UNIVERSITÉ D'ALGER

ÉCOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ECONOMIE

THÈSE DE FIN D'ÉTUDES

PROJET MIX

UN ASSEMBLEUR POUR MIXAL

Etudié:

M.TAOUCHICHET
S.BENDIFALLAH

Direction et Patronnage:

F.D.ARMINGAUD
N.BOUMAHRAT

JUIN 1974

U N I V E R S I T E D' A L G E R

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ECONOMIE

THESE DE FIN D'ETUDES

P R O J E T M I X

U N A S S E M B L E U R P O U R M I X A L

Sujet etudie par:
Salah BENDIFALLAH
Mustapha TAOUCHICHET

propose par:
F.D. ARMINGAUD
N. BOUMAHRAT

Juin 1974

A

tous ceux qui, au sein de l'ECOLE POLYTECHNIQUE, ont contribue a notre formation;

MM. ARMINGAUD et BOUMAHRAT, pour leurs conseils et la sollicitude qu'ils ont temoignee a l'endroit de nos problemes;

MM. OUFERHAT et DAMINE, dont la collaboration au Centre de Calcul a ete a la mesure de nos besoins;

nos collegues du PROJET MIX, pour l'experience que nous avons acquise a leur contact;

Mr. DOLIATOVSKI, qui nous a inities aux problemes de la CYBERNETIQUE;

notre reconnaissance;

a tous les cybernéticiens, notre adhesion.

PREAMBULE

Le PROJET MIX devait essentiellement se distribuer en 3 sujets coopératifs*:

- un ASSEMBLEUR ;
- un SIMULATEUR ;
- une GESTION DE MEMOIRE, avec simulation de MEMOIRE VIRTUELLE; mais cette derniere partie, ayant ete jugee trop ambitieuse, a ete abandonnee en phase d'ebauche.

Une solution s'est presentee finalement dans la creation d'une deusieme version de l'Assembleur, les deux produits ne devant avoir en commun que la conception primaire, due aux exposes introductifs de notre directeur de projet.

C'est au cours de ces exposes que nous avons ete inities a la forme de programmation tres "coule de source" qu'est la PROGRAMMATION STRUCTUREE. Cependant, comme il apparaitra lors de l'etude que nous allons introduire, nous avons eu recours a la forme "ORGANIGRAMME" pour la presentation des modules au niveau des details.

---

* joker: la plupart des accents auront saute du texte, tant sous l'influence de KNUTH qu'a cause d'une machine a ecrire tres ordinaire.

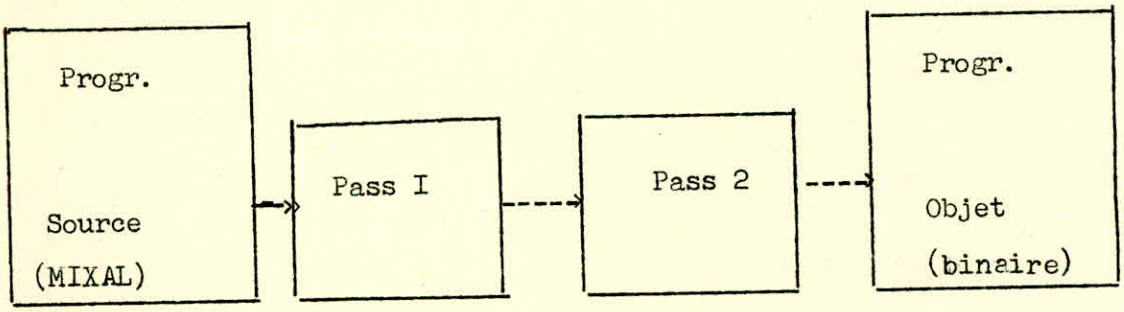
TABLE DES MATIERES

- INTRODUCTION -

I.- BUT DU TRAVAIL :

Etant donné un programme source en langage symbolique MIX, le rôle de l'Assembleur consistera à fournir au simulateur, au bout de deux (2) étapes, un programme codé prêt à être exécuté.

II.- DESCRIPTION :



Une carte MIX se presentera sous la forme suivante :

COL

I)
)---> ETIQUETTE
IO)

II -----> Blanc de séparation

I2)
)---> CODE OPERATION
I6)

I7 ----> Blanc de séparation

I8)
)---> OPERANDE, Blanc de séparation, COMMENTAIRES
76) I8 76

Dans la Ière étape, ou Ier passage, on s'intéresse à :

- la zone ETIQUETTE pour construire la table des symboles et la table des adresses correspondantes.
- aux zones CODE OPERATION et OPERANDE pour le traitement des pseudo-instructions, qui n'ont pour but que la reservation de place mémoire et la définition des constantes.

Dans la 2ème étape, ou 2ème passage on s'intéresse à :

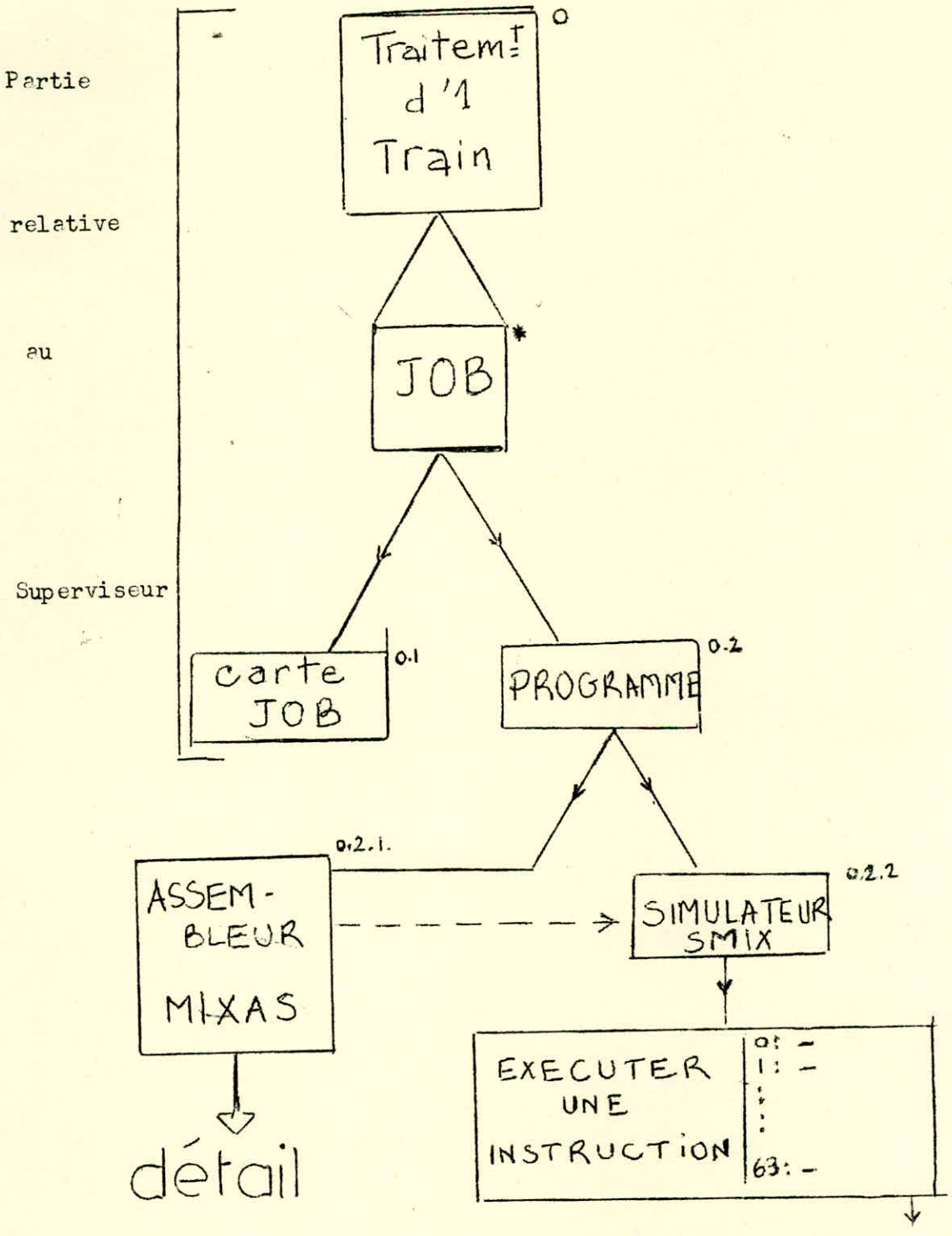
- la zone CODE OPERATION, pour reconnaître et coder l'opérateur symbolique MIX.
- la zone OPERANDE, pour reconnaître et coder l'opérande en se réferant aux valeurs d'adresse données par la table des symboles.
- au regroupement des zones ainsi codées pour remplir la table des instructions à transmettre au simulateur.

En plus des parties correspondants à ces deux (2) étapes, une 3ème partie traitera de l'option "Listing", sur test d'un indicateur fourni par le superviseur.

Le Listing reproduira les indications d'erreurs eventuelles.

Pour ce qui est de la presentation, nous montrerons prealablement comment se place notre "travail" dans le schema d'ensemble d'un train de travaux.

Schéma Structuré du traitement d'un train.



Le role du Superviseur a ete,dans notre travail,confie entierement au Programme Principal charge de:

-definir les zones-memoires(fichiers,zone COMMON)

-appeler MIXAS pour assurer l'assemblage

-si l'execution est permise,(pas de diagnostic d'erreur)appeler SMIX pour l'assurer

PROGRAMME PRINCIPAL"superviseur"

*Cartes de controle FORTRAN;

*Definition des fichiers utilitaires:(I,INSTR),(2,TSYMB),(3,MOPIX);

INSTR:fichier des instructions:accepte jusqu'a I00 instructions d'ou 200 enregistrements de 80 mots dont I00 pour le listage des diagnostics d'erreurs(cf module LEURR);
DF I(200,80,U,II); occupe 50 secteurs;

TSYMB:fichier de la table des symboles :accepte jusqu'a 50 symboles de I0 caracteres au maximum et leur valeur d'adresse,d'ou II mots par enregistrement;
DF 2(50,II,U,IS);occupe 2 secteurs;

MOPIX:fichier des operateurs MIX:chaque POPR occupe 2 mots,
DF 4(I44,2,U,IC);occupe I secteur;

*Creation du fichier 4;auparavant,definition de la zone COMMON;

*Initialisation du compteur d'erreurs(LAW=0);

*Indication de l'option"Listing":LIST=I si oui,LIST=0 si non;

*Appel de MIXAS,l'ASSEMBLEUR MIX;

*Test de LAW:IF LAW≠0 THEN INDICATION du nombre(LAW) d'ERREURS,
EXECUTION SUPPRIMEE;
ELSE appel de SMIX,le SIMULATEUR MIX:
CALL LINK(SMIX)Pour sauvegarder la zone COMMON;

