessairement en passant par le L.M.U.

Pour qu'A.T.F. (en particulier A.T.F.-GESTION que nous avons retenu) puisse s'imposer à d'autres langages tels que COBOL, PL1, il est nécessaire que les programmes soient performants. Pour cela, il faut écrire un compilateur qui tienne compte de la machine sur laquelle le langage est à implémenter.

Un compilateur "amorce", performant, écrit en langage machine (sous forme de procédures indépendantes) traitant les entrées-sorties (sans entrée-sortie l'amorce ne serait pas efficace) permettrait l'écriture en A.T.F. du système A.T.F. entrevu précédemment.

ANNEXE

Exemple I.

PROCEDURE ECLATEMENT (TEXTE-DON;;SMOT) ;

Introduction

La procédure ECLATEMENT permet à partir d'un texte constituant un fichier sur carte, de créer un fichier sur bande représentant la suite des mots du texte avec leurs caractéristiques.

Les paramètres

I. ECLATEMENT admet un paramètre donné : TEXTE-DON qui est un fichier sur carte, correspondant au texte.

Un texte est codifié sur trois zones A,B,C. Trois articles consécutifs du fichier correspondent à ces trois zones.
Les articles ont la même structure CTEX1 :

. Les 10 premiers caractères constituent un tableau de renseignements :

- 2 caractères pour l'auteur
- 3 caractères pour le numéro du chapitre
- 3 caractères pour le rang de la phrase dans le chapitre
- 3 caractères pour préciser la zone et le numéro de séquence dans la phrase.

. Les 70 caractères restants, constituent le texte proprement dit. Il est composé dans la zone A d'un nombre entier de mots délimités par des séparateurs ; la partie qui suit le dernier séparateur est constituée d'espace. Les caractères correspondants, des zones B et C, apportent les renseignements sur les mots.

2. ECLATEMENT admet un paramètre résultat : SMOT qui est un fichier sur bande, correspondant à la suite des mots du texte et leurs renseignements.

Les articles sont structurés par la partition BMO :

- un naturel (2) pour la longueur du mot
- trois mots de même longueur variable correspondant aux mots issus respectivement des zones A,B,C.
- trois naturels de longueur 3 correspondant respectivement au chapitre, au rang de la phrase dans le chapitre, au rang du mot dans la phrase.
- un booléen qui vaut : VRAI lorsque le mot est un nom propre, FAUX sinon.
- deux caractères correspondant respectivement au séparateur aval et amont.

Généralités sur l'écriture de la procédure

La lecture d'un article du fichier TEXTE-DON se fait par l'intermédiaire de l'article TEX. Il est à remarquer que TEX n'a pas exactement la même structure CTEX que l'article de TEXTE-DON : certaines parties de l'article de TEXTE-DON sont superflues pour le traitement (auteur, zone et numéro de séquence) il leur correspond RIEN. La file à traiter est tronquée. CTEX et CTEX1 sont des partitions en parallèle. La file TRAV, nécessaire au traitement, possède deux entrées, une pour les 70 caractères du texte, l'autre pour les zones A,B,C.

