EUR 2583.e

EUROPEAN ATOMIC ENERGY COMMUNITY - EURATOM

THE COMPUTER PROGRAMS OF THE "SLC" SYSTEM FOR MACHINE TRANSLATION

by

S. PERSCHKE

1965

Joint Nuclear Research Center
Ispra Establishment - Italy

Scientific Data Processing Center - CETIS

LEGAL NOTICE

This report is on sale at the addresses listed on cover page 4

at the price of FF 7,—	FB 70	DM 5.60	Lit. 870	Fl. 5.10

When ordering, please quote the EUR number and the title, which are indicated on the cover of each report.

Printed by L. Vanmelle, S.A. - Gent
Brussels, December 1965

EUR 2583.e

THE COMPUTER PROGRAMS OF THE « SLC » SYSTEM FOR MACHINE TRANSLATION by S. PERSCHKE

European Atomic Energy Community — EURATOM
Joint Nuclear Research Center - Ispra Establishment (Italy)
Scientific Data Processing Center - CETIS
Brussels, December 1965 - 52 Pages - FB 70

The computer programmes of the Russian-English machine translation system used at Euratom are described.

The System has been developed at the Georgetown University, Washington, and programming system, written by A.F.R. Brown, has been called « SLC » - Simulated Linguistic Computer. The procedure of linguistic data processing, and the absolute form of the symbolically coded data are described for the use of a systems programmer.

The principal routines have been represented in flow charts.

EUR 2583.e

THE COMPUTER PROGRAMS OF THE « SLC » SYSTEM FOR MACHINE TRANSLATION by S. PERSCHKE

European Atomic Energy Community — EURATOM
Joint Nuclear Research Center - Ispra Establishment (Italy)
Scientific Data Processing Center - CETIS
Brussels, December 1965 - 52 Pages - FB 70

The computer programmes of the Russian-English machine translation system used at Euratom are described.

The System has been developed at the Georgetown University, Washington, and programming system, written by A.F.R. Brown, has been called « SLC » - Simulated Linguistic Computer. The procedure of linguistic data processing, and the absolute form of the symbolically coded data are described for the use of a systems programmer.

The principal routines have been represented in flow charts.

EUR 2583.e

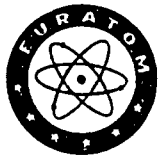
EUROPEAN ATOMIC ENERGY COMMUNITY - EURATOM

THE COMPUTER PROGRAMS OF THE "SLC" SYSTEM FOR MACHINE TRANSLATION

by

S. PERSCHKE

1965



Joint Nuclear Research Center
Ispra Establishment - Italy

Scientific Data Processing Center - CETIS

SUMMARY

The computer programmes of the Russian-English machine translation system used at Euratom are described.

The System has been developed at the Georgetown University, Washington, and programming system, written by A.F.R. Brown, has been called « SLC » - Simulated Linguistic Computer. The procedure of linguistic data processing, and the absolute form of the symbolically coded data are described for the use of a systems programmer.

The principal routines have been represented in flow charts.

Contents

Introduction

The present report is supposed to close a gap in the documentation of the Russian-English machine translation system developed by the Georgetown University and made available to Euratom through a research contract (Contract No. 033-63-9 CETU).

The theoretical approach and the linguistic analysis carried out for the realisation of the system are described in the Occasional Papers of the Georgetown University Machine Translation Project, and the General Report 1952-1963 (Paper No. 30, June 1963) summarises the research carried out. Some of the Occasional Papers of the later period will be published as Euratom Reports.

The needs of the users of the machine translation system are satisfied by "The 'SLC' Programming Language and System for Machine Translation" by A.F.R. Brown, Euratom Report EUR 2418 e, 1965,which is a programming manual for the symbolic programming language in which the dictionary and the linguistic classification and operations are coded, and the use of the utility programmes, which make it possible to make and to maintain the translation system, and to update or change the machine dictionary and the linguistic operations.

However, there is no description of the machine programmes which carry out the machine translation process, nor of the techniques used for input - output, the dictionary look-up and the execution of the symbolically coded linguistic operations.

Thus, there was no information available for the systems programmer who would modify or enlarge the system (for instance introduce some new SLC function for a particular use) or adapt the system for some supervisor (such as the IBSYS monitor) or for another computer.

The description of the programmes is based essentially on the assembly listings (all IBM 7090 programmes are written in

Manuscript received on October 6, 1965

FAP), and on some information obtained from the author of the system, A.F.R. Brown, who worked for a certain period at the Joint Data Processing Center of EURATOM (CETIS).

The Georgetown Translation System, in its actual form is independent of any supervisor or monitor system. The system is recorded on one magnetic tape which contains four files:

File 1: The Dictionary look-up programme
File 2: The Russian-English dictionary in absolute form
File 3: The Translation programme
File 4: The symbolically coded independent operations in absolute form.

All machine programmes are written for the IBM 7090 computer in the symbolic programming language FAP. Some of the programmes originally were written for the IBM 704 , and one can notice some residue of these programmes, such as the scrambling of the octally coded data in the Symbolic Assembly Programme.

For creating and updating the translation system three more 7090 programmes are needed: The symbolic assembly programme which transforms the symbolically coded dictionary entries and operations into an absolute form, the dictionary creating and updating programme, which sorts the assembled dictionary entries alphabetically and merges them with the existing dictionary (if there is any), and the Assemble Operations programme, which organises the assembled Independent Operations as one binary file on the system tape.

For programme control and the different optional features the 36 keys of the computer console are largely used.

The use of an isolated system without any monitor control, and the necessity of setting a series of keys and switches before each run have proved disadvantageous , and it seems desireble to adapt the system for some supervisor control, such as the IBSYS monitor, or the FORTRAN monitor. Especially the

tape assignments should be made compatible with the current monitor convention.

The present report contains the description of the two basic programmes of the system: the dictionary look-up programme, and the translation programme. The attention has been focussed to the programme parts which are essential for the procedure of language processing. Marginal features as on-line read and print conversion routines have been neglected.

Apart from the procedure itself, also the form of the system's data has been largely described, such as the dictionary in absolute form, the organisation of the pseudo entries, the organisation of the operation file etc. The description of data in the absolute form, which are supplied by the different housekeeping programmes should also be a certain ersatz for another report: the description of these programmes from the point of view of the systems programmer.

1. The Dictionary Look-Up Program

This programme which is the first file on the system tape reads in a batch of input text (approximatively 2.ooo words), and looks for the corresponding dictionary entries. The programme loads itself in core storage by pressing or simulating LOAD TAPE. Its main functions are:

1. Reading the pseudo entries and locating them as reference tables in core storage.
2. Reading the input text, recognising the word boundaries, eventual segmentation from the left hand end and locating them in core storage.
3. Sorting the input batch in alphabetical order.
4. Looking up the words in the dictionary, morphological analysis, locating the matched entries in core storage, and eventual analysis of the words not found in dictionary.
   (see diagramme 1)

1.1. Loading and Initialisation

The programme is located into core storage by pressing or simulating the Load Tape button of the console:

```
(       RTBA    1
        RCHA    * + 2
        LCHA    0
        TRA     1
        IOCT    0,,3        )
```

Tape loading is checked for errors. Computer halts at loc. 6 if no End Of File is detected, and after rewinding tape A1 at loc.13 if tape error is signalled.

1.2. Initialisation

1. The keys are loaded into the sense indicators.
2. If key 20 is ON, the program is prepared to write the Russian text on tapes B1 and B2 (see diagramme 2) alternatively per batch.
3. The pseudo entries are read in and located in the respective table (see diagram 3, 3a, 3b, 3c).
4. Set up convert tables ATAB and SPS
   ATAB converts binary zero into blank
   SPS converts $ into 1
   - into 2
   bl into 3
   77 into 4 (77 octal defines the end of the significative field on the input cards. This table is used for recognising word boundaries)
5. The TDEX table is set up continuously numbered from o to 3777 octal in bits 1 - 11

1.3. Subroutine RDIC

This subroutine is provided for two purposes:

1. In the initialisation phase, it reads in the pseudo entries and locates them in their respective storage zones. This function is terminated upon reading the first normal entry.
2. In the dictionary look-up phase it reads in the entries one by one.

Dictionary entries are read in alternatively in two buffers: DIX and FIX. At each entry in the subroutine the program checks whether the last block processed is finished, and, if so, reads the next block. It defines the start and the final address of the entry, and displays the information of the title word of the entry in different registers (see table 1). If the entry is

a pseudoentry, it is located in the respective storage zone, and the program reads in the next entry.
When the first normal entry is recognized (the first word of the Russian stem is not ooooo1 or ooooo2) the test for pseudo-entries is deleted and the normal return (TRA 2,4) is taken.

If during reading the dictionary an end of file is met, program precludes further reading by storing a TRA 1,4 at the entry point of the subroutine, and takes the end of file exit TRA 1,4. This exit is error condition during reading the pseudo-entries. In the dictionary look-up phase it terminates the look-up, and is used at the end of the program for proper positioning of the system tape.
(See diagrammes 3, 3a, 3b, 3c).

1.3.1. Dictionary Organisation

The dictionary is the second file on the system tape. The entries may be blocked to a mximum length of 500 machine words per physical record. An end of record gap must necessarily concide with the end of a dictionary entry.
Each entry is organised as follows:

1. The first word of each entry is a title word subdivided in 6 zones

a	b	c	d	e	f

a. Bit S - 1 (2 bits) Entry type:

- 00 : + (built in paradigm with valid nil ending)
- 01 : - (built in paradigm with no valid nil ending)
- 10 : $ (entry referring to standard paradigm)
- 11 : * (unsplit entry)

b. Bit 2 - 10 (9 bits) Total length of entry in machine words.

c. Bit 11 - 17 (7 bits) Number of machine words occupied by the English output.

d. Bit 18 - 24 (7 bits) Number of tailwords.

e. Bit 25 - 28 (4 bits) Length of Russian stem

f. Bit 29 - 35 (7 bits)
1. If $ type entry: paradigm number (2 octal digits)
2. If + or minus type entry: number of machine words occupied by the built in paradigm
3. If * type entry: zero

2. The words following (e words) are occupied by the Russian stem. The Russian stem is left-adjusted, and the remaining positions of the last machine word are occupied by binary zeros. The character zero is represented by octal 12 (oo1o1o)

E.g. The Russian stem QELESOOBRAZNOST is assembled as follows:

Q	E	L	E	S	Ø	word 1
Ø	B	R	A	Z	N	word 2
Ø	S	T	o	o	o	word 3

3. If the entry is of the + or - type, the words following are the built in paradigm. Each entry of the built in paradigm may be two or three words long, e.g.:

- E R Y X o	suffix	+ V W E G O	suffix word 1
212121212121	1st headword	S 4 o o o o	suffix word 2
		212121212121	1st headword

N.B. All octally coded data of the symbolic dictionary are recorded in a scrambled form on the dictionary tape, i.e.:
The 12 octal digits are read in BCD as 2 machine words:

o1o1o1o1o1o1		o2o2o2o2o2o2
word 1		word 1

Then the second word is shifted left 3 bits: 2o2o2o2o2o2o
and ored into the first word. The result is: 212121212121

If the entry is $ or * type, the Russian stem is immediately followed by the first headword (see 4. below).