END

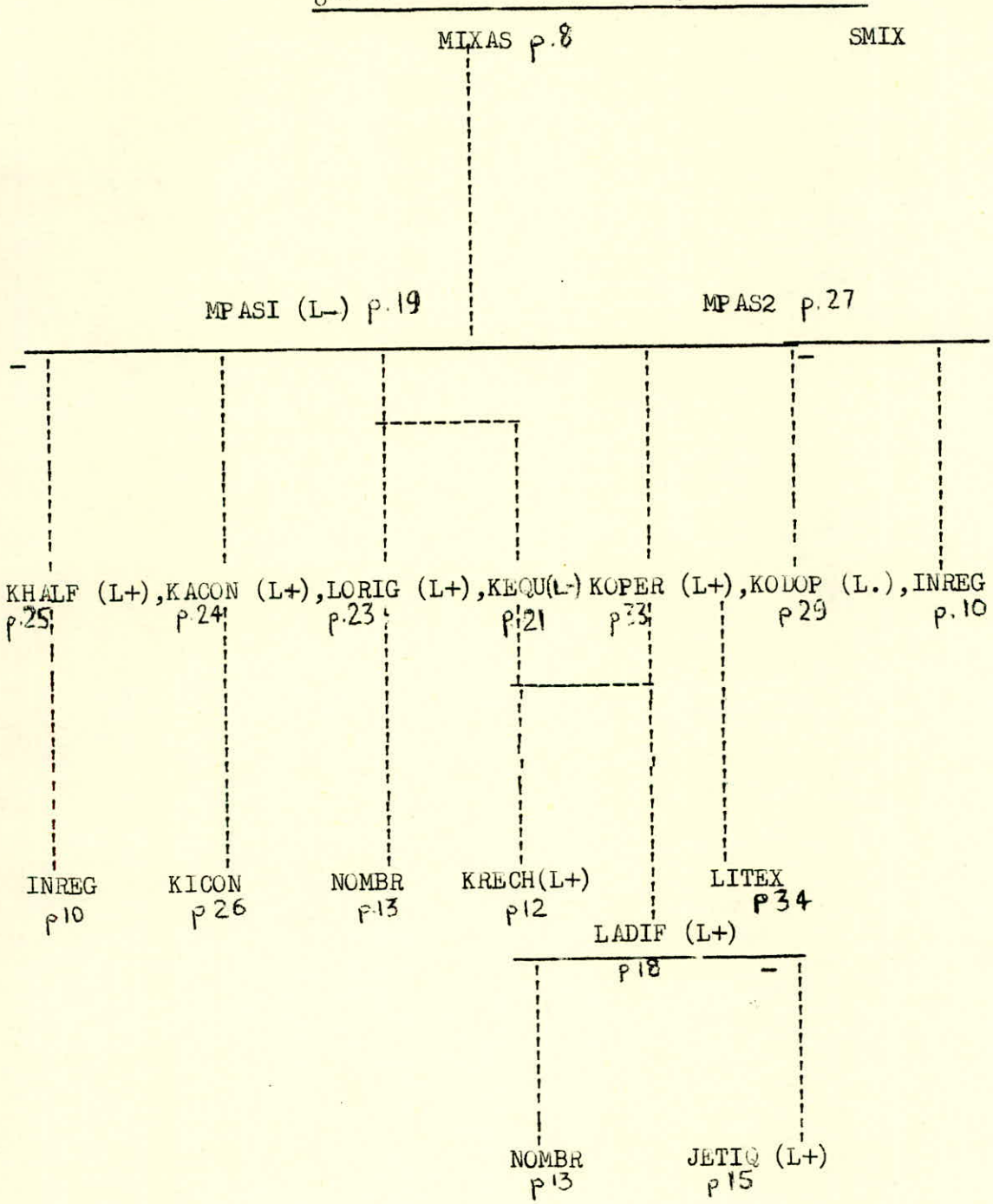
*Cartes de controle FORTRAN;
// XEQ 2
*FILES(4,MOPIX)
*FILES(I,INSTR)

Operateurs,Programme en MIXAL

Remarque:l'utilisation du FORTRAN s'est imposee d'emblee, vu l'importance des automates de reconnaissance;seuls les programmes de regroupement de bytes ont ete ecrits en ASSEMBLEUR II30.

*Occupation-memoire#370 + zone COMMON=302

STRUCTURE MODULAIRE DE L'ENSEMBLE DU PROGRAMME



(L+): appel du module LEURR pour indication d'erreur sur OPERANDE;
(L.): " " p9 " " OPERATEUR;
(L-): " " " " LETIQUETTE.

Voici la MEMOIRE MIX a laquelle nous allons nous referer:

ACCumulateur = registre RA: 32 bits:-byte de signe:2 bits;
-5 bytes de 6 bits;
I mot MIX= 2 mots II30

EXTension de l'ACC = registre RX:identique a RA

6 registres d'INDX:RII,RI2,RI3,RI4,RI5,RI6
auxquels correspondent dans la zone OPERANDE
les nombres (NBRI):=I/2/3/4/5/6
-chaque RI comporte le
byte de signe + 2 bytes
de 6 bits(par extension,
I mot II30);

I registre RJ,auxiliaire des instructions de Sauts
le byte de signe toujours positif,+ 2 bytes de 6 bits
(par extension,I mot II30)

Indicateur d'Overflow:TOG(Overflow toggle)

Indicateur de comparaison:LEG (Less than-Equal-Greater than)

Comments:

Le Byte est l'unite de base de l'information:si b est sa taille en bits,il peut contenir les valeurs comprises entre 0 et (2**b-I);donc pour 6 bits:0,63
pour 2 bytes adj:0,4095
3 " " :0,262I43
4 " " :0,I67772I5
5 " " :0,I07374I823

De telles valeurs seront par ex. traitees par les operateurs d'Entree/Sortie.

: MIX :est une machine fictive "parfaite",que DONALD E. KNUTH a imaginee a des fins pedagogiques dans son cours "THE ART OF COMPUTER PROGRAMMING";elle est "superieure"a presque toutes les machines existantes, ou elle peut aisement etre simulee.Son langage,MIXAL,est presente comme assez puissant pour permettre d'ecrire de petits programmes simples pour la plupart des algorithmes.

Module general de l'ASSEMBLEUR: MIXAS

1/ Mise a blanc des enregistrements pairs du fichier INSTR, destines a recevoir les indications d'erreurs;

2/ Sauvegarde de l'adresse de chargement du programme executable: IAR=ADDR(TINST(100)); TINST est la zone COMMON qui doit recueillir les instructions codees; le Simulateur traitera cette zone en BSS;

Initialisation du compteur ordinal (compteur d'instructions) au contenu de l'IAR; M=IAR;

3/ Initialisation a 0 des differents bytes de l'instruction MIX;

4/ Appel du module traitant le Ier Passage, MPASI;

5/ Appel du module traitant le 2eme Passage, MPAS2;

6/ Test de l'indicateur LIST pour impression, si demandee, a partir du fichier INSTR, DES informations suivantes:

M, (MEMIX(I), I=I, 5), KOUNT, (LETIQ(I), I=I, II), POPR, (IPOPD(I), I=I, 60);

M: adresse de l'instruction;

MEMIX: les 5 champs de l'instruction:

Signe, AAdresse, INDX, Field, Kode-operation;

KOUNT: numero de la carte;

LETIQ: 10 caracteres pour l'etiquette, + blanc de separation;

POPR: operateur MIX;

IPOPD: zone operande: blanc, operande, blanc, commentaires.

*Occupation-memoire# 360

MODULES DE BASE

Nous entendons par modules de base les modules communs aux Ier et 2eme Passages. Ils sont au nombre de 5:

LEURR , INREG , KRECH , NOMBR , JETIQ .

LEURR:

a-But: Traitement des erreurs.

Nous nous proposons avec LEURR, qui sera appele par les autres modules en cas d'erreur,

-de positionner un curseur a l'emplacement exact de l'erreur sur la ligne qui suit immediatement l'instruction:

*pour une erreur en zone ETIQUETTE ou zone OPERANDE;

**** pour une erreur sur l'OPERATEUR;

Ex.:

```
LOO+INGbbbbCNP AbMINIIMAXIO,8   .enregistrement de l'instruction
                                :
'-_*_____1 ****'_____*_____1    :.     "  des indications d'erreurs,
                                              dans le fichier I.
 z.etiquette    z.operande
```

pour cela ce module a un parametre: INDIC, qui peut prendre 3 valeurs suivant la zone ou a ete detectee l'erreur:

INDIC= -I pour la zone "etiquette"

INDIC= 0 " " "operateur"

INDIC= +I " " "operande"

-de determiner le nombre total d'erreurs dans le programme:

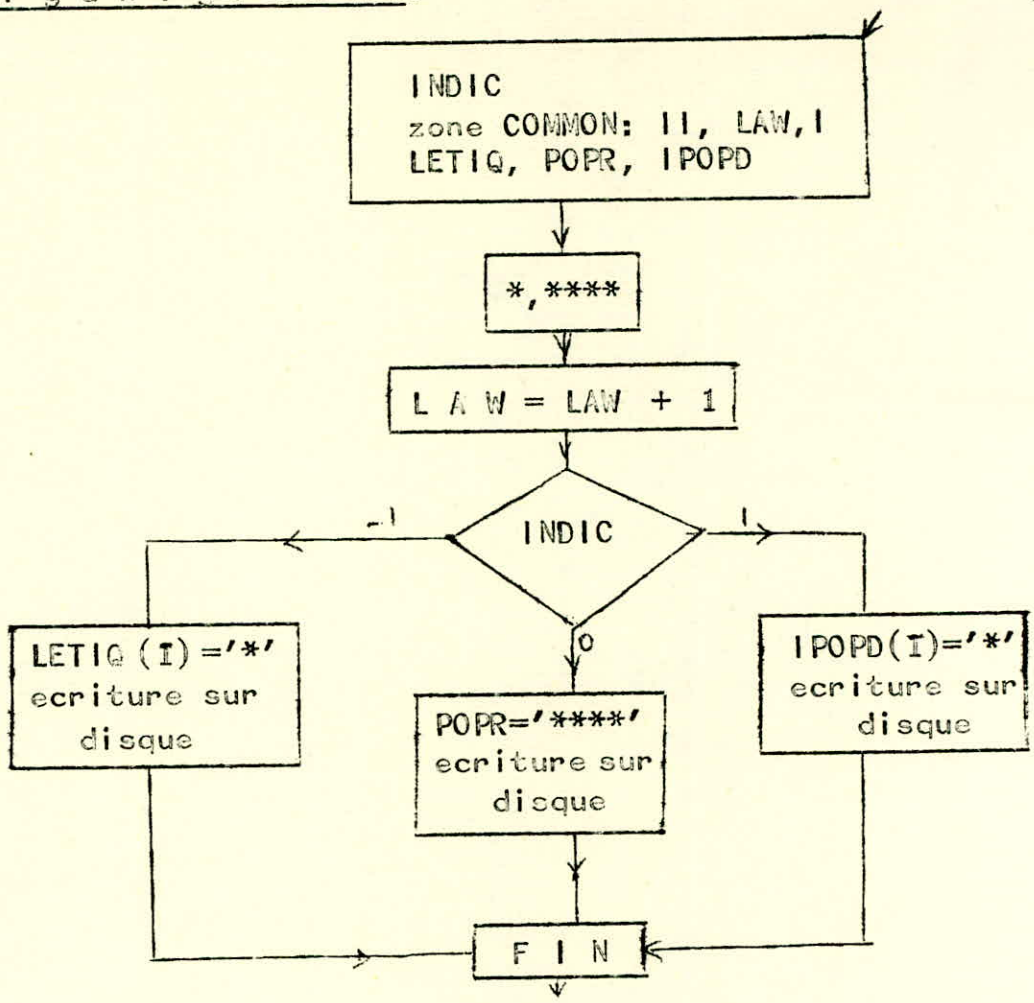
pour cela un compteur LAW a ete defini en zone COMMON, qui est incremente de I a chaque intervention de LEURR.

occupation-memoire: #80 mots

b - Organigramme:

paramètres

Data Curseurs



I N R E G :

a - But : A la fin du 2ème passage, l'instruction codée se trouve dans 5 mots (1130) (zone COMMON, MEMIX (5). Or le simulateur doit recuperer cette instruction dans 1 mot MIX (soit 2 mots (1130).

C'est le rôle de ce module de faire le regroupement selon le schema suivant:

2/...