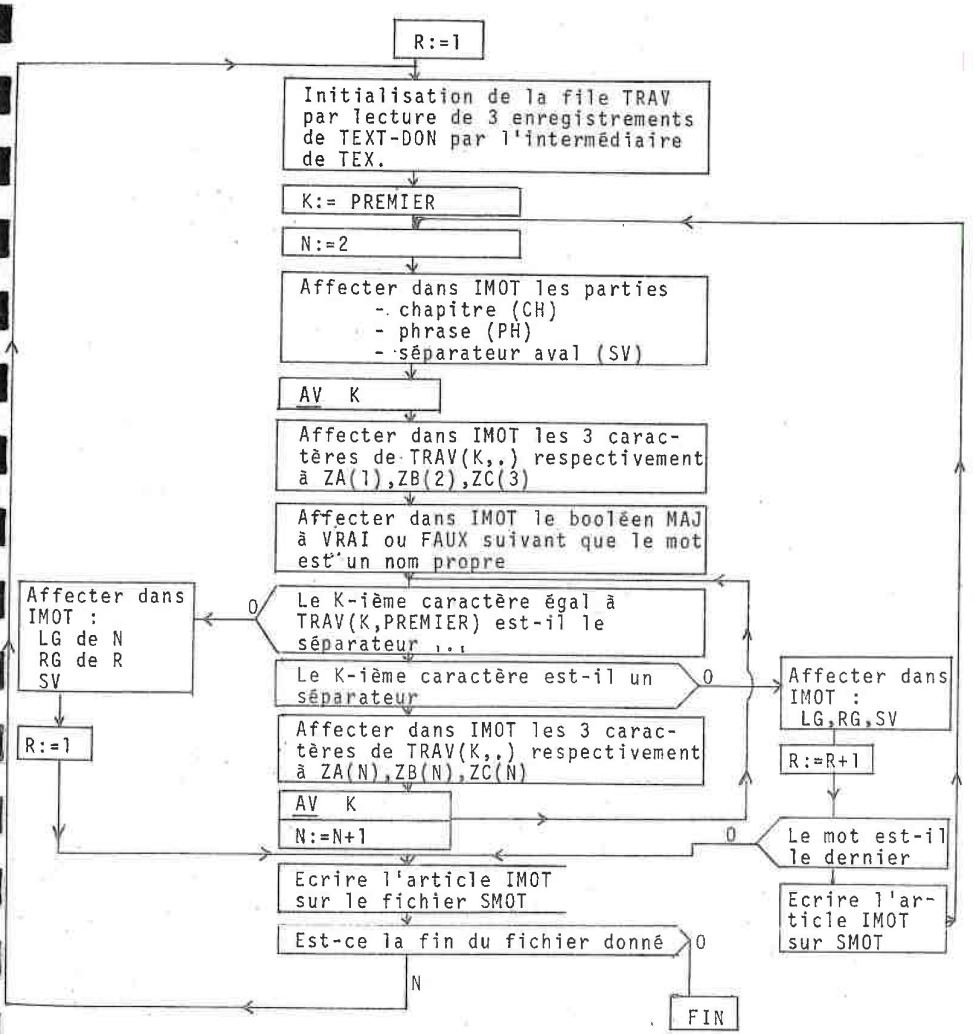
La lecture de 3 enregistrements correspondant aux 3 zones A,B,C du texte donné permet de remplir la file TRAV.

La reconnaissance d'un mot s'effectue caractère par caractère sur la file TRAV, séquentiellement sur la première entrée correspondant au texte, en parallèle sur la deuxième entrée correspondant aux zones.

L'article IMOT - de même structure que l'article de la file SMOT - se remplit durant le traitement et est écrit ensuite dans la file SMOT.

R contient le rang du mot dans la phrase, (information qui n'est pas donnée).

Organigramme



TYPE SPECIFICATION

STATUT	NATURE	NOM	LONGUEUR	PARTITION NOM	PARTITION DANS	E	INDEX NOM	PLACE
DO	FILE	TEXTE-DON		CTEX1			I	CARTE
RE	FILE	SMOT		BMO1			J	BANDE
AU	ARTICLE	TEX		CTEX				
AU	ARTICLE	IMOT		BMO				
AU	FILE	TRAV	70,3	CAR		1	K,K1	
						2	L	
AU	NA	N	2					
	NA	R	2					
AU	INDEX	I						
AU	INDEX	J						
AU	INDEX	K						
AU	INDEX	K1						
AU	INDEX	L						

TYPE PARTITION

PARTITION	NB.FOIS	NATURE	LONGUEUR	NOM	FB	;
BMO		NA	2	LB		
	LB	MOT	1	ZA		
	LB	MOT	1	ZB		
	LG	MOT	1	ZC		
		NA	3	CH		
		NA	3	PH		
		NA	3	RB		
		BO	1	MAJ		
		MOT	1	SV		
		MOT	1	SM	X	;
CAR		MOT	1	C	X	;

TYPE CHANGEMENT DE SUPPORT

PARTITION	NB.FOIS	NATURE	LONGUEUR	NOM	FB	;
CTEX1		MOT	2	AUTEUR		
		MOT	3	CHAPITRE		
		MOT	2	PHRASE		
		MOT	3	ZONE		
	70	MOT	1	CAR	X	;

PARTITION	NB.FOIS	NATURE	LONGUEUR	NOM	FB	;
CTEX		RIEN				
		NA	3	CHAP		
		NA	2	PHR		
		RIEN				
	70	MOT	1		X	;