4. First headword. (scrambled) If the entry is of the * type, it is definitive, otherwise it is ored with the first headword of the paradigm matched during dictionary look up.

5. Second headword. (scrambled) The second headword is organised as follows:

o	LLLLL	TTT	EEE

LLLLL is the lexical number of the entry
TTT is the number of tailwords
EEE is the number of English words

N.B. During dictionary look up the information of the second headword is saved, while the title word is lost.

6. The words following are the tailwords, if there are any (d ≠ o) There are the following types of tailwords distinguished by their characteristic:

1. Local operations. (TL in Symbolic dictionary entry)
The title word of a local operation is organised as follows:

4 o o	C C C	F F F	P P P	o o

CCC is the number of constants
FFF is the number of SLC coded functions
PPP is the priority number

2. English coded constants. (TW in Symbolic dictionary entry)
The title word contains 6oo6 as characteristic and the

length of the English constant, e.g.

6 o o 6	o 2	o o o o o o			
*	*	*	*	A	B
I	L	I	T	Y	b

3. Diacritics. (TD)
   The first 4 octal digits of a diacritic must be 6ooN, where N is a digit $\neq 6$ for distinction from English coded constants.

4. Independent instructions. (TI)
   Independent instructions are calls for instructions in the operations file, and are assembled in the absolute form of an instruction. ooooIIIPPPEE

7. The words following if any contain the English equivalent of the dictionary entry (c words)

1.3.1.1. Pseudoentries

Pseudoentries have the function of reference tables for morphological analysis (paradigms), segmentation of words (analysis of compound words), and grammatical analysis of words not found in the dictionary.

They are coded and assembled exactly in the form of normal dictionary entries with the only difference that the first 6 characters of the Russian stem (originally coded as 111111 or 111112 in symbolic form) are assembled as octal oooooooooooo1 and oooooooooooo2 and are all in front of the normal entries on tape.

There are four types of pseudoentries:

1. + and - type entries
   These entries refer to paradigms(with or without valid nil-suffix respectively, and are matched with the suffixes of $ type entries.
   The Russian stem of this pseudoentry is assembled form is:

o o o o o 1	3 6 o o o o
word 1	word 2

   The first two characters of the second word of the stem are the paradigm number.

2. * type pseudo entries:
   Suffixes for the analysis of words not found in dictionary. The Russian stem contains as first word +1. The first six bits of the second word contain the 12-complement of the length of the suffix (12 minus the number of characters of the suffix), i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B. Further it contains the first headword, one tailword which is normally an independent instruction and the assembled English equivalent of the suffix.

3. $ type pseudo entries are a list of detachable elements of compound words, originally intended for the analysis of chemical terms. In dictionary they are organised in the following way:
   The first word of the Russian stem is +1 if the entry may be either initial or medial in the compound word, and +2, if it is only medial. The following word(s) contains the 12-complement of the length of the segment and the segment. There is no further information. Segments must also appear as normal entries in the dictionary. A segment must not be longer then 9 characters.

1.3.1.2. Organisation of Pseudoentries in Core Storage

In the first phase of the dictionary look-up programme, the pseudo entries are read in and stored as tables in the memory.

1. Fixed paradigms
   There are two tables identifying the paradigms:

   1. At location SFDEX to SFDEX + 63 (5214 - 5313) there is a reference number for each paradigm. At location SFDEX + Par. Nr. the location and the length of the paradigm are stored. The index word for a + type paradigm is set up as follows:

MZE	b		a + b

   where a is the starting address of the paradigm, and b the length. The paradigm goes from a to a+b.

   For - type entries the index word is as follows:

PZE	b		a + b

   The paradigm goes from a to a+b-1 since there is no first headword for a nil suffix.

   2. At location SFS to SFS + 1935 (5314 - 11134) the paradigm itself is stored. (There are 1936 machine words available for fixed paradigms).
   The paradigms are organised as follows:

a. + type:

- O M U o o
Headword 1
+ U H 5 E G
O S 4 o o o
Headword 1
Headword 1

(of the nil suffix)

b. The - type paradigms finish with the first headword of the last suffix. There is no first headword for the nil suffix.

N.B. + and - in the paradigms are characters (octal 20 and 40)

2. $ type pseudo entries

The word segments are stored at location CF to CF + 499 (13104 - 14067). Each pseudoentry occupies 2 machine words, so that as many as 250 detachable prefixes can be provided for. A segment must not be longer than 9 characters. The entries are left adjusted, and the second word contains in its address part the length of the segment in bits. E.G.

G I D R A Z	initial or medial
I N o 111111	
PZE	
MZE	
G L I Q E R	medial only
I N o 11111	

3. Termination of unknown entries

Stored at TER to TER + 999 (1000 words --- 166 entries)
Each entry is right adjusted and contains in the first 6 bits of the first word the length of the suffix. e.g.:

4 o o o o o
o o E T S 4
Headword 1
Tailword
* * * * * *
* E T S 4

Each entry occupies 6 machine words. The tailword is normally an independent instruction. The English equivalent may be either the repetition of the suffix itself or some English suffix, which can be useful for the translation of common technical terms. e.g. Russian PERIODI-CESKI, not found in dictionary could be translated PERIODI-CALLY by an appropriate pseudo entry.

1.3.1.3. Display of the Title Word of the Dictionary Entry in Storage

DIXA	414	Start address of entry
DIXB	415	Start address + length of entry read in
SEG	416	Title word
CHAR	422	Entry tape (a)
LGTH	426	Entry length (b)
EEE	417	Number of English words (c)
TTT	42o	Number of tailwords (d)
TIT	421	Number of Russian words (e)
STIT	423	Length of paradigm or paradigm number (f)
SEG+6	424	Start address of dictionary entry

1.4. Subroutine RDC

1.4.1. Reading and Locating the Input Text

This subroutine reads and locates the input text in core storage. Up to 2o48 different words or 5ooo machine words may be occupied by the input text. The main function of this routine is to recognise the word limits according to the punching conventions. Each card is read in from the card reader or tape, and the blank field at the end of card is signed by octal 6o77. If the card is all blank, the next card is read in. Then the

card is copied on tape B1 or B2 if key 1 is on. If End of File is encountered during reading, the rest of the text is translated, and then the machine waits for a new input tape to be mounted, or new cards put in the card reader. The wait loop may be interrupted by setting sense switch 1.

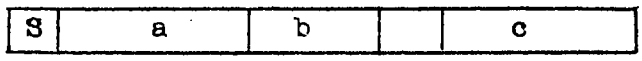
Each card is tested for word boundary characters (blank, hyphen, Dollar sign). The text words are stored left adjusted in the core storage. If key 18 is ON the leftsplitting subroutine is executed (see SLC manual p.144 , p.21 of this report) Each separate text word, also those resulting from alternatives of reading (hyphen and left splitting) is given a proper index word in TDEX.

When a input batch is full (less then 128 positions free after a sentence end, end of file exit etc.), the programme passes to the sort routine.
(See: diagrammes 4 - 4c)

1.4.2. Storage of Input Text After Reading

For each word recognised through the punching conventions (see SLC manual p. 163) one index word is provided at location TDEX to TDEX + 2047 (62400 - 66377).

Each Index word is organised as follows:



a.: current number of the word
b.: length of the word in machine words
c.: starting address of the word

The text is continuously stored at location TEXT (2o.o7o) and each word is left adjusted. Alternatives of reading (hyphens, left splitting etc.) are stored as different words. The rules

for defining word boundaries and alternative solutions are described in the SLC manual p. 163)

The end of the significative field of the TDEX table is defined by a word containing MZE in the prefix.

1.4.3. Storage of Russian Words for Monitoring

Location RUSSTO to RUSSTO + 4095 (66400 - 76377)

Russian words are stored up to 12 characters, left adjusted. The word is completed to 12 characters with blanks. If the word is longer then 12 characters, the first 6 and the last 5 are taken. The seventh character is an apostrophe

e.g. KAPITALISTICESKI1 becomes KAPITA'ESKI1

1.4.4. Segmentation of Words from the Left Hand End

The beginning of the word read in is compared with the pseudo entries stored at CF. If a pseudoentry is matched, it is stored in TEXT separately, the rest of the word is left adjusted, and compared again with the pseudo entries. A word must be at least 6 characters long for permitting this routine. For programming reasons, maximum length has been limited to 13 machine words, i.e. 78 characters.

The two types of pseudoentries: initial or medial, and medial only are separated from each other in core storage by an entry containing PZE and MZE. The medials only are tested only after a successful match. This is indicated by sense light 4. Between each segment the link word ++++++ is stored. The segments and the complete word are considered separate items by the program, and a symbolic SLC routine must be provided for choosing between the unsplit and the split variant, and setting up one only item.
(see SLC manual p. 147)

1.5. Sorting the Dictionary Entries

For machine convenience, only the index words of the text are sorted, and the order of the text words remains unchanged. The standard sorting procedure by comparing groups of powers of 2 is applied. The sequence adapted is the absolute octal value of the characters. However, blank corresponds to octal 00 and zero to octal 12 .

Two words are taken up through their index words and compared, and the index word of the smaller value is stored in front of the grater value. If two words are equal, the index word is set minus, the first index is stored, and the second is compared with the next word.

Sort is finished, when the sum of the two groups to be compared covers the entire input batch.
(See diagramme 5)

Example of Text storage before and after sort

TEXT

oo	P	R	I	o	o	o	o
o1	3	T	O	M	o	o	1
o2	V	O	Z	M	O	J	2
	N	O	o	o	o	o	3
o3	K	V	A	N	T	O	4
	V	O	E	o	o	o	5
o4	O	P	I	S	A	N	6
	I	E	o	o	o	o	7
o5	V	S	1	o	o	o	1o
o6	P	L	A	Z	M	Y	11
o7	I	o	o	o	o	o	12
1o	P	O	S	L	D	N	13
	I	1	o	o	o	o	14
11	U	C	E	T	o	o	15
12	E	E	o	o	o	o	16
13	K	O	R	R	E	L	17
	4	Q	I	I	o	o	2o
14	S	o	o	o	o	o	21
15	A	T	O	M	O	M	22
16	*	o	o	o	o	o	23

TDEX before SORT

o	oooo o1	TEXT
1	ooo1 o1	TEXT+1
2	ooo2 o2	TEXT+2
3	ooo3 o2	TEXT+4
4	ooo4 o2	TEXT+6
5	ooo5 o1	TEXT+8
6	ooo6 o1	TEXT+9
7	ooo7 o1	TEXT+1o
1o	oo1o o2	TEXT+11
11	oo11 o2	TE
12	oo12 o1	
13	oo13 o	
14	oo	
15		

TDEX after SORT

	ooo1 o1	T+1
	oo15 o1	T+18
	oo12 o1	T+14
	ooo7 o1	T+1o
	oo13 o2	T+15
	ooo3 o2	T+4
	ooo4 o2	T+6
	ooo6 o1	T+9
	oo1o o2	T+11
	oooo o1	TEXT
	oo16 o1	T+19
	oo11 o1	T+13
	ooo2 o2	T+2
	ooo5 o1	T+8

1.6. The Dictionary Look Up Phase

The dictionary entries are read in one by one through the subroutine RDIC and compared with the text words.