1 signe: 2 bits

2 adresse: 12 bits

3 index : 6 bits

4 champ : 6 bits

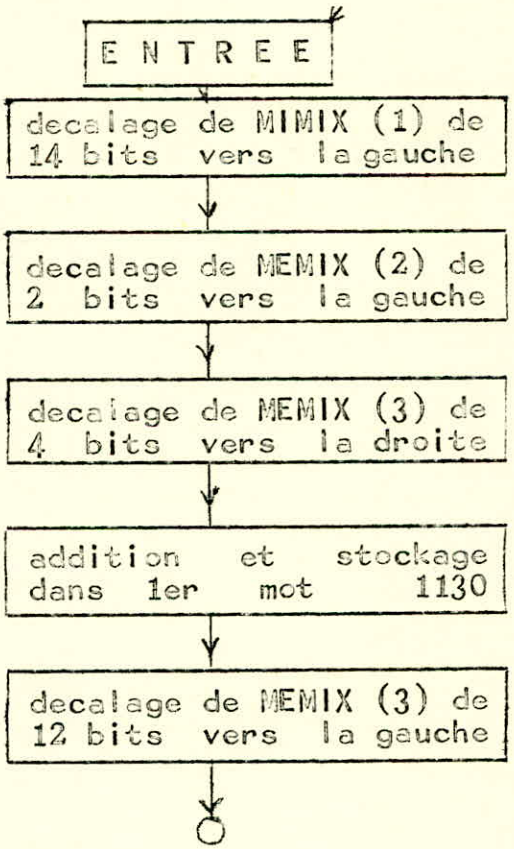
5 codop : 6 bits

INREG ⟹

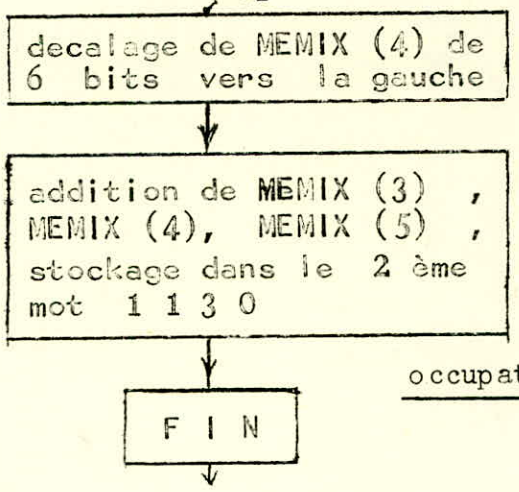
1er mot 1130			2ème mot 1130	
1	2	3	4	5
2bits	12 bits	6 bits	6 bits	6 bits

Remarque: le langage de programmation de ce module est l'ASSEMBLEUR 1130, qui permet, de faire les opérations de decalage d'une part , et l'accés à la cellule mémoire d'autre part.

b - Organigramme :



3/...



occupation-memoire# 40 mots

K R E C H

a - But : Reconnaître et calculer la valeur d'une specification de champ (L:R) avec $0 \leq L \leq R \leq 5$.

On definit dans ce module un indicateur (KRACK) destiné à être tester dans le module qui appelle KRECH. S'il est ON sortir de ce module appelant pour éviter de faire un traitement inutile (une erreur ayant été déjà detectée).

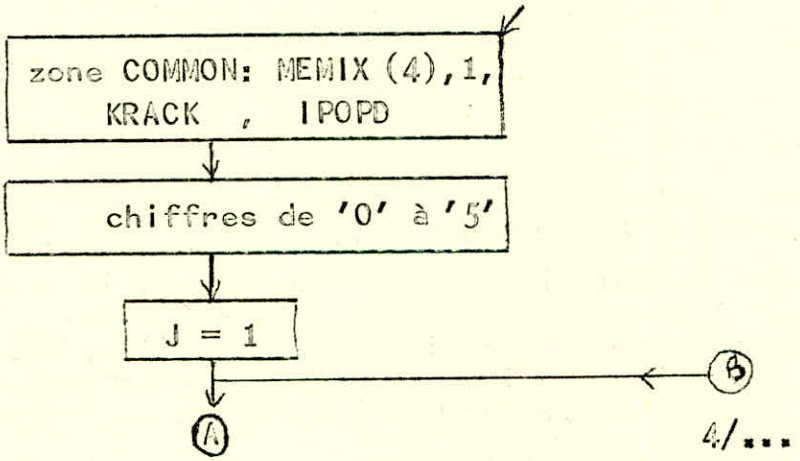
Remarques: - Le caractère ':' qui indique pour KNUTH, l'opération binaire L:R dont le resultat est 8L + R a été remplacé par le caractère '.' C'est la reconnaissance de ce p o i nt qui permet l'appel de ce module.

- Nous utilisons dans ce module le sous-programme IVAL (FUNCTION) pour prendre la valeur numérique d'un chiffre donné sous format A 1.

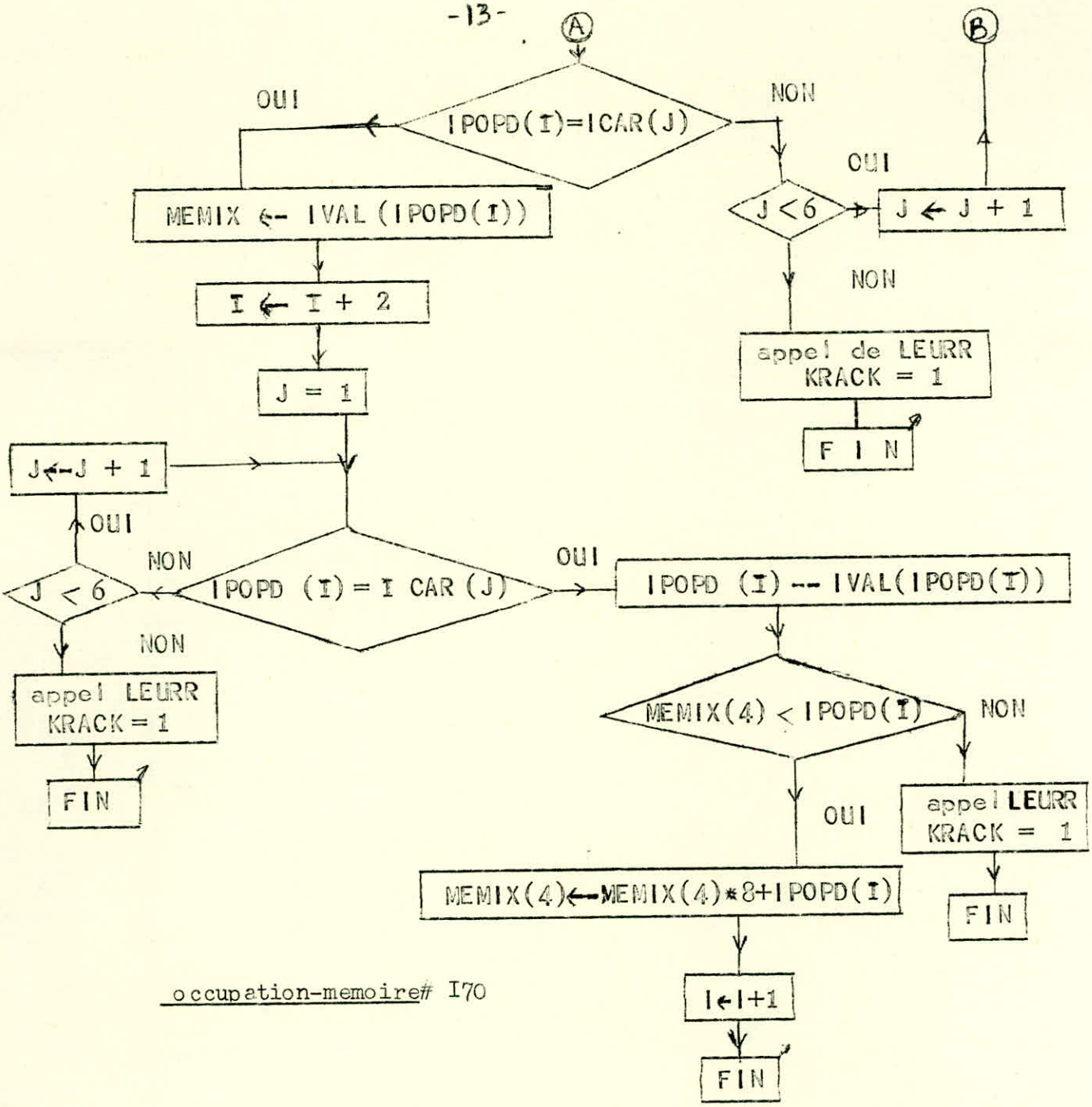
b - Organigramme:

paramètres

Data I C A R



4/...



occupation-memoire# I70

N O M B R:

a - But: Reconnaissance et calcul d'un entier sans signe (ESS) dans l'opérande. l'ESS sera une suite d'au plus 5 chiffres (32767).

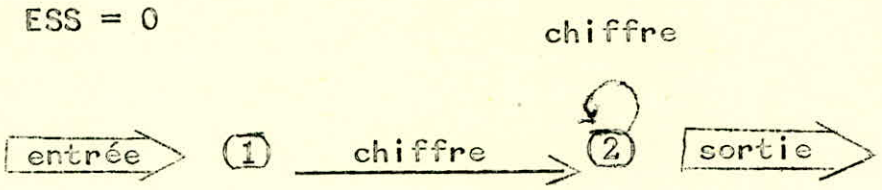
Ce module utilise aussi l'indicateur KRACK et la FUNCTION IVAL (voir module KRECH).

occupation-me moire# I60

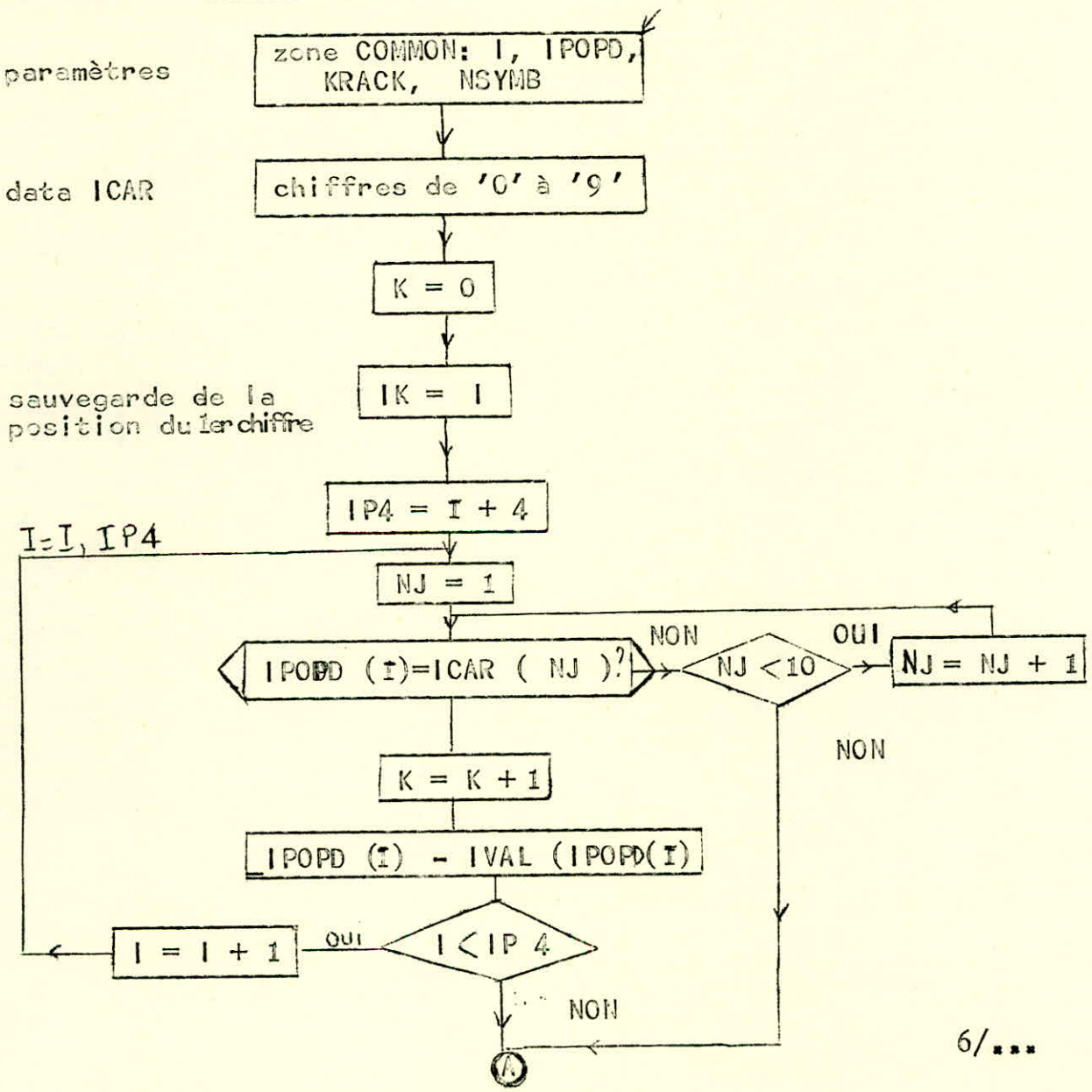
5/...

b - Automate de Reconnaisse de l'ESS :