Type TRAITEMENT

```
lec     I:=PREMIER; R:=1; J:=PREMIER ;
LEC     L:=PREMIER;
E       TEX:=TEXTE-DON(I);
        TRAV(PREMIER:DERNIER,L):=Z DE TEX;
        SI L=DERNIER VERS E1; AV I; AVL; VERS E;
E1      K:=PREMIER;
CONSTR  CH DE IMOT:=CHA DE TEX; PH DE MOT:=PHR DE TEX;
        SM DE IMOT:=TRAV(K,PREMIER); AV K;
        ZA(1) DE IMOT:=C DE TRAV(K,PREMIER);
        ZB(1) DE IMOT:=C DE TRAV(K,1AP PREMIER);
        ZC(1) DE IMOT:=C DE TRAV(K,DERNIER); AV K; N:=N+1;
        MAJ:=VRAI; SI TRAV(K,PREMIER)='*' VERS E2; MAJ:=FAUX;
        N:=2;
E2      SI C DE TRAV(K,PREMIER)='.' VERS E3;
        SI C DE TRAV(K,PREMIER)=',''OU'-'ou'%' VERS E4;
        ZA(N):=C DE TRAV(K,PREMIER); ZB(N):=C DE TRAV(K,1 AP PREMIER);
        ZC(N):=TRAV(K,DERNIER); AV K; N:=N+1; VERS E2;
E4      LG DE IMOT:=N; RG:=R; R:=R+1;
        SV:=TRAV(K,PREMIER); SI K=DERNIER VERS E5;
        K:=K1; AV K; TO DE TRAV(K,PREMIER)≠ C DE TRAV(K1,PREMIER)VERS E5;
        SMOT(J):=IMOT; AV J; VERS CONSTR;
E3      LG DE IMOT:=N; RG:=R; R:=1;
        SV:=C DE TRAV(K,PREMIER);
E5      SMOT(J):=IMOT; SI I=DERNIER VERS FIN;
        AV J; AV I; VERS LEC;
        FIN
```

Exemple 2.

TRI(ACLASSER,INDICATIF;;CLASSER)

Introduction

La procédure TRI permet de classer un fichier sur bande. Le fichier, d'abord réécrit sur disque est trié sur deux fichiers auxiliaires placés sur disque, par ventilation et interclassement.

Les paramètres

Deux paramètres-donnée

- ACLASSER est une file sur bande dont l'article est structuré par QUELCONQUE.
  QUELCONQUE a ceci de particulier : elle admet n'importe quel mode élémentaire (ADLIB).
- INDICATIF est une partition de QUELCONQUE plus exacteme un identificateur de partie d'article structuré par QUELCONQUE.

Un paramètre résultat

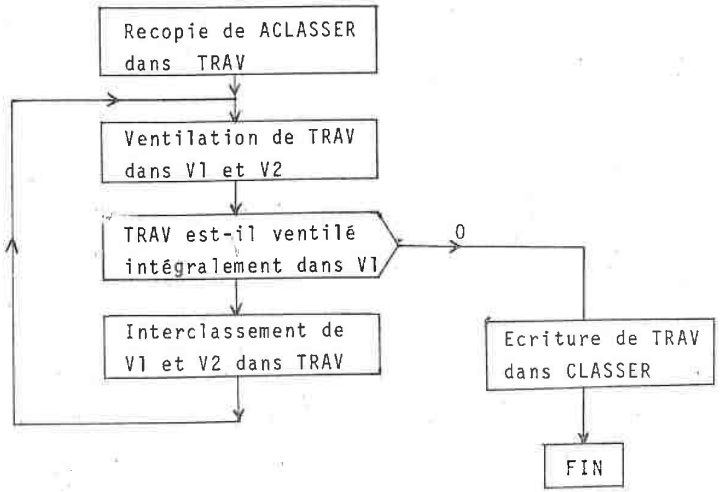
- CLASSER est une file sur bande dont l'article est structuré par QUELCONQUE.

Généralités sur l'écriture de la procédure

Les files TRAV, V1 et V2 situées dans le disque ont la longueur de la file ACLASSER et leurs articles sont aussi structurés par la partition QUELCONQUE, de même que les articles AUX1 et AUX2.
ACLASSER est recopié intégralement dans TRAV.

Le tri s'effectue de TRAV sur V1 et V2 en fonction de l'indicatif INDICATIF. L'interclassement s'effectue de V1 et V2 dans TRAV. L'opération de tri se poursuit jusqu'à ce que TRAV soit ventiler intégralement dans V1. La file TRAV est alors écrite sur la file CLASSER sur bande.