In the case that a dictionary entry results greater than the current text word to be matched, it is tested for precedent matches. If there are, the test is considered definitive, if there are none, the word is analyzed, if desired through the Not In Dictionary subroutine NID.

If a word is greater than the dictionary entry, a test is made for a partial match, i.e.: up to 11 characters of the text word are detached as the suffix (if the dictionary entry to be matched is not an unsplit entry), and the rest is compared with the dictionary entry. If such a partial match has occurred, the suffix is looked up in the general paradigm, if the dictionary entry is of the $ type, or in the built-in paradigm, if it is of the + or - type.

The present morphological analysis procedure permits also the splitting of a suffix into its components, i.e. the suffix itself can be analyzed as a series of morphomes. This analysis is determined by the symbolical coding as, for instance, - UH5XX$37 . In this case the first three characters of the suffix must be UH5, and the rest must be found in paradigm No. 37. The number of references to other paradigms for the analysis of the rest of the suffix is not limited. However, the limitation of the entire suffix length to 11 characters remains valid. This procedure permits, especially for Russian, a more accurate definition of the morphological formants, and makes the system of the paradigms more concise. For instance, it was necessary before to repeat the entire declension of the participles in such paradigm of the verb on the reflexive and non reflexive form. Now, it is sufficient to code the formants of the participle e.g. - UH5 - with the reference to the declension type, e.g.

par. No. 37 which contains, for instance - EGO - plus the reference to a possible reflexive formant -S4. Thus, the Russian suffix -UH5EGØS4 is analysed as -UH5+EGØ+S4. The final grammatical inforamtion of this suffix is obtained by logical addition of the first headwords of the three morphemes.

For *, +, - type entries the first match is considered definitive, while all possible matches of $ type entries are accepted provisionally by the dictionary look-up programme. Through a key setting the linguist can decide whether to accept for good only the last entry matched which is the most probable (the longest stem and the shorter suffix), or to make further tests for resolving the look-up ambiguities. This is to be coded symbolically as an independent operation)

However it seems to be recommenable to create unsplit entries and the respective decision routines for cases of true morphological ambiguities: e.g. BEREGU that is Dat. or Loc. of BEREG, or 1st Pers. Sg. Pres. of BEREC6.

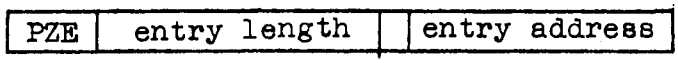
The index word of an entry matched definitively is set to zero, and in TDEX the address and the length of the entry are stored. In the case of multiple look-up the index word is stored in front of the entry itself at NSL.

If a word is not matched with any dictionary entry, depending on the key setting either the analysis routine NID is executed or the artificial entry for words not in dictionary is set up immediately.
(See diagramme 6)

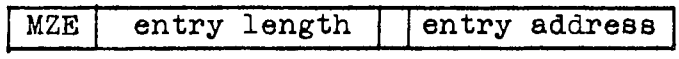
1.7. Storage of Dictionary Entries Matched with Text Words

During dictionary look up the dictionary entries matched with text words are stored downwords starting from location NSL (TDEX - 1), 62377. The TDEX table contains in its address part the address of the dictionary entry matched last with the

textword, and the decrement part the length of the entry. The location NSL itself contains the address of the first location not used by dictionary entries. Normally the prefix of the TDEX word is PZE. If a multiple look-up has taken place, the prefix is MZE.



Single look up



Multiple look up

In the case of multiple look up the TDEX word is stored in front of the first headword in the NSL block.

Storage map at dictionary look-up phase

ooooo	Dictionary look-up program actually 4162 locations location 4163 - 4227 are unused
o423o	DIX Dictionary buffer
o5214	SFDEX and SFS Fixed paradigms
11134	TER Termination of unknown entries
13104	CF Segments for left splitting
14o7o	NDEX Index table for sorting and dictionary look-up
2oo7o	TEXT Input text
62377	Storage of looked-up dictionary entries NSL
624oo	TDEX Index table of input text
664oo	RUSSTO Storage of Russian words for monitoring
764oo	FIX Dictionary buffer
77364	unused storage