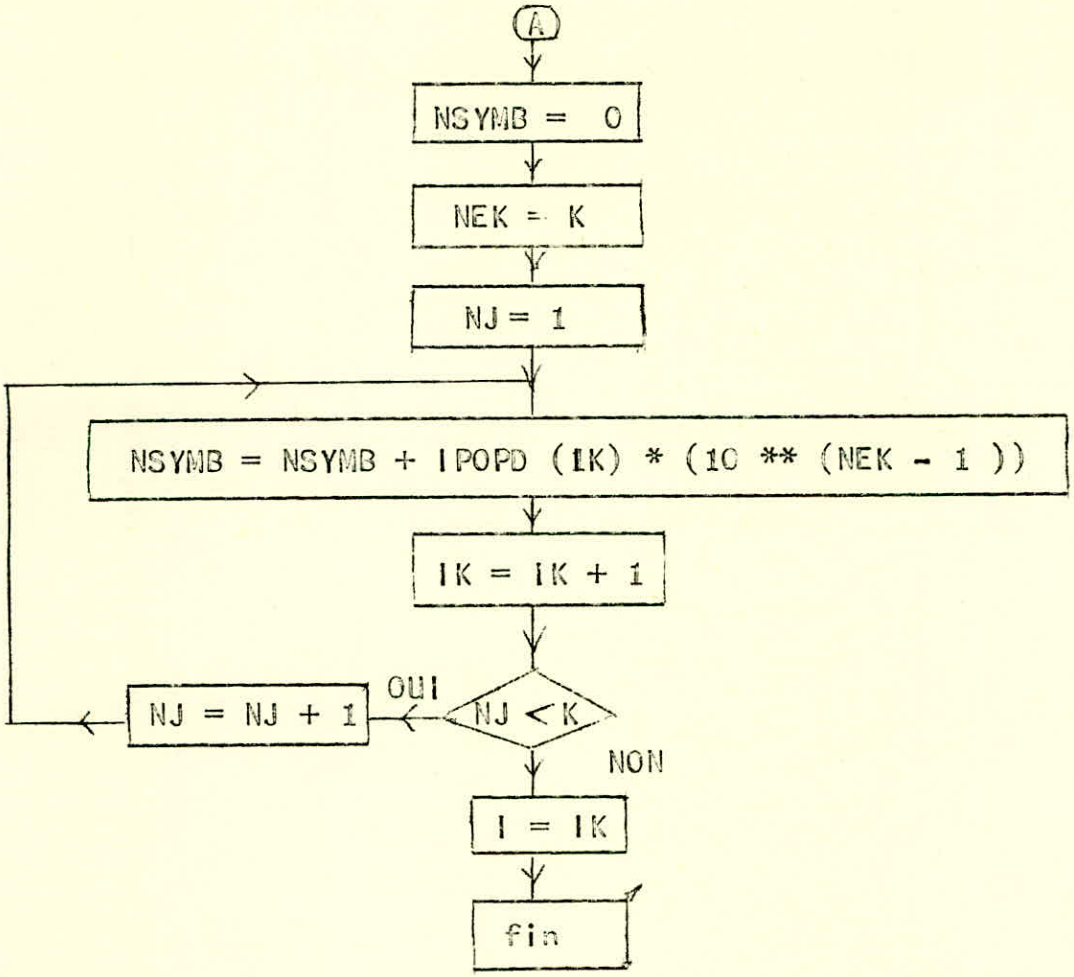
Si un chiffre a été reconnu au paravant (voir LADIF)



c - Organigramme:



6/...



J E T I Q :

a - But: - Reconnaissance d'une ETIQUETTE dans la zone OPERANDE, l'ETIQUETTE étant une suite d'au plus 10 caractères alphanumériques, dont le 1er est abligatoirement une lettre.

- L'étiquette étant reconnue, recherche de sa valeur d'adresse NSYMB dans la table des symboles.

Remarques: Au début ce module doit mettre à blanc la zone LETIQ (10), qui recevra les caractères au fur et à mesure qu'ils sont reconnus, car si l'Etiquette comprend moins de 10 caractères, il est necessaire que le reste soit à blanc pour la comparaison avec les étiquettes de la table des symboles (NSYMB).

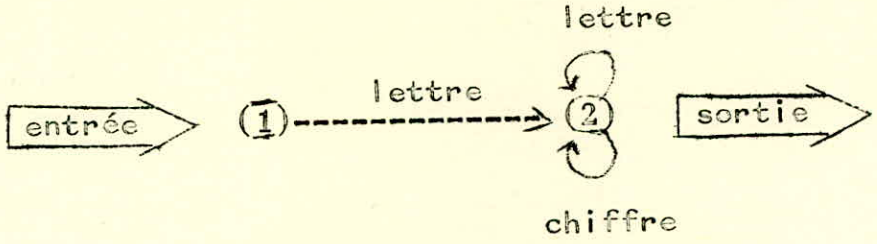
- Ce module utilise aussi l'indicateur KRACK (voir son rôle dans KRECH).

7/...

*occupation-memoire# 260

b - Automate de reconnaissance de l'etiquette:

Si une lettre a été reconnu auparavant (LADIF)



c - Organigramme :

paramètres

zone COMMON: IS, I
LETIQ,N, IPOPD, KRACK,
LSYMB NSYMB

N est le nbr d'etiq ds la table des symb, determine au Ier pass.

data ICAR

blanc , Alphanumerique

mise a blanc de la zone LETIQ (10).

LETIQ (J) = blanc

J=1,10

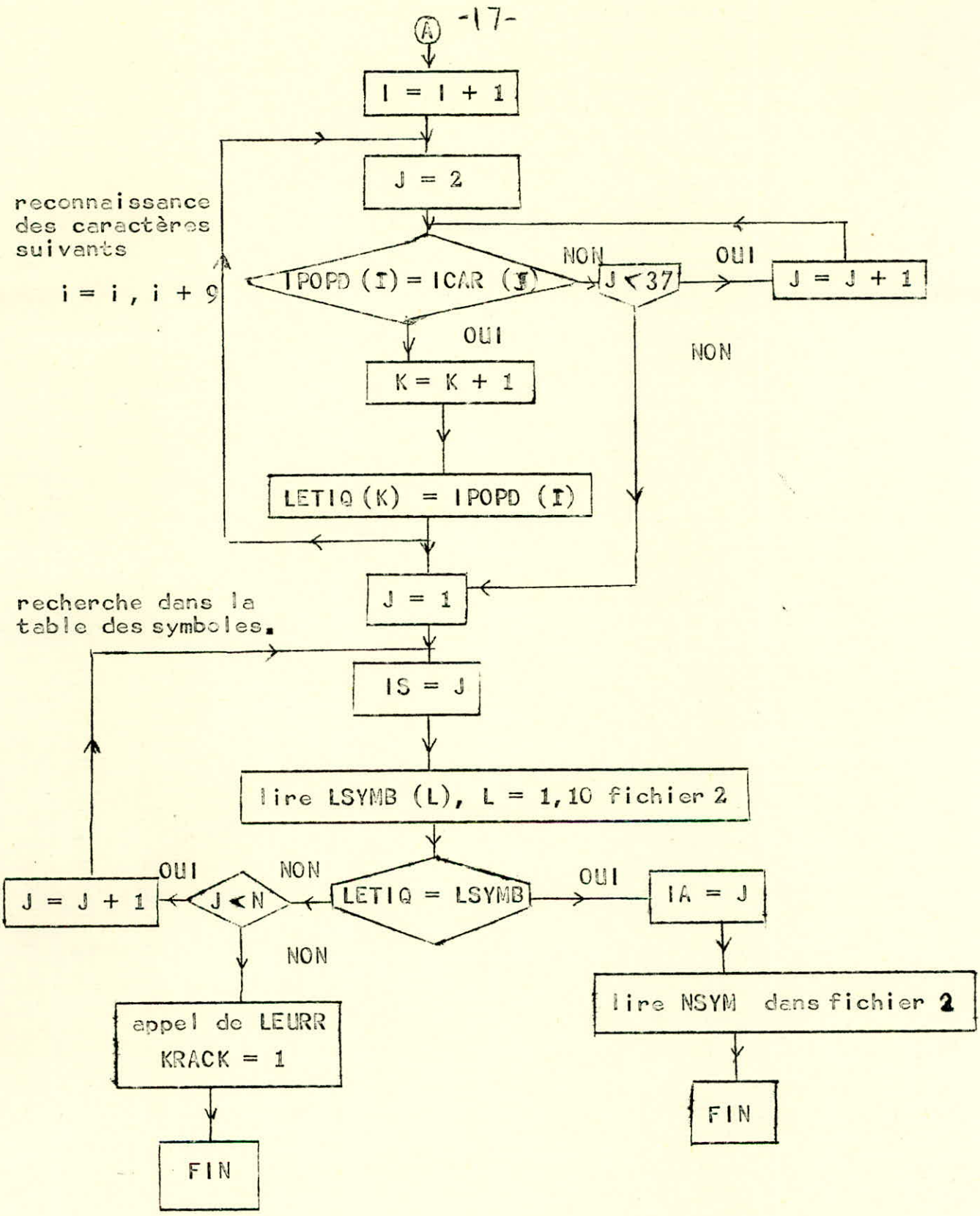
K: compteur du nombre de caractères

K = 1

LETIQ (1) = IPOPD (1)

A

8/...

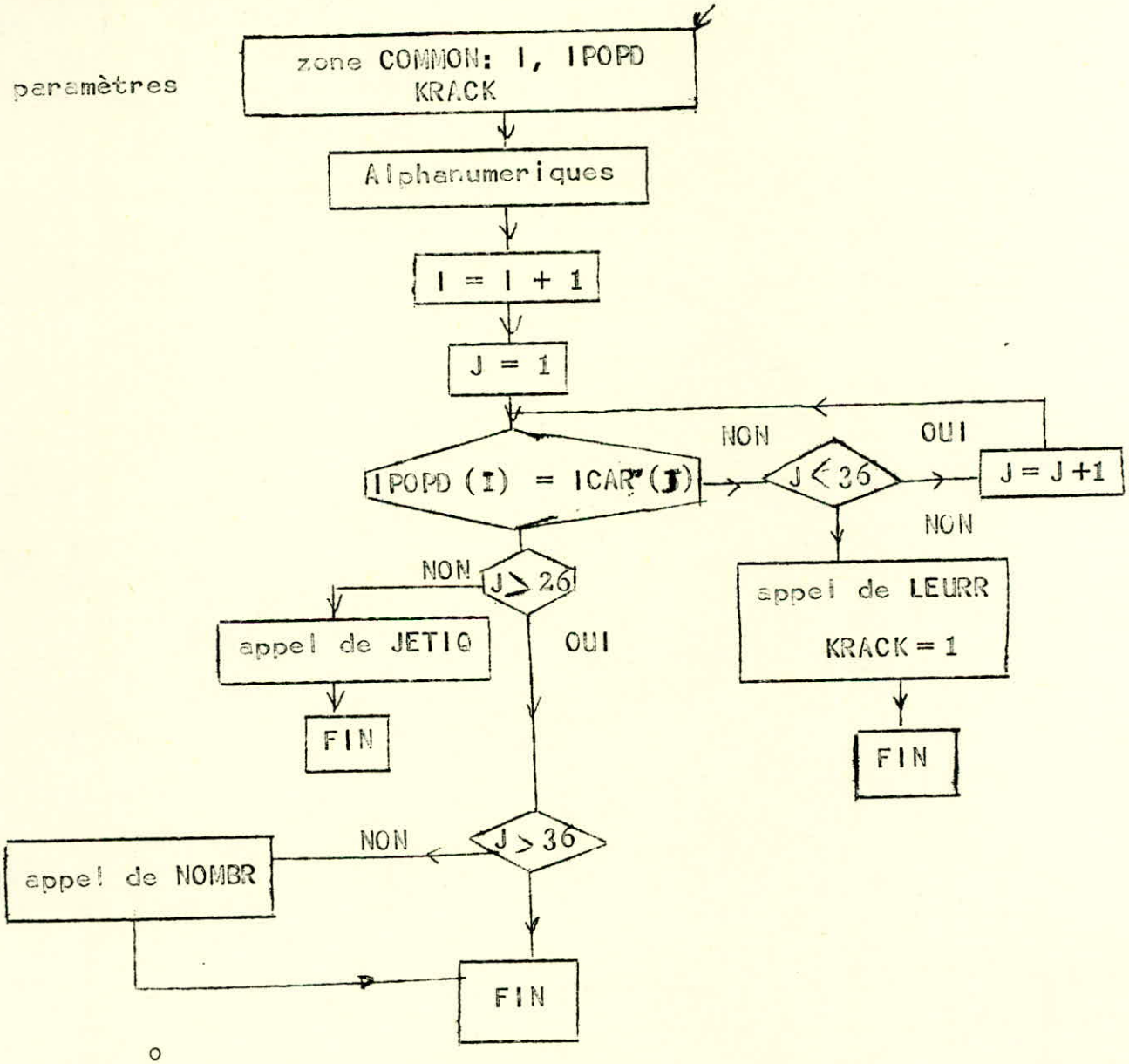

Ⓐ
I = I + 1
J = 2
reconnaissance des caractères suivants
i = i , i + 9
IPOPD (I) = ICAR (I)
NON
J < 37
OUI
J = J + 1
OUI
NON
K = K + 1
LETIQ (K) = IPOPD (I)
J = 1
recherche dans la table des symboles.
IS = J
lire LSYMB (L), L = 1,10 fichier 2
OUI
NON
J = J + 1
J < N
LETIQ = LSYMB
OUI
IA = J
NON
appel de LEURR
KRACK = 1
lire NSYM dans fichier 2
FIN
FIN

R E M A R Q U E :

Un 6ème module peut être considéré comme module de base, bien qu'il ne reponde qu'à un souci de performance.