Organigramme



TYPE SPECIFICATIONS

STATUT	NATURE	NOM	LONGUEUR	PARTITION		INDEX	PLACE
				NOM	DANS	NOM	
DO	FILE	ACLASSER		QUELCONQUE		I	BANDE
DO	PARTITION	INDICATIF		QUELCONQUE			
RES	FILE	CLASSER		QUELCONQUE		R	BANDE
AUX	FILE	TRAV	LON-ACLASSER	QUELCONQUE		J	DISQUE
AUX	FILE	V1	LON-ACLASSER	QUELCONQUE		K	DISQUE
	FILE	V2	LON-ACLASSER	QUELCONQUE		L	DISQUE
AUX	INDEX	I					
	INDEX	J					
	INDEX	K					
	INDEX	L					
AUX	ARTICLE	AUX1		QUELCONQUE			
AUX	ARTICLE	AUX2		QUELCONQUE			
AUX	BO	B	1				

TYPE PARTITIONS

PARTITION	NB.FOIS	NATURE	LONGUEUR	NOM
QUELCONQUE		ADLIB		

```
        Type TRAITEMENT


        TRAV(PREMIER:DERNIER):=ACLASSER(PREMIER:DERNIER);
        K:=DERNIER; L:=DERNIER; J:=DERNIER;
E1      B:=FAUX;
        AUX2:=TRAV(J);
E5      AUX1:=AUX2; V1(K):=AUX1; SI J=PREMIER VERS E3;
        AR J; AUX2:=TRAV(J);
        SI INDICATIF DE AUX2   INDICATIF DE AUX1 VERS E4;
E7      AR K; VERS E5;
E4      SI B=VRAI VERS E6; B:=VRAI;
E8      AUX1:=AUX2; V2(L):=AUX1; SI J=PREMIER VERS E3;
        AR J; AUX2:=TRAV(J);
        SI INDICATIF DE AUX2   INDICATIF DE AUX1 VERS E7;
E6      AR L DE 1; VERS E8;
E3      SI B=FAUX VERS E9; AUX1:=V1(K); AUX2:=V2(L);
E12     SI INDICATIF DE AUX1   INDICATIF DE AUX2 VERS E10;
        TRAV(J):=AUX1; SI K=DERNIER VERS E11;
        AV J; AV K; AUX1:=V1(K); VERS E12;
E10     TRAV(J):=AUX2; SI L=DERNIER VERS E13; AV J; AV L;
        AUX2:=V2(K); VERS E12;
E11     AV J; TRAV(J):=AUX2;
E14     SI J=DERNIER VERS E1; AV J; AV L;
        TRAV(J):=V2(L); VERS E14;
E13     AV J; TRAV(J):=AUX1;
E15     SI J=DERNIER VERS E1; AV J; AV K;
        TRAV(J):=V1(K); VERS E15;
E9      CLASSER(PREMIER:DERNIER):=TRAV(PREMIER:DERNIER);
        FIN
```

Exemple 3.

PROCEDURE SYMETRIQUE(;MOT;)

Introduction

La procédure SYMETRIQUE permet de remplacer un mot par son réfléchi.

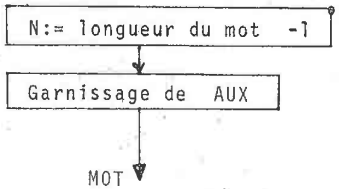
Paramètre

Le seul paramètre est un paramètre mixte MOT.
Mot désigne un article structuré par la partition SM.
La partition comporte :

- un naturel de longueur 2 : NB qui désigne la longue en caractères du mot
- le mot lui-même de longueur variable M.

Généralités

La file AUX dont l'article est un caractère reçoit le mot qui est réfléchi dans MOT, reçoit la longueur du mot.



TYPE SPECIFICATIONS

STATUT	NATURE	NOM	LONGUEUR	PARTITION		E	NOM	PLACE
				NOM	DANS			
MIXTE	ARTICLE	MOT		SM				
AUX	FILE	AUX	30	CAR	CATAL	1	A	
AUX	INDEX	A						
AUX	NA	N	2					

TYPE PARTITIONS

PARTITION	NB.FOIS	NATURE	LONGUEUR	NOM
SM		NA	2	NB
	NB	MOT	1	M

TYPE TRAITEMENT

```
N:=NB DE MOT-1;
A:=N APRES PREMIER;
AUX(PREMIER:A):=M DE MOT ;
N:=1;
E  M(N)DE MOT:=AUX(A);
SI A=PREMIER VERS FIN ;
AR A DE 1; N:=N+1; VERS E;
FIN
```

NOM DE L'ETUDIANT : DIRAND Jean Marie

Nature de la Thèse : Spécialité en Mathématiques Appliquées

Vu, Approuvé

et Permis d'Imprimer

NANCY, le 26 août 1969

Le DOYEN :

J. AUBRY