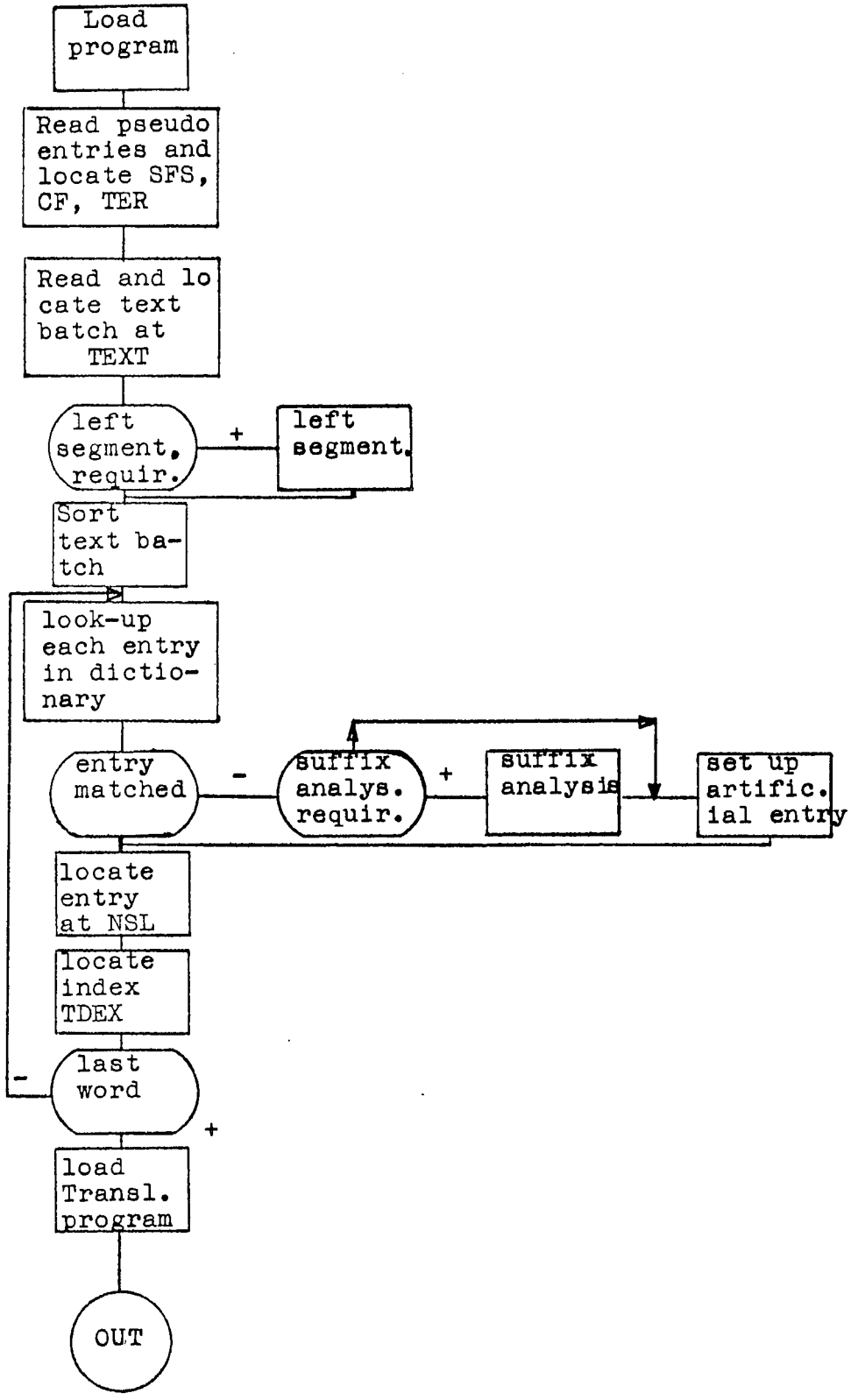


Diagramme 1: The general flow of the Dictionary Look-up Program

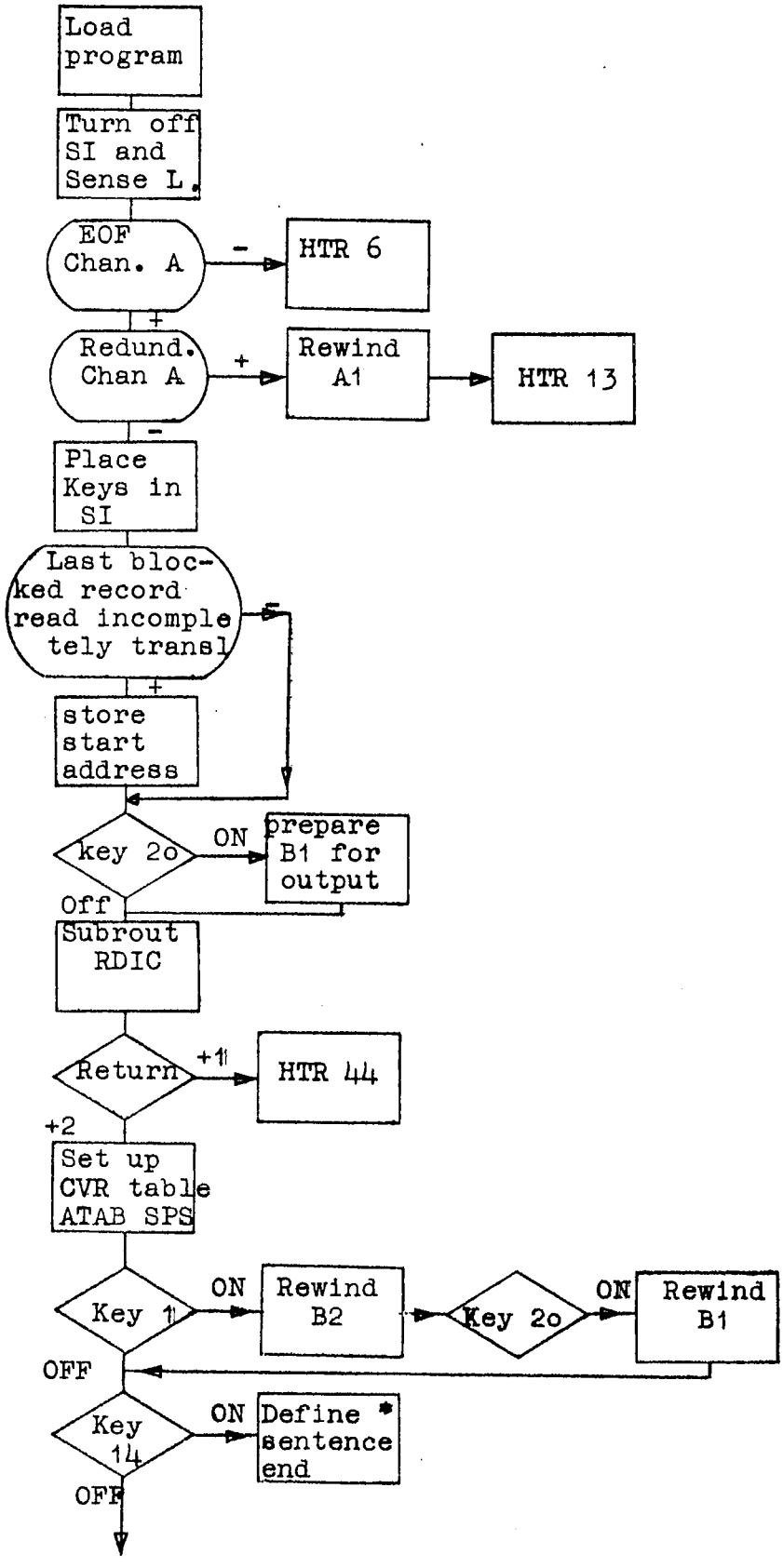


Diagramme 2 Loading and Initialisation

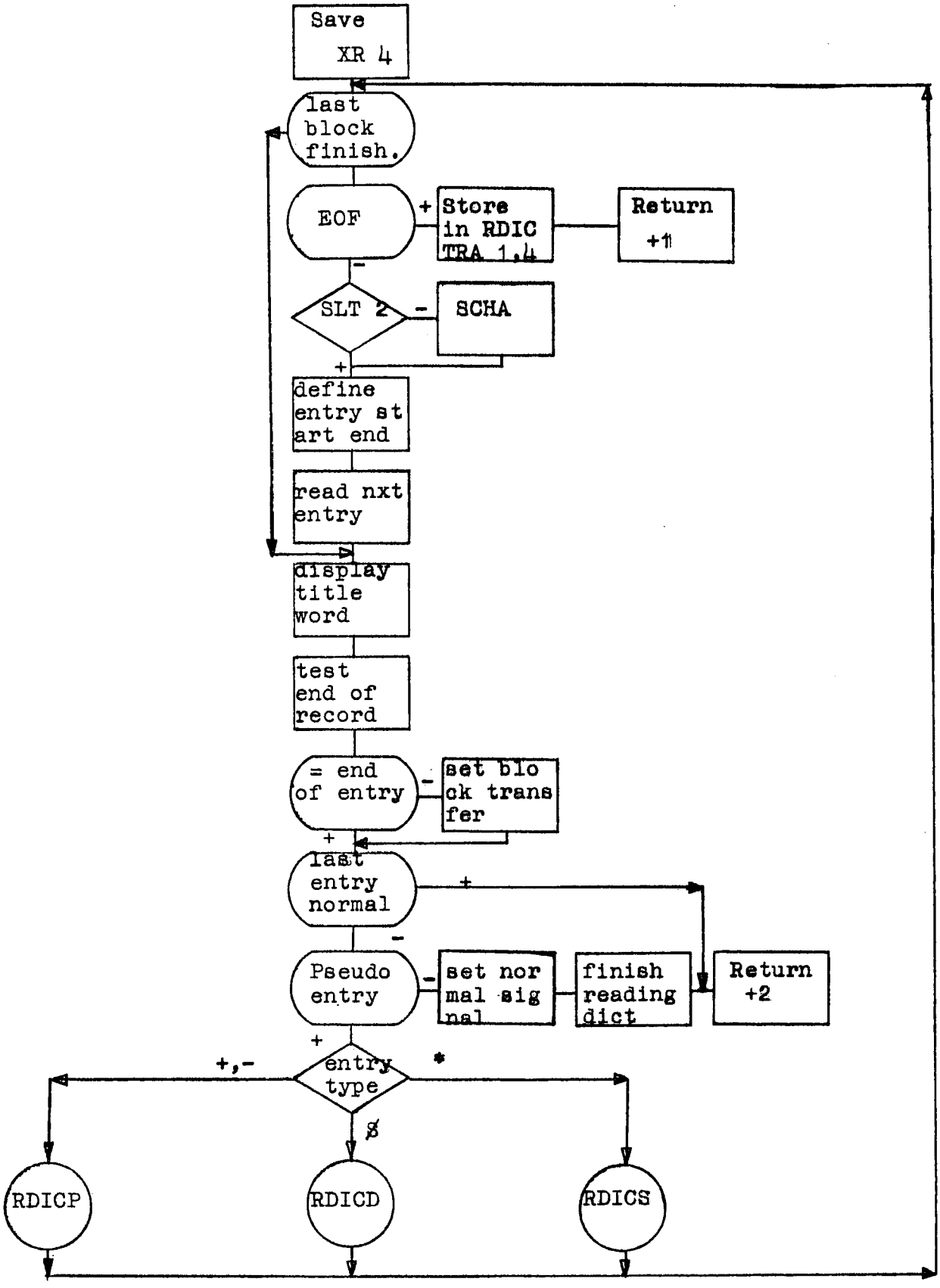


Diagramme 3 Subroutine RDIC: Read dictionary entries and locate pseudo entries

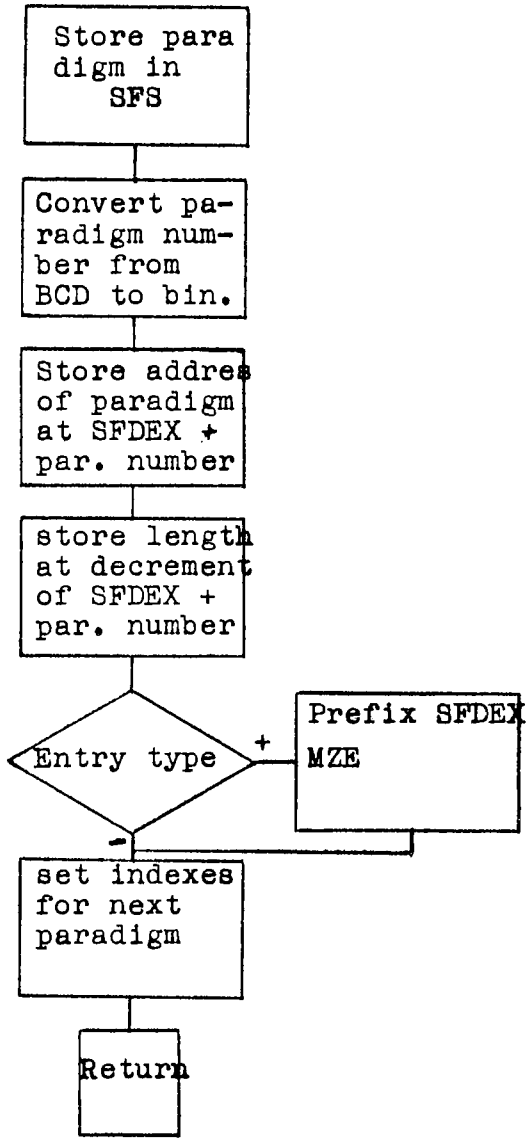


Diagramme 3a Subroutine RDICP: Storage of fixed paradigms

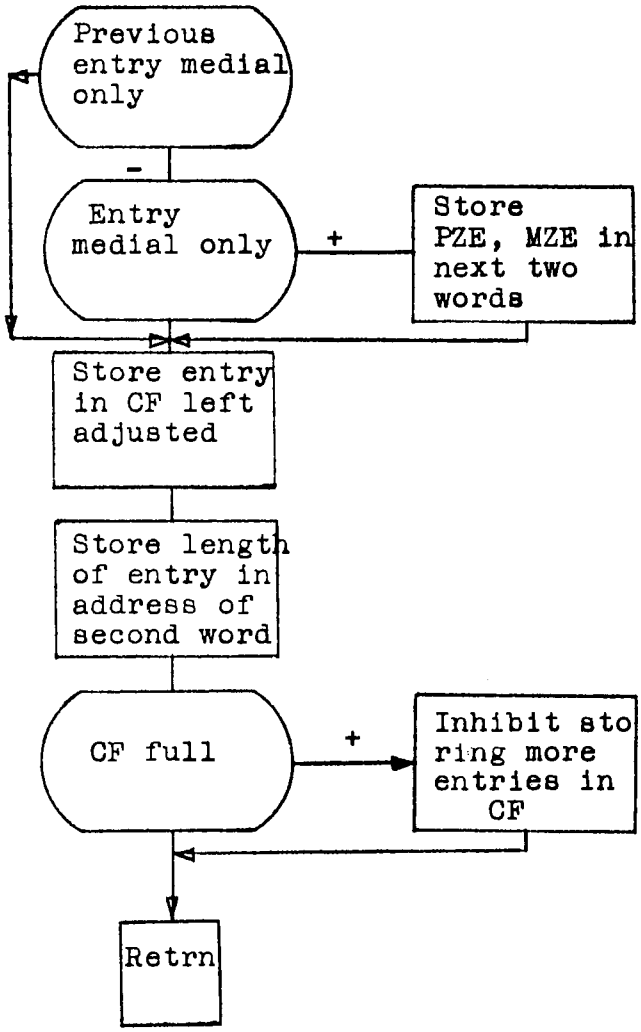


Diagramme 3b Subroutine RDICD: Store detachable prefixes for word segmentation

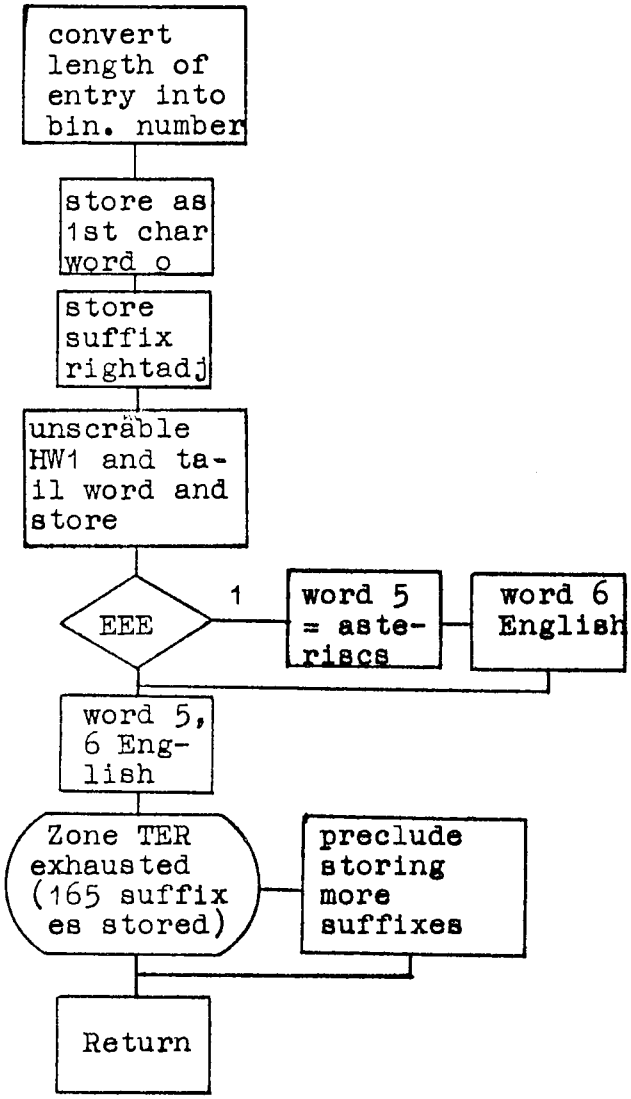


Diagramme 3c Subroutine RDIS: storage of suffixes for analysis of unknown entries.