C'est le module LADIF, utilisé dans le traitement de l'opérande. Il appelle les modules NOMBR ou JETIQ selon qu'il ait reconnu comme 1er caractère un chiffre ou une lettre.

O r g a n i g r a m m e :



o

occupation-memoire# II0

1er PASSAGE : MODULE MPAS1

BUT : Le 1er Passage doit atteindre un double objectif.

1 - CONSTRUCTION DE LA TABLE DES SYMBOLES ET DES ADRESSES CORRESPONDANTES :

En effet le problème qui se pose à l'assemblage est celui de l'attribution des valeurs d'adresses aux étiquettes de la zone OPERANDE.

Soit une instruction comportant une étiquette en zone OPERANDE, il y a 2 cas possibles :

- l'étiquette a été dérinie dans une instruction précédente en zone ETIQUETTE ;
- l'étiquette sera définie dans la suite du programme. Dans ce cas il y a indétermination de la valeur d'adresse.

Ainsi la solution consiste-t-elle à faire dans un premier temps le recensement systématique des étiquettes (zone ETIQUETTE). Ce qui revient à construire une table, établissant la correspondance entre les étiquettes et leurs valeurs d'adresses.

Il découle de ce processus qu'aucun symbole ne doit apparaitre en zone ETIQUETTE plus d'une fois, et que tout symbole apparaissant en zone OPERANDE doit être définie en zone ETIQUETTE.

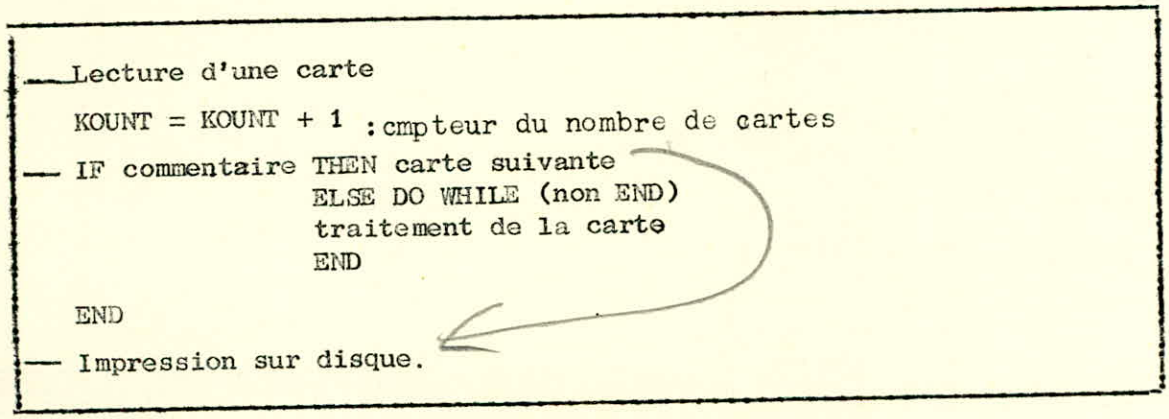
2 - TRAITEMENT DES PSEUDO-INSTRUCTIONS :

Les opérations de pseudo-instruction, sont des opérations de définition de constantes ou de réservation de place mémoire. C'est pour cela qu'elles doivent être traitées au 1er passage.

DESCRIPTION :

L'enchainement des opérations en programmation structurée est :

- Boite 1 : 1er Passage.



.....

- Boite 2 : Traitement de la carte

```
→ IF étiquette  THEN  traitement de l'étiquette
                IF  pseudo-Instr. THEN traiter pseudo-Instr.
                                  ELSE  IAR = IAR + 1
                                        (incrementation MIX)
                END
→  END
```

- Boite 3 : Traitement de l'étiquette

```
→ Automate de reconnaissance de l'étiquette
→ Recherche sur disque
  IF étiquette déjà sur table  THEN  erreur
                               ELSE  l'y mettre avec sa valeur d'adresse qui dst dans IAR.
  END
```

- Boite 4 : Traitement de la pseudo-instruction

```
→ CASE  PSIN OF :
        EQU : Traitement de EQU (appel de KEQU)
        ORIO : Traitement de ORIO (appel de LORIG)
        ALF :  Traitement de ALF (appel de KHALF)
        CON :  Traitement de CON (appel de KACON)
  ESAC
```

a) - Description détaillée de la Boite 1

- La lecture de la carte se fait sous le format suivant :

  . Zone ETIQUETTE : caractères par caractères (11 A1)
  LETIQ (11), le blanc de séparation étant compris.

  . Zone OPERATEUR : lue enbloc (A4) POPR

  . Zone OPERANDE : caractère par caractère, le blanc de séparation étant compris (60A1), IPOPD (60)

- Impression sur disque :

Nous avons défini un fichier de 200 enregistrements INSTR (Nº 1), de 80 mots chacun. Il peut recevoir un programme d'au plus 100 instructions. En effet l'autre moitié est réservé à l'indication des erreurs, selon la méthode exposée dans le module LEURR.

.....

- Commentaires :

Une carte commentaire est annoncée par le caractère Astérix ('*') en première colonne.

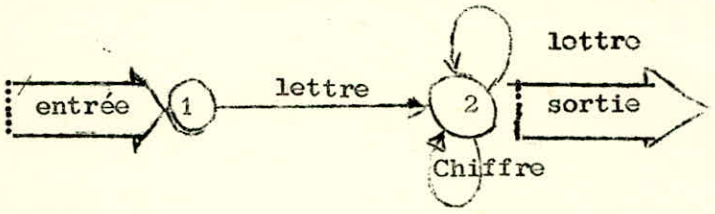
b - Description détaillée de la Boite 2 :

- La présence de l'étiquette est testée sur l'existence du 1er caractère (LETIQ (1) seulement ce qui correspond à l'étiquette de longueur minimum. Ce caractère doit obligatoirement être une lettre.

- La reconnaissance d'une pseudo-instruction se fait par le test de POPR par rapport à une table de pseudo-instructions (PSIN: EQU, ORIG, CON, ALF, END) donnée en DATA.

c - Description détaillée de la Boite 3 :

- Automate de reconnaissance de l'étiquette.



- Recherche dans la table des symboles :

La table des symbolesest définie dans le fichier T SYMB (Nº 2) DE 50 enregistrements, de 11 mots chacun (10 pour L'ETIQUETTE, le 11ème pour sa valeur d'adresse).

Soit N le nombre d'étiquettes se trouvant déjà dans la table. On compare le symbole à recherche, caractère par caractère à chaque enregistrement lu dans le fichier 2.

occupation-memoire du module MPASI# 600

d - Description des modules de la boite 4 :

- KEQU :

a - but : Traitement de la pseudo-instruction EQU qui fait correspondre à un symbole une valeur numérique non nulle.

Cette valeur représente :

- Soit une adresse
- soit une spécification de champ.

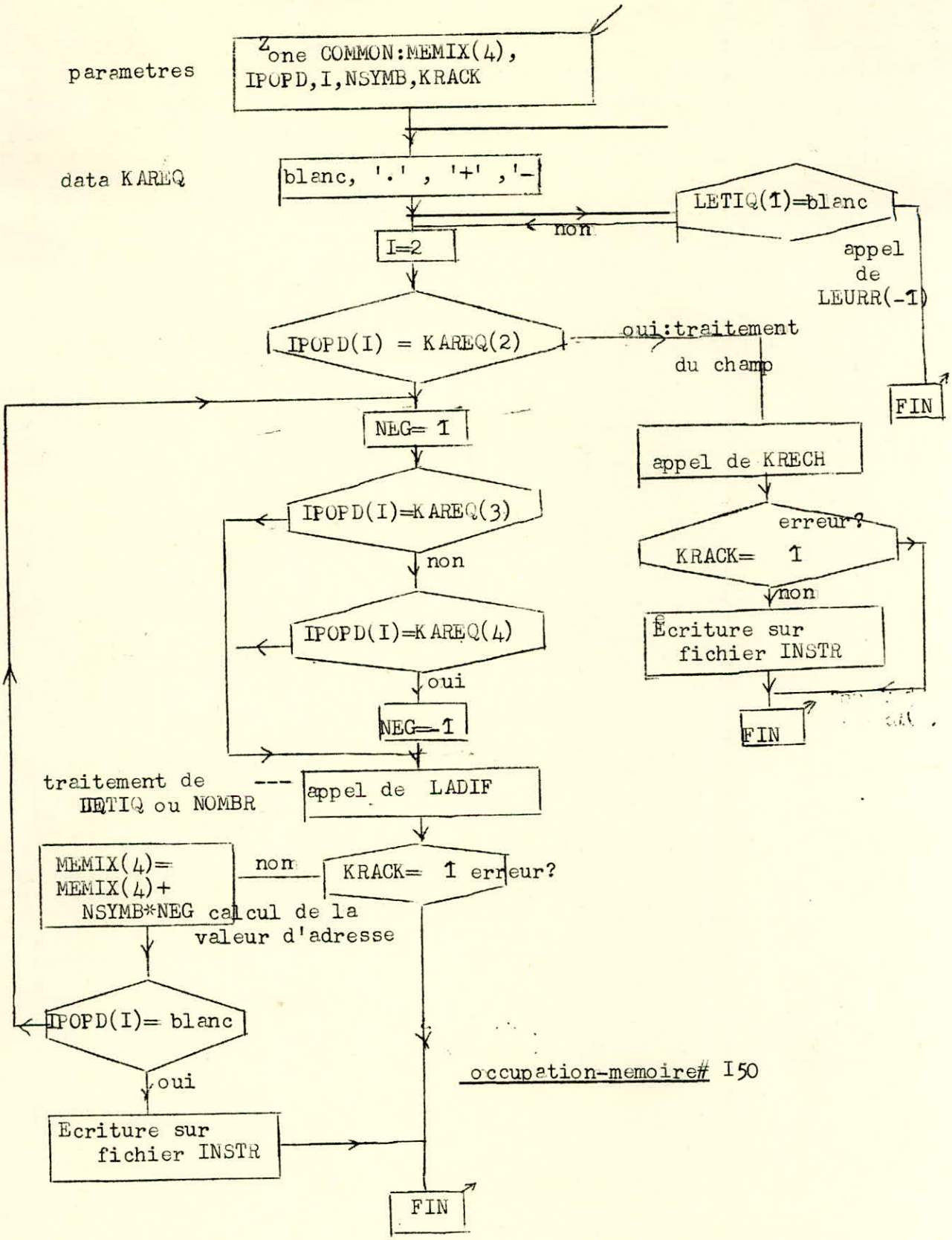
b - Syntaxe :

- Symbole EQU L.R
- Symbole EQU Nombre
- Symbole EQU Symbole + nombre

Le symbole est obligatoire en zone ETIQUETTE.