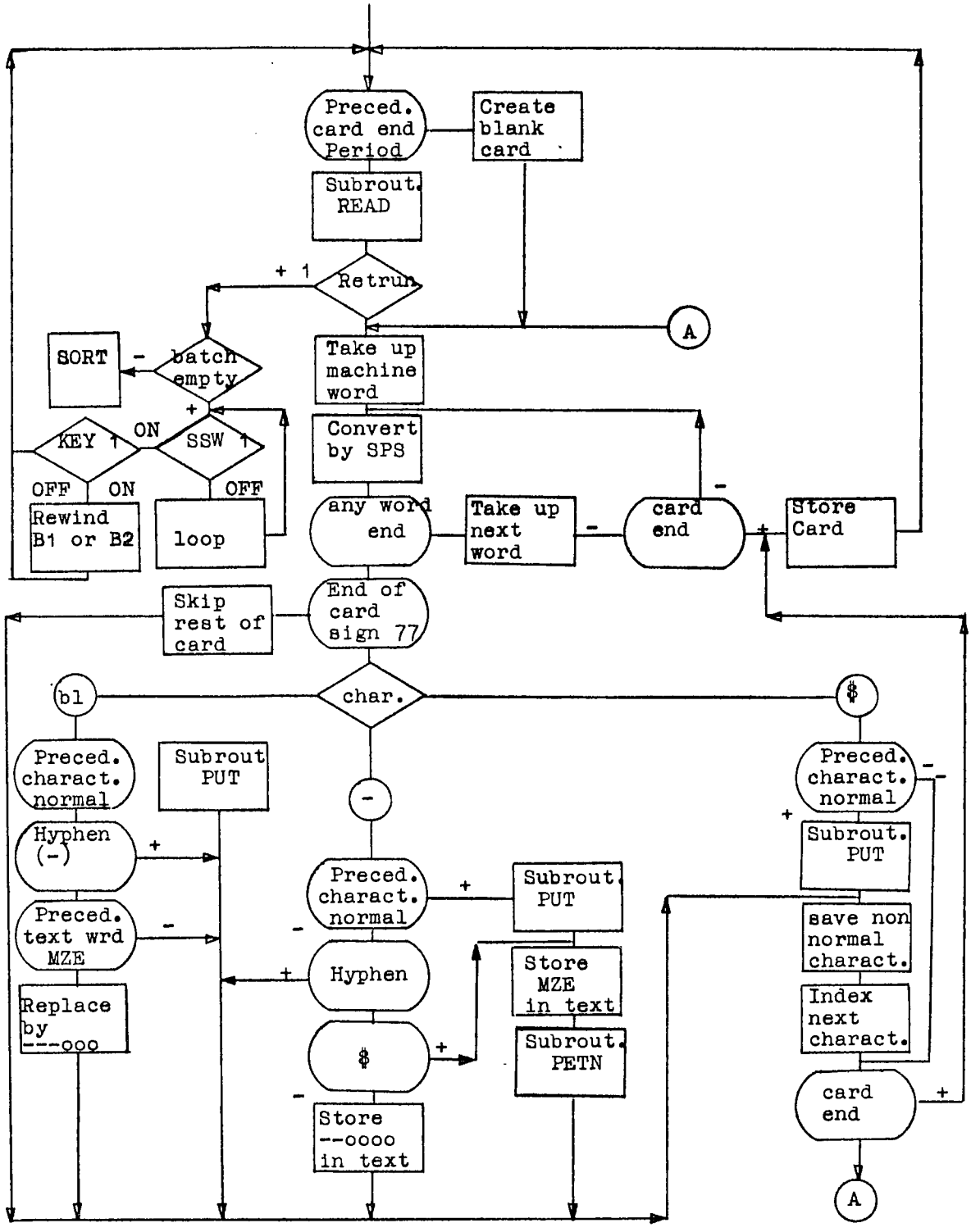


Diagramme 4 Subroutine RDC: Read and locate input text

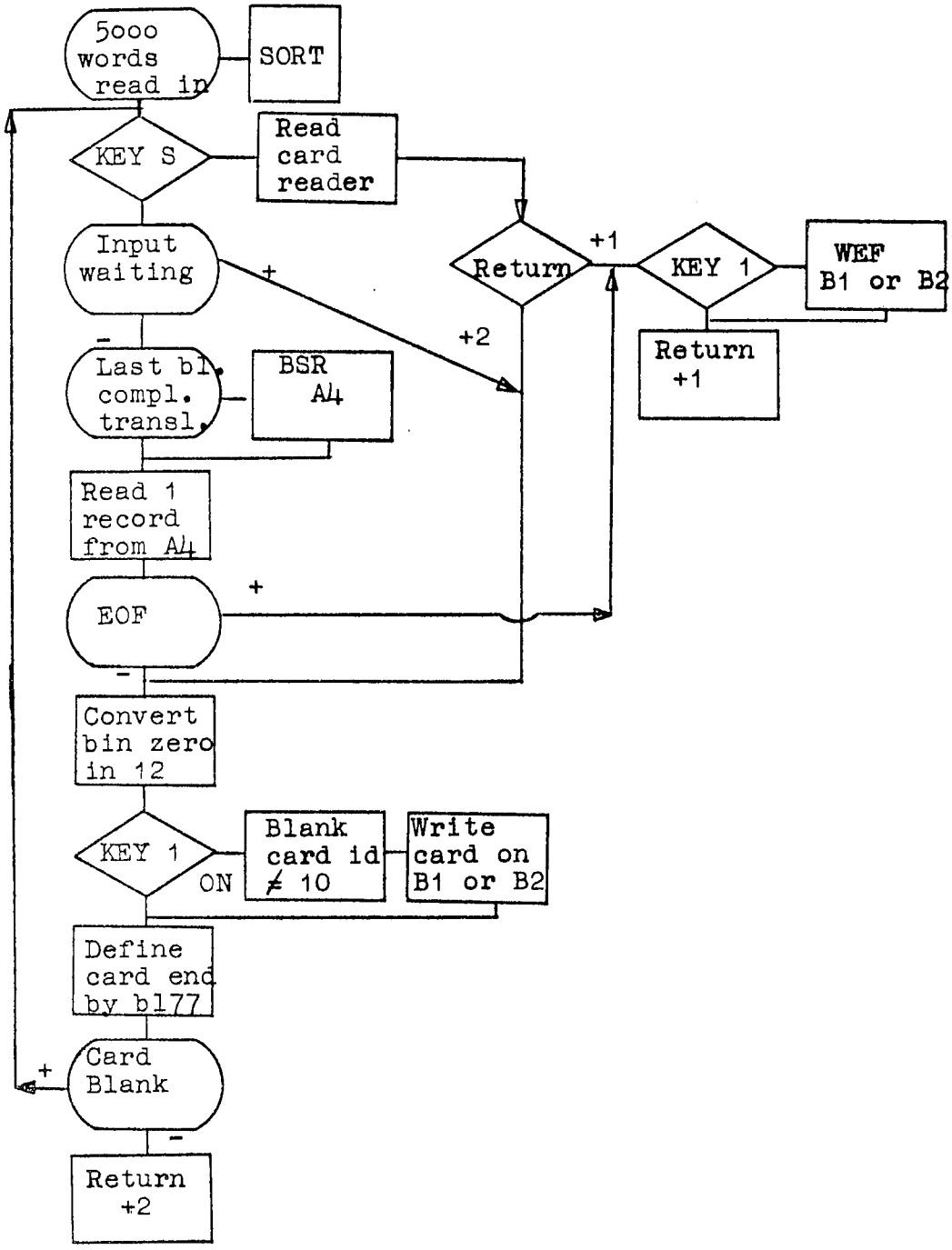


Diagramme 4a Subroutine READ: Read 1 card into memory.