.....

c-ORGANIGRAMME:



- LORIG

a - But : Traitement de la pseudo-instruction ORIG.ORIG assure 2 fonctions principales l'une excluant l'autre. Elle sert :

- soit à imposer une origine au programme MIX ;
- Soit à réserver de la place mémoire (rôle du BSS en Assembleur 1130).

b - Syntaxe :

- Dans le cas où elle impose une origine nous aurons :

  ORIG entier sans signe - La zone ETIQUETTE doit rester rester vide ;

- Dans le cas où elle réserve de la place mémoire (définition de table) nous aurons :

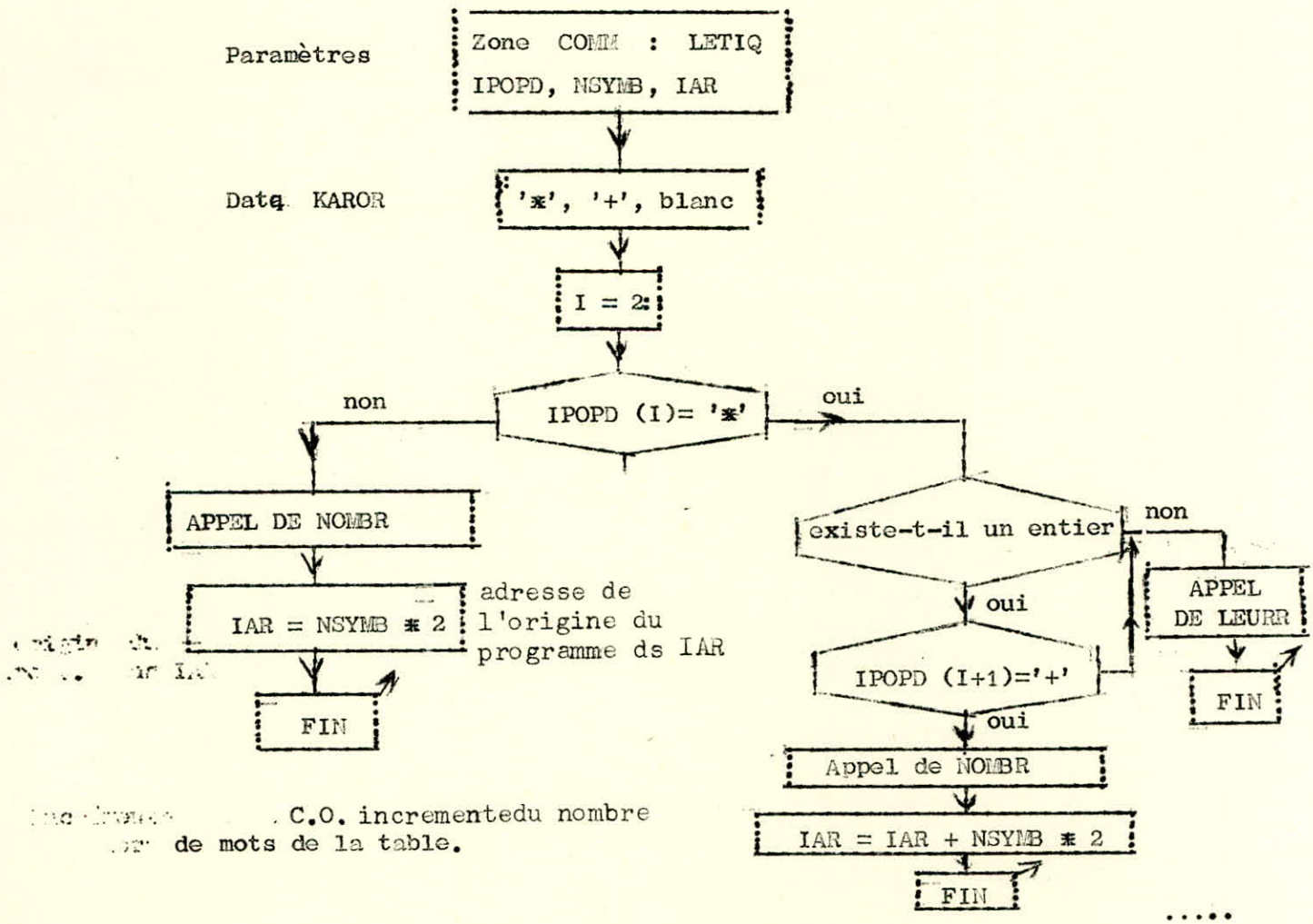
  TABLE 100 ORIG * + entier

  . La zone ETIQUETTE doit contenir obligatoirement un symbole ;

  . En zone OPERANDE l'Astérix est obligatoire. L'entier indique le nombre de mots MIX que doit contenir la table

*occupation-memoire# I00

c - Organigramme :



.....

- KACON :

a- BUT : traitement de la pseudo- instruction CON, dont restreignons le rôle à la définition de constante numérique.

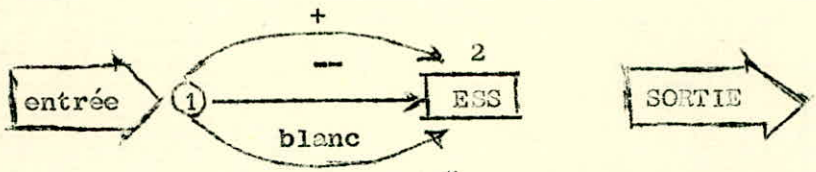
b- SYNTAXE :

Symbole u u u u CON u u nombre.

1 2

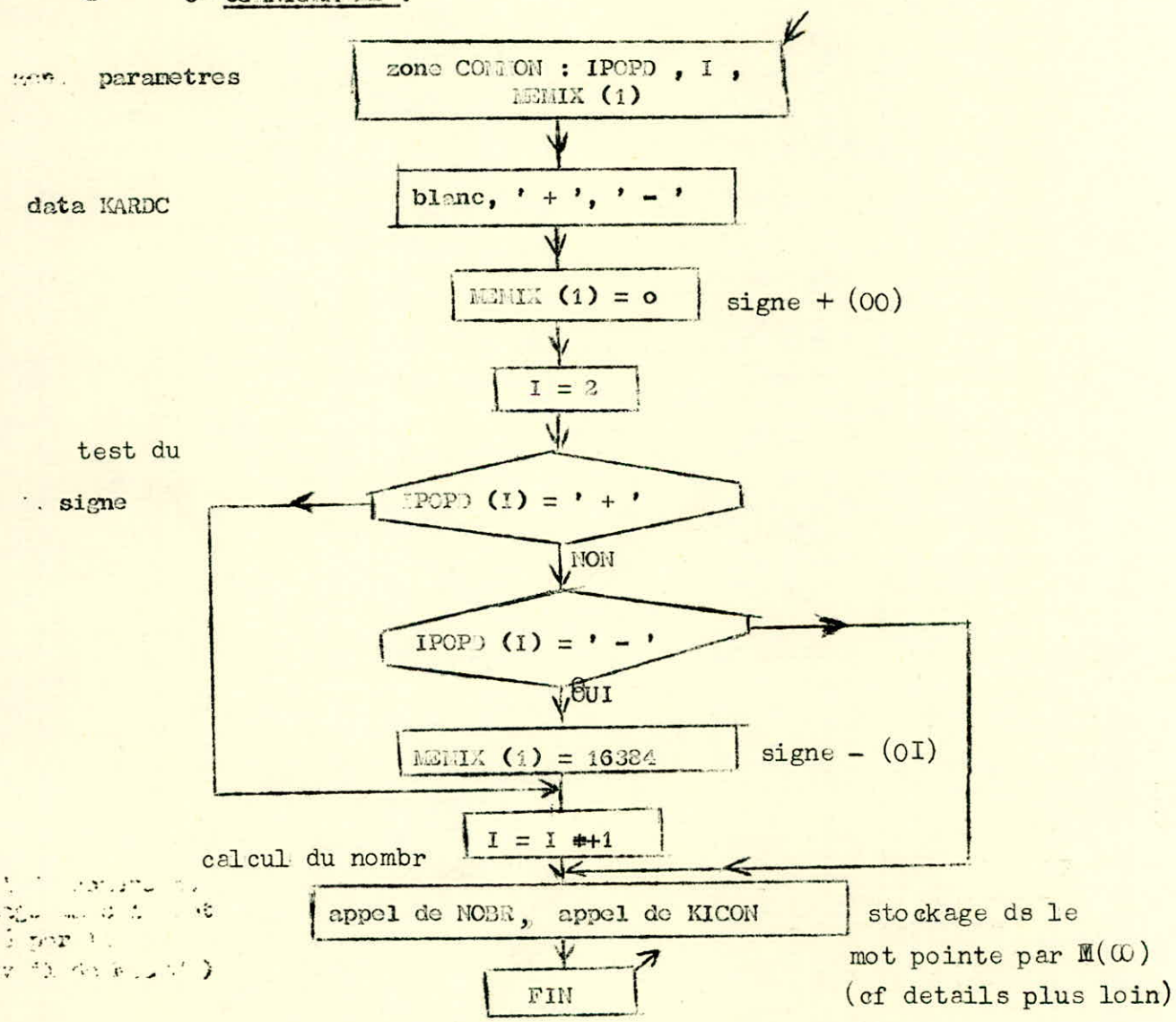
zone 1 : etiquette ou vide

zone 2 : entier avec signe (EAS) ou sans signe (ESS).



* occupation-memoire# 80

- c- ORGANIGRAMME :



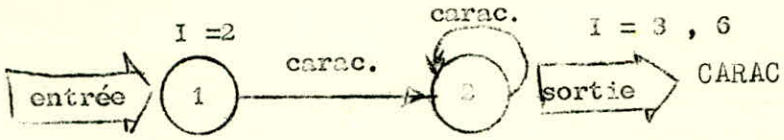
- KHALF :

a- BUT : traitement de la pseudo-instruction ALF qui sert à définir une chaine de caractères (alphabètiques, numeriques et speciaux).

b- SYNTAXE : Symbole u u u u ALF u u Chaine de caracteres.
1 2

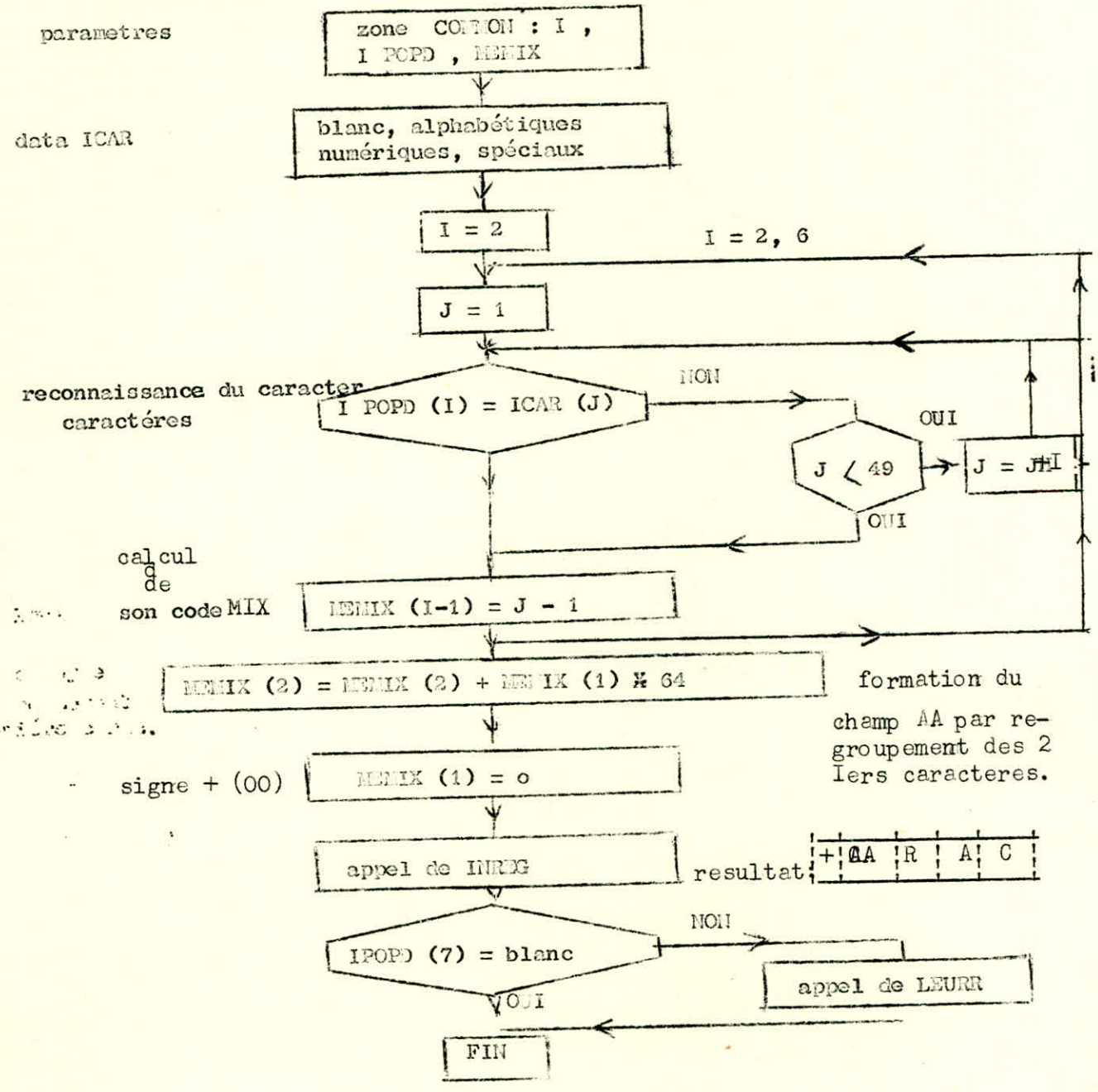
zone 1 : doit contenir un symbole ou blanc.
zone 2 : chaine de 5 caracteres à ranger dans le mot MIX dont l' adresse est indiquée par compteur ordinal.
Ce mot doit avoir le signe positif.
Le rangement se fera grace à l' appel du module de base INREG.