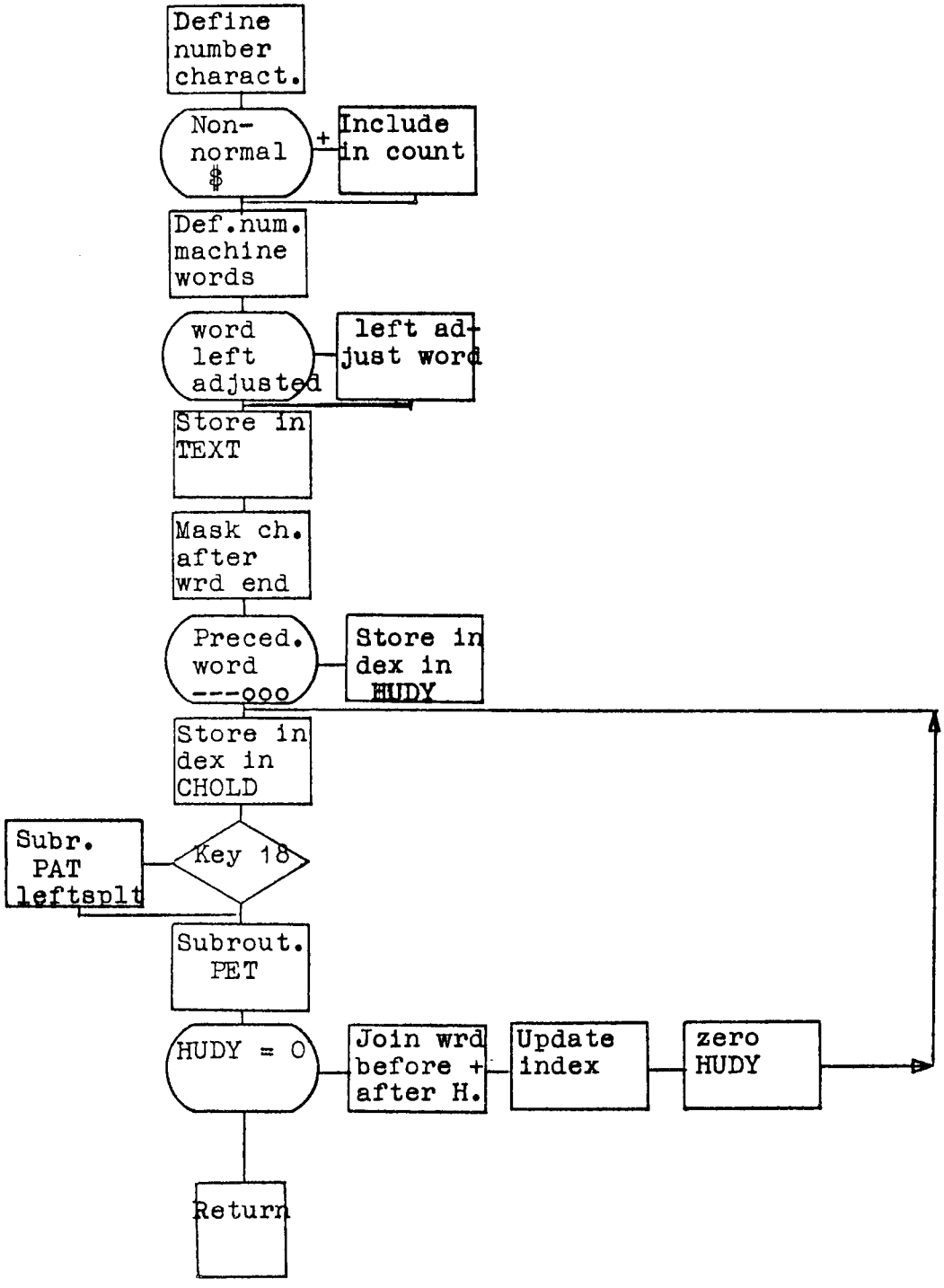


Diagramme 4b Subroutine PUT: Locate words read in TEXT

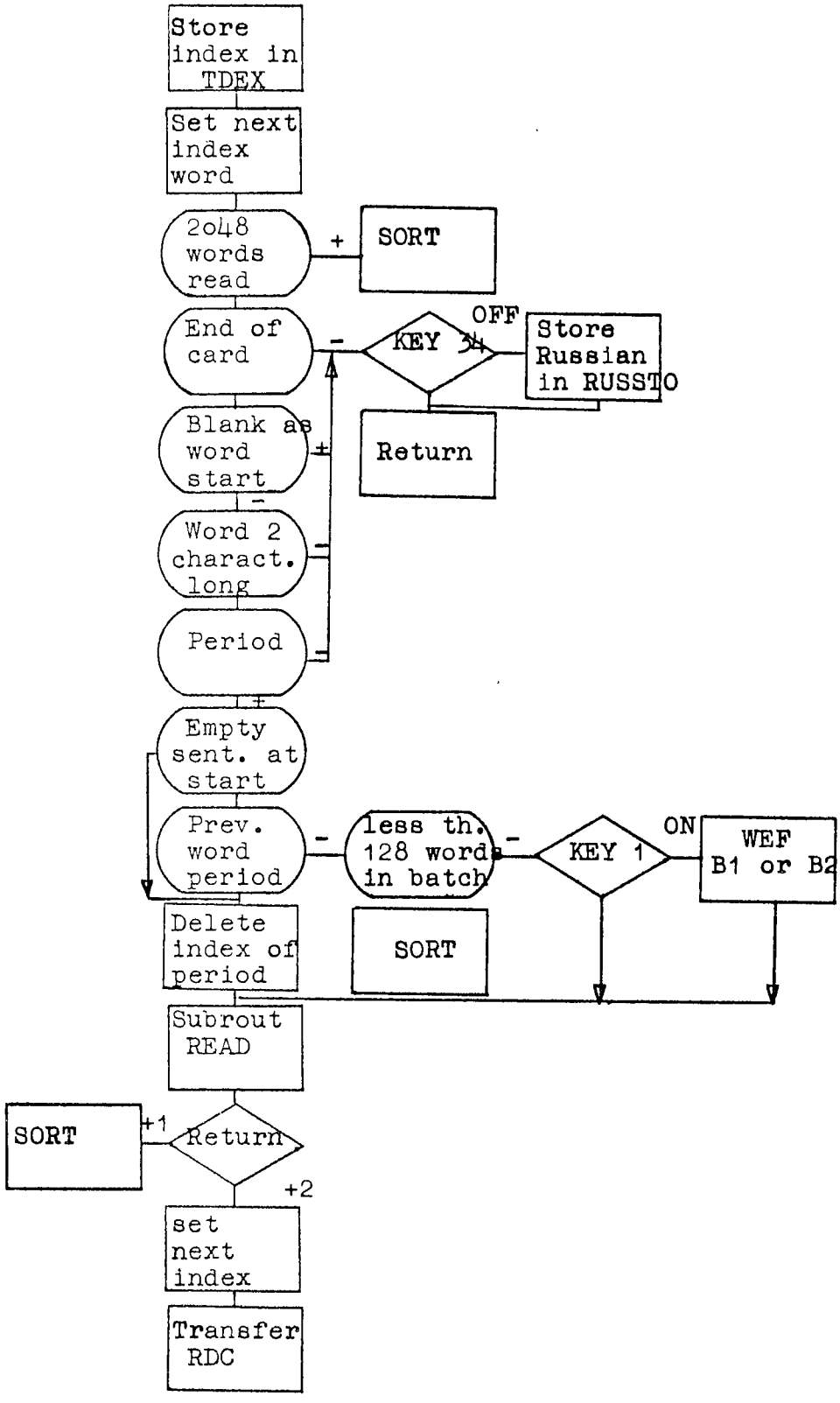


Diagramme 4c Subroutine PET: Define indexes of word read in

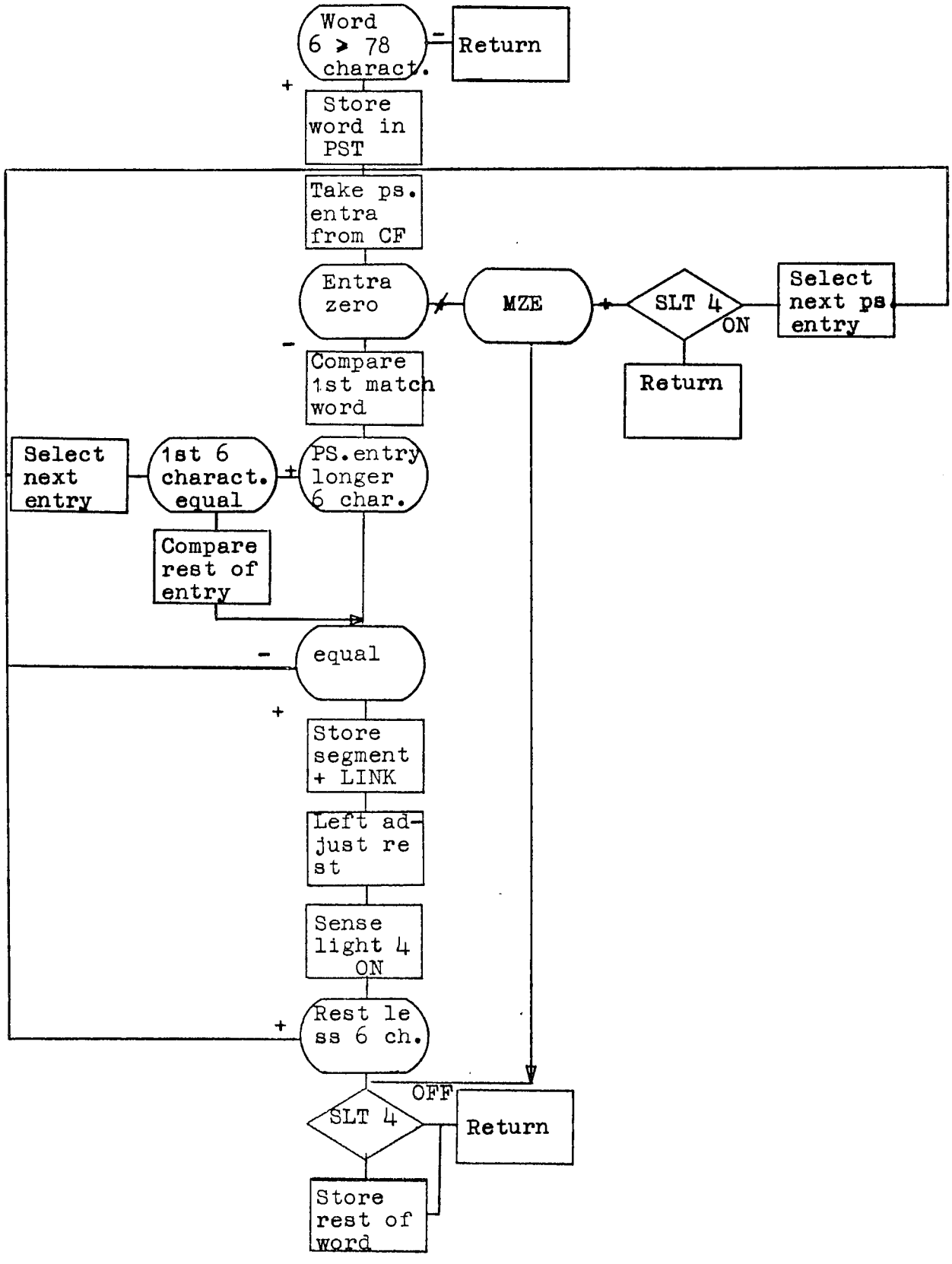


Diagramme 4d Subroutine PAT: Segmentation of Words from Left Hand

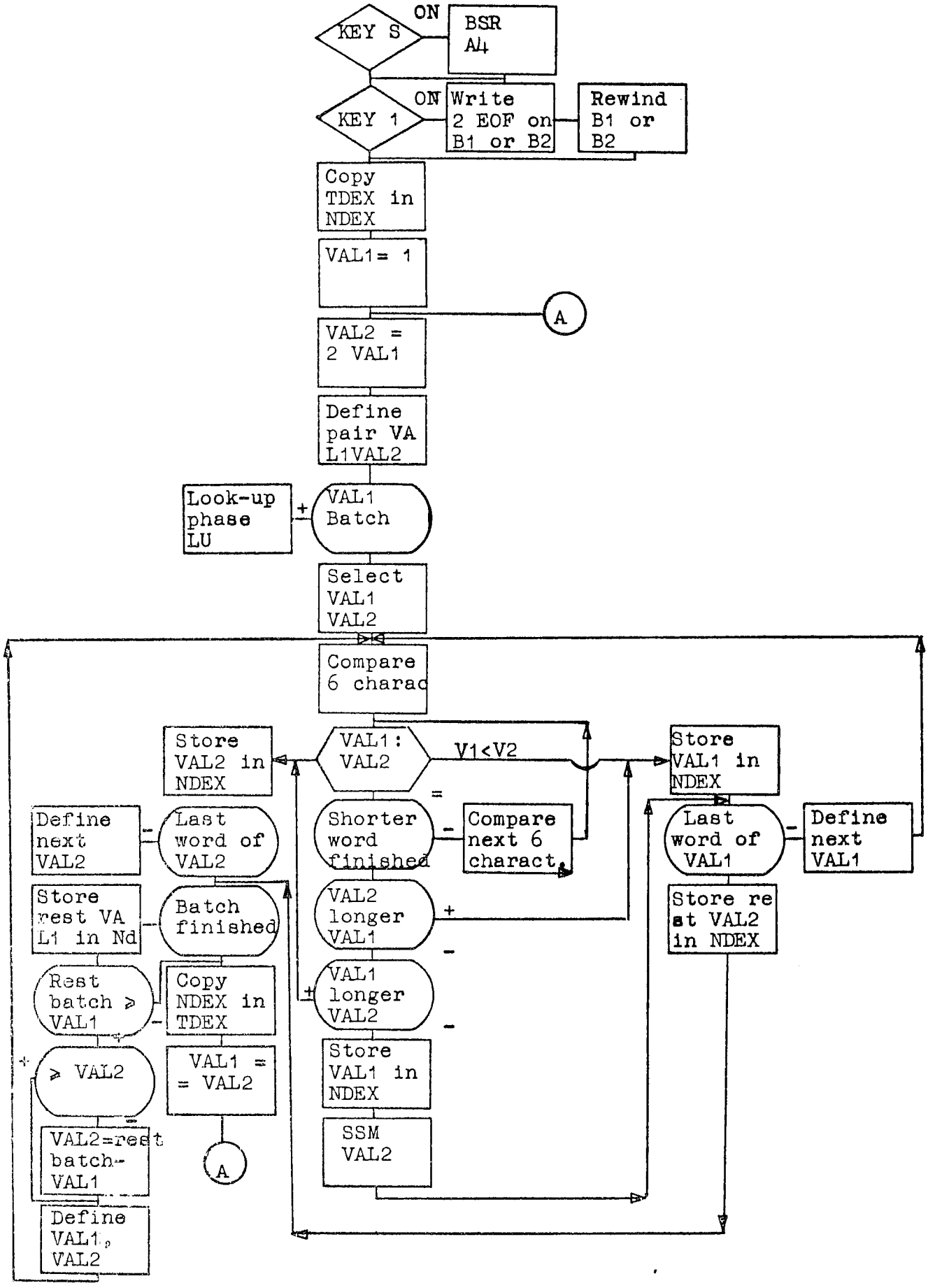


Diagramme 5 Subroutines DEZN, DEZE, SORT: Sort input batch

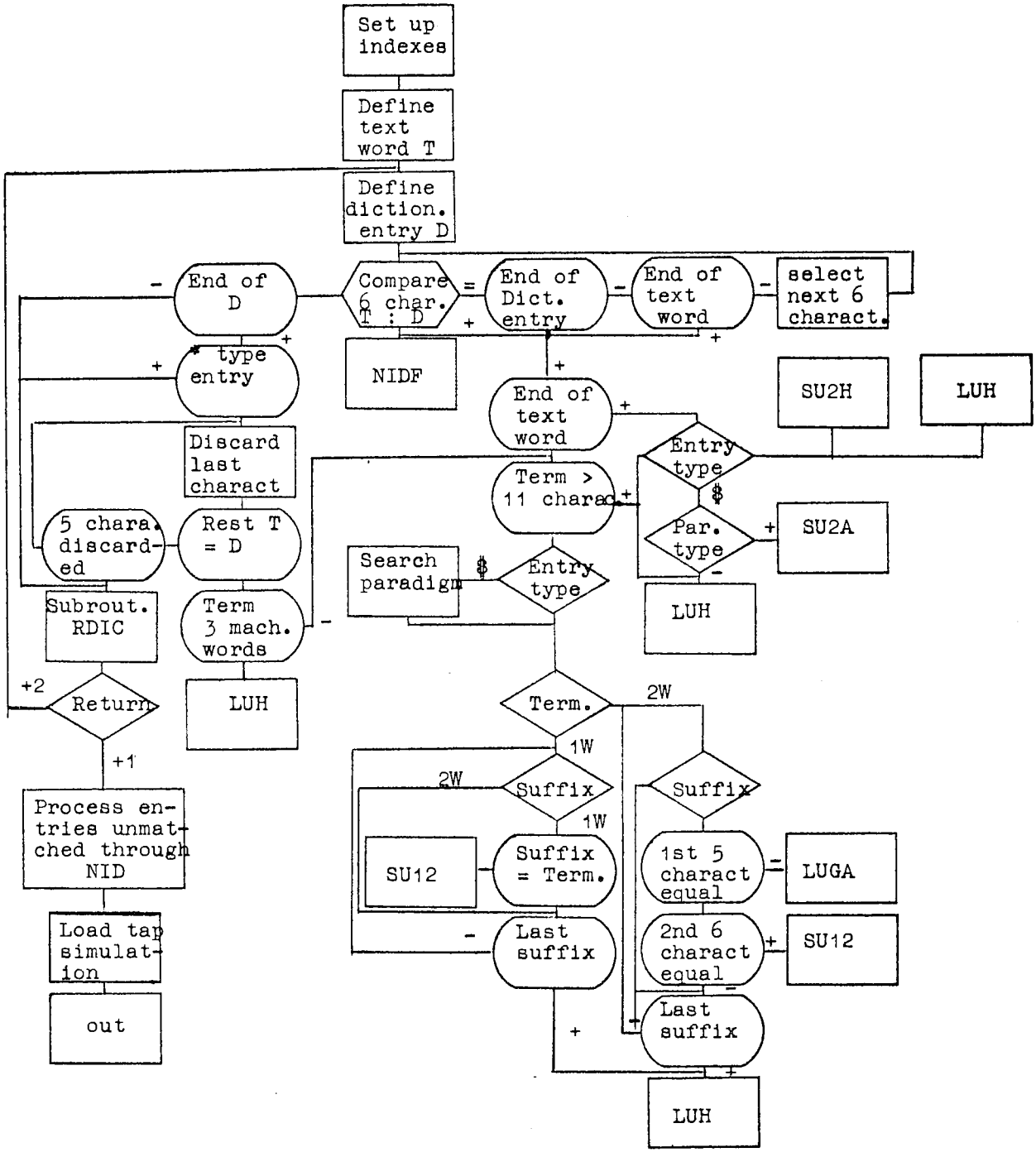


Diagramme 6 Subroutine LU: The dictionary look-up phase

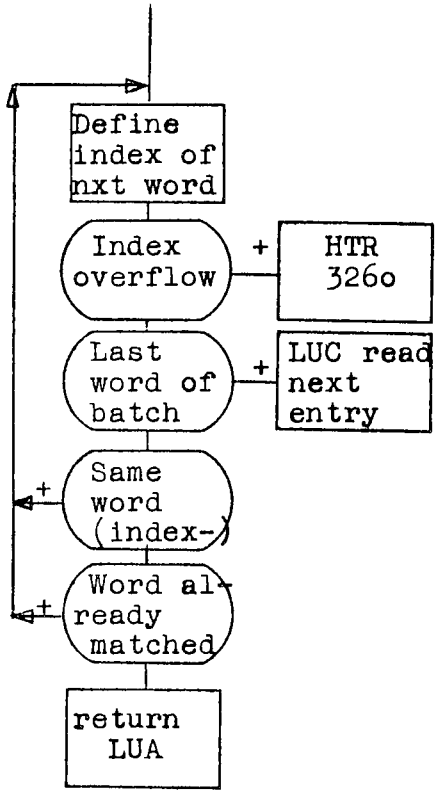


Diagramme 6a Subroutine LUH: Continuation after bad match

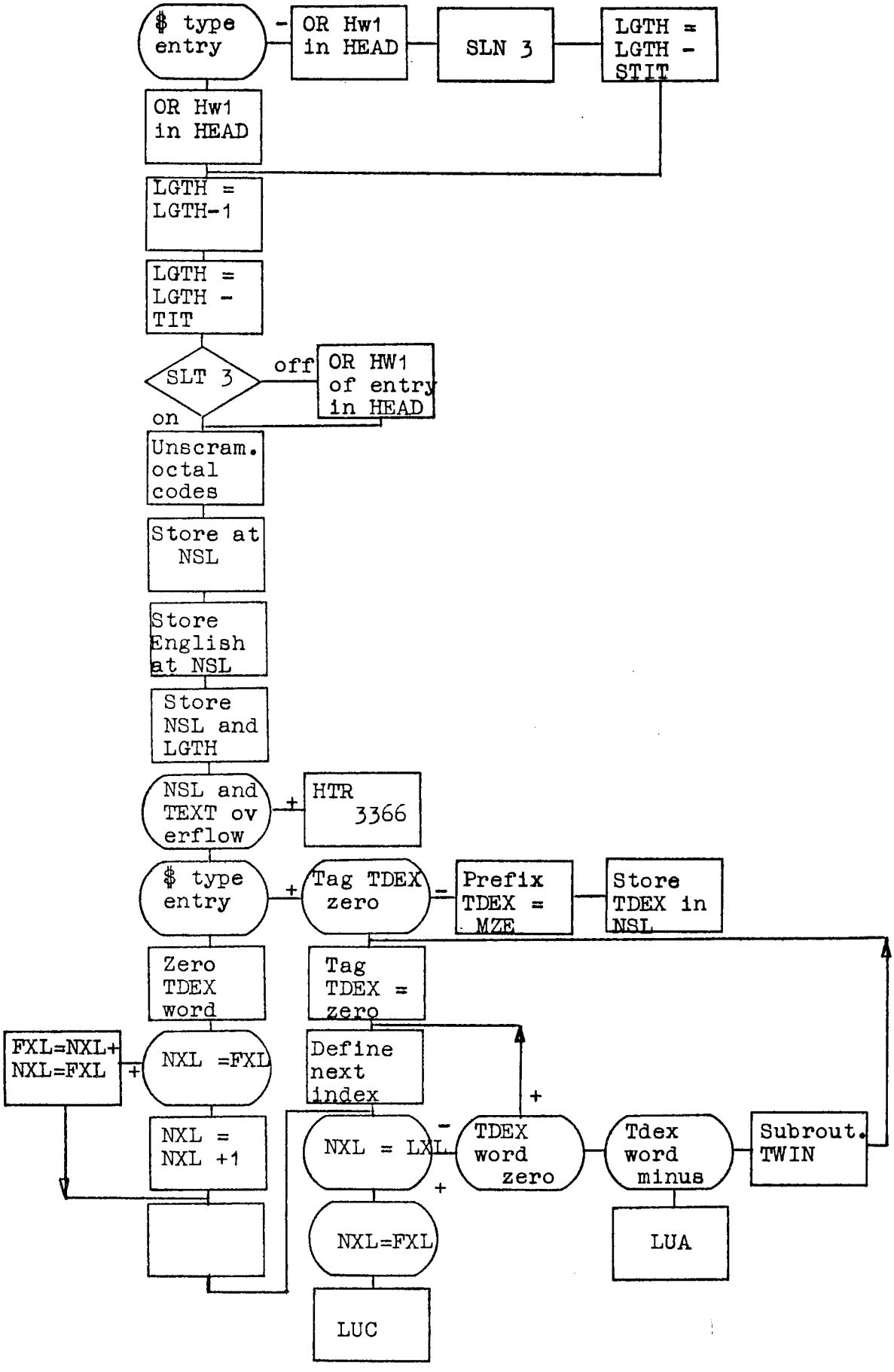


Diagramme 6b Subroutine LU: Successful match

2. The Translation Programme

The translation programme has the task to organise each sentence of the input batch read in by the dictionary look-up programme in storage, to execute the linguistic programme which has been coded in symbolic form as independent or local operations, and to write out the result which is supposed to be the translation of the input sentence.

2.1. Loading and Initialisation (see diagramme 1)

The translation programme which is the third file of the SLC system tape is loaded at the end of the dictionary look-up phase by Load Tape simulation. The usual tests for tape errors (absence of an end of file and redundancy check) are executed, and loading is attempted again, if there is any of them.

Then the fourth file of the system tape, which contains the symbolically coded independent operations in absolute form as one binary record, is read in. This OPFILE is organised as follows: The first word contains the highest numerical name of an independent operation assigned during the assembly run. This word is stored in the symbolic location ONLIM (106), and each independent instruction is tested for correctness by comparing it with ONLIM. The next 512 words are read into loc. OPDEX (11550). The address and the decrement part of each word are organised as an index table which contains the address of the independent instruction at the location addressed by the numerical name of the instruction.(The address part contains the reference to operations with an even numerical name, and the decrement part the address of the odd ones)

Tests for tape errors are executed, and if a second read is still bad, a message (OPFILE TPCHK) is printed, and the computer halts. It is up to the operator to accept the bad reading, or to interrupt the run.