c- AUTOMATE DE RECONNAISSANCE



d- O R G A N I G R A M M E :

*occupation-memoire# I50



- KICON :

Le role de ce module est de stocker la constante définie dans le mot pointé par l' IAR. pour cela nous transmettons l' adresse de l' IAR et l' adresse du mot ou se trouve la constante.

KICON a été écrit en assembleur 1130, ce langage nous permet d' acceder facilement a ces 2 adresses. KICON ne fait en définitive que le transfert de la constante d' une adresse a l' autre, ou elle est cadree a droite.

*Occupation-memoire# I5

REMARQUES :

Nous avons défini dans le module MPAS1 2 compteurs.

- KOUNT : donnera le nbre de cartes du programme. IL sera incrementé à chaque lecture.
- IAR : C' est le compteur ordinal. Il reserve de la place memoire dans la table des instructions. Il sera incrementé chaque fois qu' on rencontre une instruction exection faite des pseudo-instructions EQU et ORIG lorsque cette derniére impose une origine.

-le code MIX des caracteres ,utilise dans KHALF,est le suivant:

blanc	'A'	''	'I'	'&'	'J'	''	'R'	'&'	'S'	''	'Z'	:alphabetiques
00	0I		09	I0	II		I9	20	2I	22		29																

0'	''	'9	:numeriques	
30		39		

.'	,'	('	)'	+'	-'	*'	/'	='	:speciaux
40	4I	42	43	44	45	46	47	48	

Les caracteres de codes :10,20 et 2I sont chez KNUTH les caracteres grecs:
nous les avons remplaces par le caractere & (ET), sous la forme duquel le compilateur FORTRAN imprime tout caractere non valide.

2eme PASSAGE: Module MPAS2

Boite I: Traitement de la passe 2

```
-Lecture d'enregistrement a partir du fichier I (INSTR)
-IF (carte commentaire ) THEN
                    passer a l'enregistrement suivant
                         ELSE
                    IF (POPR non HLT) THEN
                                Traitement de l'enregistrement
                    END
END
```

Boite 2: Traitementt de l'enregistrement

```
-IF (non PSIN ) THEN    DO;
          -Traitement de l'operateur(appel de KODOP)
          -Traitement de l'operande( appel de KOPER)
          -Mise des codes dans la zone reservee du fichier I
           en prevision du Listing
          -Regroupement de l'instruction(appel de INREG)
                         END
          -passer a l'enregistrement suivant
END
```

POPR=Identificateur de l'operateur

PSIN=Identificateur de pseudo-instruction(ici EQU,ORIG,CON,ALF)

Boite 3:Traitement de l'operateur:Module KODOP

```
-Recherche de POPR dans le fichier des codes(4,MOPIX)
-IF (POPR existe) THEN DO;
            -Mettre la valeur du code dans MEMIX(5)
            -Mettre la valeur de la specification de champ
             s'il y a lieu dans MEMIX(4)
                      END
                ELSE
            -Erreur(appel de LEURR)
END
```

Boite 4: Traitement de l'operande: Module KOPER

```
-Reconnaissance des delimiteurs
(  'b','.',',','(',')','+','-','*', /  '='  )

et conformement  a la syntaxe gene rale de l'operande

(OPD):=(AA)/(AA)(INDX)/(AA)(F)/(AA)(INDX)(F)    (cf details)

 appeler:LITEX ou
         LADIF pour le traitement de l'AAdresse
                                et de l'INDeX

         KRECH ou LADIF pour le traitement
                                   du   Field(champ)
-Sortie lorsqu'on reconnait le blanc de separation de l'operande
 et des commentaires
```

Detail de la PASSE 2

La zone reservee dans le fichier(I,INSTR) en prevision du Listing comprend la partie "instructions codees" qui se compose de 5 mots (MEMIX(J),J=I,5).

Le contenu de ces mots est donc determine lors du 2eme passage, apres la reconnaissance de la zone operateur(KODOP) et de la zone operande(KOPER), puis stocke dans la table des instructions (TINST(I00))**a l'aide du module INREG.

La table TINST, de 200 mots II30, soit de I00 mots MIX, peut donc recevoir I00 instructions codees, ce qui permet d'accepter des programmes Mix assez importants.

*Occupation-memoire# 230

Module KODOP:

Les operateurs MIX sont repartis en classes de la maniere suivante:

- le NOP; MEMIX(5) ou KOD=0, MEMIX(4) ou F=0;
- les operateurs arithmetiques, limites au traitement des nombres entiers: ADD; KOD=I, F=(0.5): I byte(SIGN) à 5 bytes(I mot MIX);

  SUB; KOD=2, F=(0.5);

  MUL; KOD=3, F=(0.5);

  DIV; KOD=4, F=(0,5);
- les operateurs speciaux; KOD=5:

  NUM; conversion code carac-code num.; F=0;

  CHAR; " " num - " carac.; F=I;

  HLT; fin du programme executable; F=2;
- les operateurs de mouvement:

  *decaler AA bytes: KOD = 6;

  SLA; decaler a gauche l'ACC ; F=0;

  SRA; " a droite " ; F=I;

  SLAX; " a gauche l'ACCEXT; F=2;

  SRAX; " a droite " ; F=3;

  [illegible] circulaire [illegible]

  [illegible] exit; F=5;

SLC;decalage circulaire de l'ACCEXT
a gauche;F=4
SRC;""" a droite;F=5
*transfert de donnees :KOD=7;
MOVE;transferer F mots de AA a(RII);

-les operateurs de chargement:KOD=8,...,23;F=(0.5);
LDA;
LDI,I=I,6;
LDX;
LDAN;
LDIN,I=I,6;
LDXN;

-les operateurs de stockage: KOD=24,...,33;F=(0.5);
STA;
STI,I=I,6;
STX;
STJ;
STZ;

-les operateurs d'Entree/Sortie:F=Nro d'unite peripherique
JBUS;unite F prete?(interruption);KOD=34;
IOC;controle de l'unite F ; KOD=35;
IN;lecture par " ; KOD=36;
OUT;sortie-impression F ; KOD=37;
JRED;quasi-similitude avec JBUS;
sera simule en DUMPAGE; KOD=38;

-les operateurs de Sauts:
JMP,JSJ,JOV,JNOV,JL,JE,JG,JGE,JNE,JLE;
KOD=39;F=0,...,9;
JAN,JAZ,JAP,JANN,JANZ,JANP;
KOD=40;F=0,...,5;
JIN,JIZ,JIP,JINN,JINZ,JINP;
KOD=41;F=0,...,5;
J2N,J2Z,J2P,J2NN,J2NZ,J2NP;
KOD=42;F=0,...,5;

J3N, J3Z, J3P, J3NN, J3NZ, J3NP;
KOD=43; F=0,...,5;
J4N, J4Z, J4P, J4NN, J4NZ, J4NP;
KOD=44; F=0,...,5;
J5N, J5Z, J5P, J5NN, J5NZ, J5NP;
KOD=45; F=0,...,5;
J6N, J6Z, J6P, J6NN, J6NZ, J6NP;
KOD=46; F=0,...,5;
JXN, JXZ, JXP, JXNN, JXNZ, JXNP;
KOD=47; F=0,...,5;

-les operateurs de transfert d'adresse:

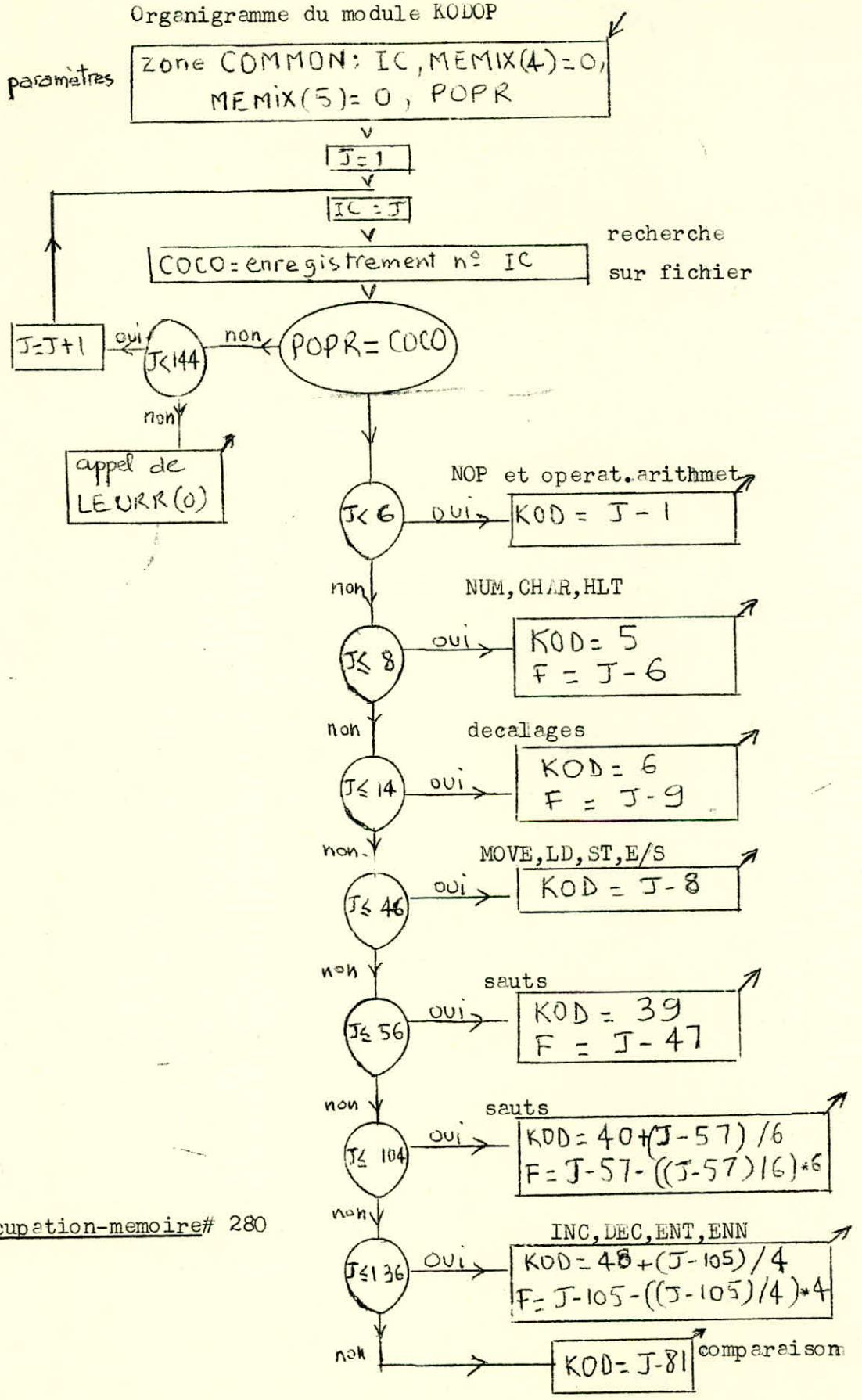
INC: incrementer le registre indique de AA(NOMBRE); F=0;
DEC: decrementer " " ; F=I;
ENT: mettre la valeur AA dans le registre indique; F=2;
ENN: " " (-AA) " " ; F=3;

si registre RA (ACC): KOD=48;
" RI(INDX): KOD=49;
" R2 " : KOD=50;
" R3 " : KOD=5I;
" R4 " : KOD=52;
" R5 " : KOD=53;
" R6 " : KOD=54;
" RX(EXT) : KOD=55;

-les operateurs de comparaison; F=(0..5); KOD=56,...,63;
limites au traitement des nombres entiers-
CMPA, CMPI, CMP2, CMP3, CMP4, CMP5, CMP6, CMPX .

Ce qui donne une collection de I44 operateurs, que nous avons mise sur disque : fichier(4,MOPIX); ainsi, la reconnaissance d'un operateur POPR se fait par parcours sequenciel du fichier. d'ou le calcul du KOD et du F, comme indique dans l'organigramme ci-joint.

Organigramme du module KODOP



*Occupation-memoire# 280

Remarque:

Cette methode de reconnaissance des operateurs MIX a ete adoptee apres essai d'un automate de reconnaissance caractere par caractere,pour ses avantages et de place-memoire et de temps;bien que le dernier avantage soit discutable au niveau de l'acces au disk.

Module KOPER: (*Occupation-memoire# 3I0)

La zone OPERANDE sera reconnue ,a la suite de POPR,Conformement a la syntaxe suivante,ecrite en notation de BACCUS:

(OPD):=(AA)/(AA)(INDX)/(AA)(F)/(AA)(INDX)(F)

(INDX):=(V)(NBRI)
(V):=,
(NBRI):=0/I/2/3/4/5/6

(F):=(PO)(LFR)(PF)
(PO):= (
(PF):=)
(LFR):=(L)(P)(R)/(ETIQ)/(NOMBR)
(P):= .
(L):=0/I/2/3/4/5
(R):=0/I/2/3/4/5 avec L .LE. R
(NOMBR):=(CH)(NOMBR)
(CH):= 0/I/2/3/4/5/6/7/8/9
(ETIQ):=(CH)(ALFANUM)/(LT)(ALFANUM)
(LT):=A/.../Z
(ALFANUM):=chaine de 9 caracteres alphanumeriques dont un au moins est une LT

IF (OPD):=(AA) THEN (AA):=(B)/(E) , (INDX):=0 , (F):=(0.5)
IF (OPD):=(AA)(INDX) THEN (AA):=(E) , (F):=(0.5)
IF (OPD):=(AA)(F) THEN (AA):=(E) ,(INDX):=0

(B):=blanc avec valeur nulle

(E):=(EX)/(S)(NOMBR)/(S)(ETIQ)/(S)(ETIQ)(S)(NOMBR)/(EQ)(EL)(EQ)

(S):=+/-

(EX):=(EA)/(AST)(S)(EA)

(EA):=(AST)/(NOMBR)/(ETIQ)

(AST):=*

IF (OPD):=(AA)(INDX)/(AA)(INDX)(F) et AA negatif THEN
il faut C(INDX) .GE. ABS(AA)

(EQ):==

(EL):=(EAS)(OB)(EL)/(ETAS)(OB)(EL)

(EAS):= entier avec signe:=(S)(NOMBR)/(NOMBR)

(ETAS):= etiquette avec signe:=(S)(ETIQ)/(ETIQ)

(OB):= +/-/*///.

addition binaire, soustraction binaire, multiplication binaire
division binaire, poitbinaire(L.R):=8*L+R
Priorite:de gauche a droite
ex:-3+4*62/-2 =+I*62/-2 = +62/-2 = -3I

Avant l'appel de KODOP Et KOPER, on initialise a Ola memoire MEMIX(J), J=I,5

A la sortie de KODOP, MEMIX(5) et eventuellement MEMIX(4) SONT connus.

A la sortie de KOPER, on obtient:

- -le signe de AA dans MEMIX(I);
- -la valeur calculee de AA dans MEMIX(2);
- -la valeur reconnue de l'INDX dans MEMIX(3);
- -eventuellement la valeur calculee de F dans MEMIX(4).

Alors, l'appel de INREG permet le regroupement de l'instruction et son chargement dans la table TINST a l'intention du Simulateur.

Remarque: Le module LITEX pour le traitement des expressions litterales(EL), dont les tests n'ont pas ete probants, n'a pas ete insere.

REALISATIONS PRATIQUES ET CONCLUSION

Les tests des programmes ont ete faits avec succes sur tous les modules, a l'exception du module general du 2eme Passage: MPAS2, et du module LITEX, devant traiter les expressions litterales.

Nous pourrons y remedier dés que le Centre de Calcul sera reapprovisionne en papier pour l'imprimante.

MIXAS pourra assembler des programmes presentes en MIXAL sous la forme suivante:

```
*Presentation triviale d'un programme en MIXAL
*Valeur d'adresse d'identificateurs
MIXALESTUN EQU  2.5
LANGAGE5   EQU  1009 C'EST LE NUMERO DE MIX(CHIFFRES ROMAINS)
BOLIQUE    EQU  1130
           ORIG 3946 ADRESSE DE CHARGEMENT IMPOSEE
*elle correspond a l'adresse du 1er mot MIX de la table TINT
START      .... ...  IERE INSTRUCTION EXECUTABLE
             .
             .
             .
EXIT       HLT   FIN DU PROGRAMME EXECUTABLE
*Definitions de constantes ,reservation de tables
PAIR       CON  2
FORMANT    ALF   ?  CONSTANTE ALPHANUMSPECIALE
OUINESSPA  ORIG *+20
             .
             .
           END   FIN DU PROGRAMME MIXAL
```

Ameliorations et Extensions:

-Traitement de l'operateur(module KODOP):

*au lieu d'utiliser-une table des operateurs caractere par caractere a raison d'un caractere par mot ,ce qui provoque l'occupation d'environ 350 mots de la MC;

-un fichier sur disque9(ici:4,MOT IX),ce qui entraine l'inconvenient oppose,en raison du temps d'acces Disque-MC;

*creer une table des TRONCS COMMUNS a tous les operateurs distribues en classes;chaque tronc commun ayant une taille maximale de 3 caracteres:

ex.: classe des operateurs de comparaison:

(TC):= CMP ;KOD= 56,...,63

caracteres differentiels:A , I , 2 , 3 , 4 , 5 , 6 , X

de reference aux registres concernes de la memoire MIX;

classe des operateurs de modification des registres:

(TC):= INC avec(F=0)/LEC(I)/ENT(2)/ENN(3)

4eme caractere:A avec KOD=48

I " " 49

.....

6 " " 54

X " " 55 ;

En choisissant des TC de 3 lettres au maximum,et en codant chaque lettre par son numero alphabetique(A-Z=== I-26) sur (en omettant Q) 5 bits,chaque TC occupera 1mot;il en est de meme pour les TC qui contiendront un chiffre de I a 6(INDX),car ces chiffres seront codes de 26 a 31=2**5-1.Pour rendre ce codage

possible,nous aurons omis la lettre Q,qui ne figure dans aucun operateur.On peut ainsi determiner 75 TC ,d'ou une table de 75 mots en MC,soit environ 4 fois moins que la Iere solution. (il faut encore y ajouter les caracteres de 4eme position: b,A,I,2,3,4,5,6,X,R,E,S,D,V,N,Z,P)

La reconnaissance de POPR (INTEGER POPR) se fera par recherche dans la table des TC:et c'est a ce niveau qu'on gagnera du temps,en procedant de la maniere suivante:au lieu d'une recherche sequentielle,qui amene a faire une quarantaine de consultations en moyenne,on utilisera la methode dichotomique (n consultations pour une table de 2**n mots,soit dans ce cas 7 consultations).Le temps de codage des 3 caracteres restant a evaluer,on peut cependant apprecier le gain considerable.

-Traitement des etiquettes au Ier Passage:

Dans MPASI,nous avons cree une table des symboles sur disque: a ce sujet ,on peut faire la meme critique que precedemment; on pourrait proceder,pour un gain de place au cas ou la table serait creee en MC,a un codage analogue au Ier.Une etiquette pouvant avoir au maximum IO caracteres,on codera de I a 36, donc chaque caractere sur 6 bits:la table des symboles sera structuree en enregistrements de 3 mots:2 pour le symbole et le 3eme pour sa valeur d'adresse;gain/format AI:2,5

-Traitement de L'operande litteral(module LITEX):

Le module LITEX devait calculer les expressions litterales lorsqu'elles sont numeriques;une amelioration pourrait se faire

selon le schema suivant:-reserver une table pour recevoir les litteraux a mesure qu'on les rencontre(apres calcul s'ils sont calculables);

-generer les instructions concernees avec dans la zone "operande" l'equivalent-adresse de la position du litteral dans la table.

-Traitement des nombres reels:

FADD: KOD=I ,F=6;

FSUB: KOD=2 ,F=6;

FMUL: KOD=3 ,F=6;

FDIV: KOD=4 ,F=6.

-Creation de MACRO-INSTRUCTIONS:

en groupant dans une seule procedure les instructions participant a un traitement determine(a ne pas confondre avec un sous-programme:une macro-instruction viendra se recopier a partir de l'endroit d'appel,entrainant un decalage du reste du programme vers le bas).

Comme exemplespossibles de macros,le traitement des operations binaires:

	Corps de la MACRO:
C= A+B	LDA A; ADD B; STA C
C=A-B	LDA A; SUB B; STA C
C= A*B	LDA A; MUL B; STA C
C= A/B	LDA A; SRAX 5; DIV B; STA C
C= A//B	LDA A; ENTX 0; DIV B; STA C
C= A.B (ou A:B)	LDA A; MUL =8=; SLAX 5; ADD B; STA C

Note concernant les Sous-Programmes:

Dans le programme principal, l'adresse de retour est sauvegardee dans le registre RJ, l'appel de SP se faisant par l'instruction: JMP NOMDUSP

(Ex.:(KNUTH, vol. I, p. 187)

```
PP:                SP A:              SP B:              SP C:
 .                  .                  .                  .
 .          A       STJ FINA   B       STJ  FINB  C       STJ  FINC
JMP A               JMP B              JMP  C             JMP  A
 .                  .                  .                  .
 .          FINA    JMP *      FINB    JMP  *     FINC    JMP  *
```

Une etude approfondie de la notion de SP et de COROUTINES pourrait etre entreprise dans le contexte d'une amelioration.

ASSEMBLEUR EN I PASSAGE

Ce serait un changement dans la conception-meme du travail, qui permettrait cependant d'atteindre de meilleures performances(entre autres, traitement des etiquettes numeriques).

CONCLUSION

Nous souhaitons, pour conclure, que le dynamisme traditionnel du departement ECONOMIE aidant, MIX suscite u n reel interet chez les etudiants de la future 5eme annee, et que ces idees ne restent pas lettre morte , mais viennent enrichir effectivement le Centre de Calcul.

Dans ce sens, nous vous reportons au KNUTH(cf reference page suivante), et au document etabli par nos collegues du projet: SYSTEME MIX.

D O C U M E N T S C O N S U L T E S

*THE ART OF COMPUTER PROGRAMMING

Vol.I/Fundamental Algorithms,DONALD E.KNUTH,Stanford University

(Addison-Wesley Publishing Company)1969

Cote-Bibliotheque:518.5 KNU t.I a;

*Documentation IBM du Centre de Calcul;

*Cours d'INFORMATIQUE,

Les Systemes d'Exploitation des Calculateurs,MICHEL ADIBA,ENPA;

*Comptes rendus des reunions de travail avec F.D.ARMINGAUD.