The programme test also, whether the operations file over-

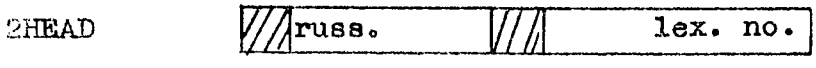
flows the dictionary entries stored by the dictionary look-up programme, and, if so, halts without a restart chance. (HTR 410). If keys 1 and 2o are on, tapes B1 and B2 with the Russian text saved for side by side output are rewound.

2.2. Organisation of a Sentence in Storage (see diagrammes 2 and 3)

During the translation run of one batch the entries found in the dictionary must remain undisturbed, since they are stored once only for each different word. Thus, each sentence is organised separately before the execution of the linguistic operations. The location and the length of the dictionary entry of the current word are looked-up in the TDEX table. The 1st headword is stored at the relative location of the HEAD table. If the sign bit of the first headword is 1, the fourth headword is set-up as described in the SLC manual (i.e.: if the value of the first three octal digits is between 445 and 51o octal one bit is shifted into the corresponding position of the AC register and stored in the fourth headword. If the value of the 3 high order octal digits is between 6oo and 777, the fourth headword is looked up in the table which must be physically the first independent operation. The 3 high- order octal digits of the first headword are set to 4oo....). If the sign bit of the first headword is zero, the fourth headword is also set to zero.

The 5th and the 6th headwords are set to zero initially. If the item is a result of a multiple dictionary match, the low order bit of the 6th headword is set to 1 (which corresponds to the symbolic operation ABT oo1).

The second headword stored at 2HEAD contains in the address part the lexical number of the word, and in the decrement part the address of the Russian word stored for monitoring purposes, if any.



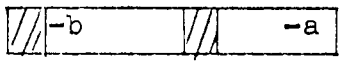
The third headword contains the address and the length of the English translation of the item. The English translation remains as it is in the dictionary entry, unless it is modified by some local or independent operation.

3HEAD 

a = address of the English word
b = length of the English word

The English coded tailwords(characterised by TW in the dictionary) are not indexed and remain indisturbed in the dictionary entry, unless they replace the English translation.

The diacritics and local and independent instructions (TD, TL, TI) are indexed with a list processing technique: The first index word is placed IHEAD and DHEAD with the address of an instruction or diacritic in the address part, and the address of the next index word in the decrement part. The following index words are placed in the free storage between the operations file and the dictionary entries. The diacritics and operations are not moved, since they cannot be modified in any way by the linguistic program. Functions like AI, DI, AD, DD etc, merely add or delete an address in the index list.

DHEAD
IHEAD 

a = address of instruction or diacritic
b = address of next index word

The word order of the sentence is indicated by an index table called LHEAD in this table, to each item there is associated an index, which contains in its address part the number of the item at the right, and in the decrement - of the one at the left the zeroth item closes the chain of the items like a ring. Functions, which insert or eliminat items, or change word order never do it physically, but simply change the index words.

LHEAD 

The zeroth item receives as address of IHEAD an instruction which contains a 0001 as numerical name. This instruction permits the loading of the independent instructions, which must be added by the function AI, as explained in the SLC manual.

There is sufficient space provided for 128 items (included the zeroth item). However, the programme stops placing items, after the 121st item. If there is a punctuation mark placed before, the sentence is cut at it, otherwise arbitrarily after the 121 item. Normally the period is encountered before, since sentences of more than 121 items are not very frequent. If a sentence has been cut before meeting a period, the low order bit of the first headword is set to 1.

2.3. The execution of the linguistic operations

For the translation programme, the independent and local operations do not present any difference except the place where they are stored, and the title word. A local operation has a title word:

4ooCCFFPPPoo

where CC is the number of constants, FF the number of commands, and PPP the priority number.

The independent operations have as title word:

OOONNNNPPPEE

where NNNN is a numerical name of the operation, PPP the priority number, and EE the entry address.

The programme scans the instructions indexed by the IHEAD lists, always starting from the zeroth item, and chooses the instruction with the lowest priority number. If two instructions have the same priority number, the one encountered before is preferred. The address of the instruction itself is removed from

the waiting list, and the instruction is displayed in storage as follows: T+6 contains the constant no. ooo, the location T+7 contains the command number oo, T+8 contains the address of the command actually being executed. This address is at the beginning ooo for local operations, and is computed with the entry address for independent operations.

T+2 contains the number of the source item, T+9 the number of the current item. T+1o contains GG, T+11 J, T+12 the YES transfer, and T+13 the NO transfer of the command.

First of all the prefix of the command is interpreted and executed, if it is not zero. If the prefixes LZ or RZ yield a NO answer, and the command itself is not a decision function, the operation is abbandoned. Otherwise the NO branch is executed.

The operation itself is defined by the operation code (the 6 low order bits of the decrement, and the NO address, if it is higher than 700 (actually higher than 773), and is interpreted as a transfer to a subroutine which carries out the function as described in the SLC manual. Return from the subroutines is either the YES, or the NO, if no abnormal condition has been encountered during execution, and the Error routine had been entered. If the new address is higher or equal 700, the operation is abbandoned. This may correspond either to the normal transfer to OUT in the symbolically coded operation (///) or to an abnormal condition during the execution of a function. In the transfer to YES or NO, also the entry and return from subroutines is considered. If the transfer address is higher or equal 400_8, the indexes T+6, T+7, and T+8 are saved. If the address is between 2oo and 377, the last indexes saved are restored. This is one of the reasons, why one can enter a subroutine once only at a time, since the entry distroys the indexes saved before. In order to eliminate this restriction, one should save the indexes with a list processing technique. in a push-down list.

After the execution of an operation the programme returns

to the search for the next operation with the lowest priority number.
(See diagramme 4,5 and 6)

2.4. The output of the translation (See diagramme 7)

When no more instructions are waiting for execution, the English correspondence is written out as what is supposed to be the translation of the text. If both, key 2 and 3 off, the programme halts, and waits for one of the keys to be set ON. Nothing is lost, as indicated in the SLC manual. The third headwords of each item are arranged as a consecutive sequence of I-O commands (IOCP a,,b). If only Off-line output on tape A2 is desired (key 3 ON), the text is not edited, and written out on tape as it stands. Editing is up to the listing programme. Records are up to 300 words long, the first record is identified by a word containing 000006, if a sentence is cut into several physical records, the following are identified by a word containing a series of 52 octal (minus zero).

If key 2 is ON for On-line printing or output on A3, the translation text is edited by suppressing the filling asterisks and equal signs. When the translation has been written out, control is transferred to the organisation of the next sentence in storage, At the end of the batch, the dictionary look-up programme is reloaded, and the next batch of input text is processed.

In this description, the subroutines constituting optional features, such as on-line printing, dumps, and monitoring instructions and functions, are not dealt with largely, since they are only marginal features. The subroutines which execute the SLC commands are described sufficiently in the SLC manual, and there is no particular interest in describing each of the machine-sub-programmes.

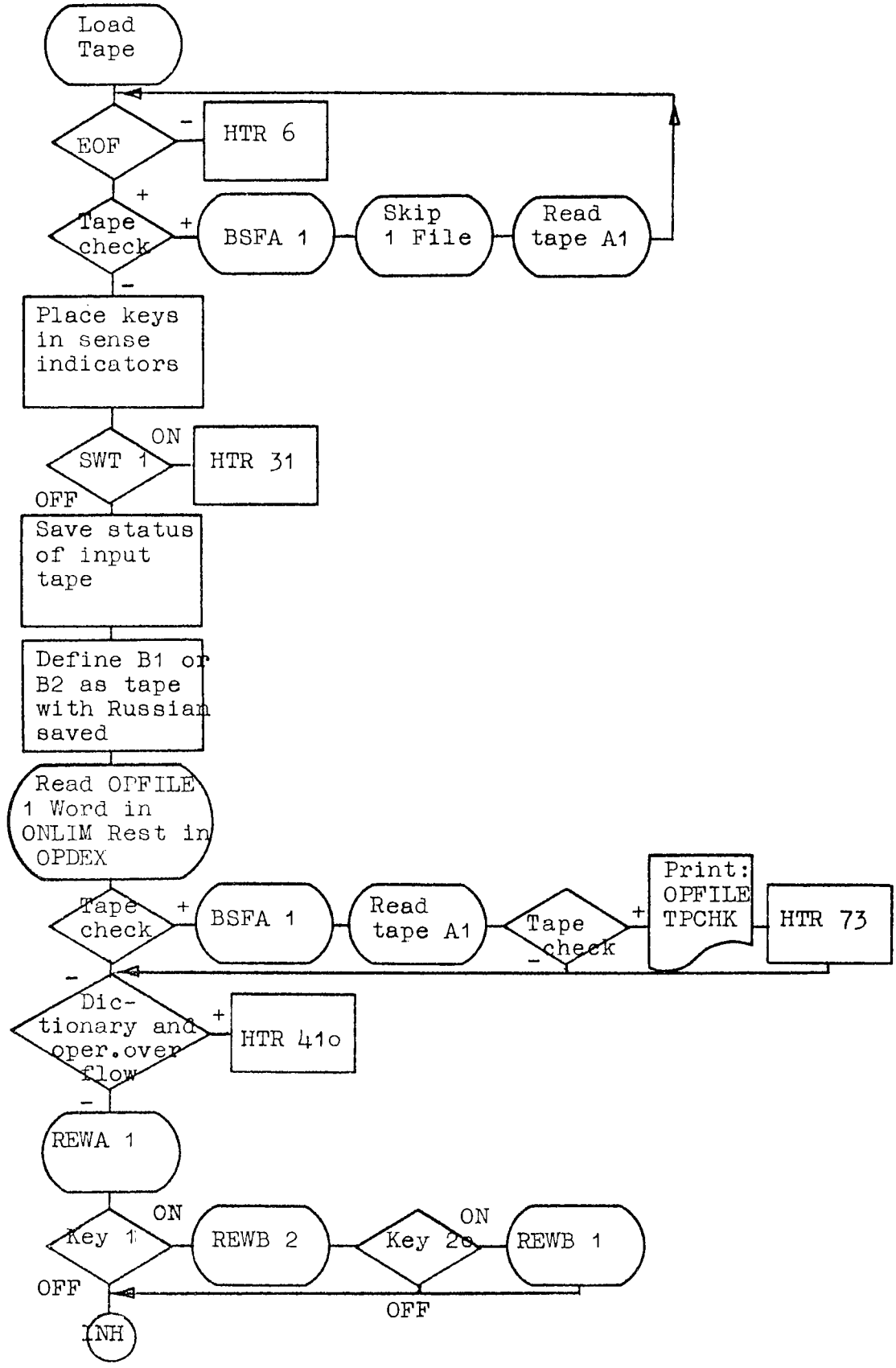


Diagramme 1 Translation program Loading and initialisation

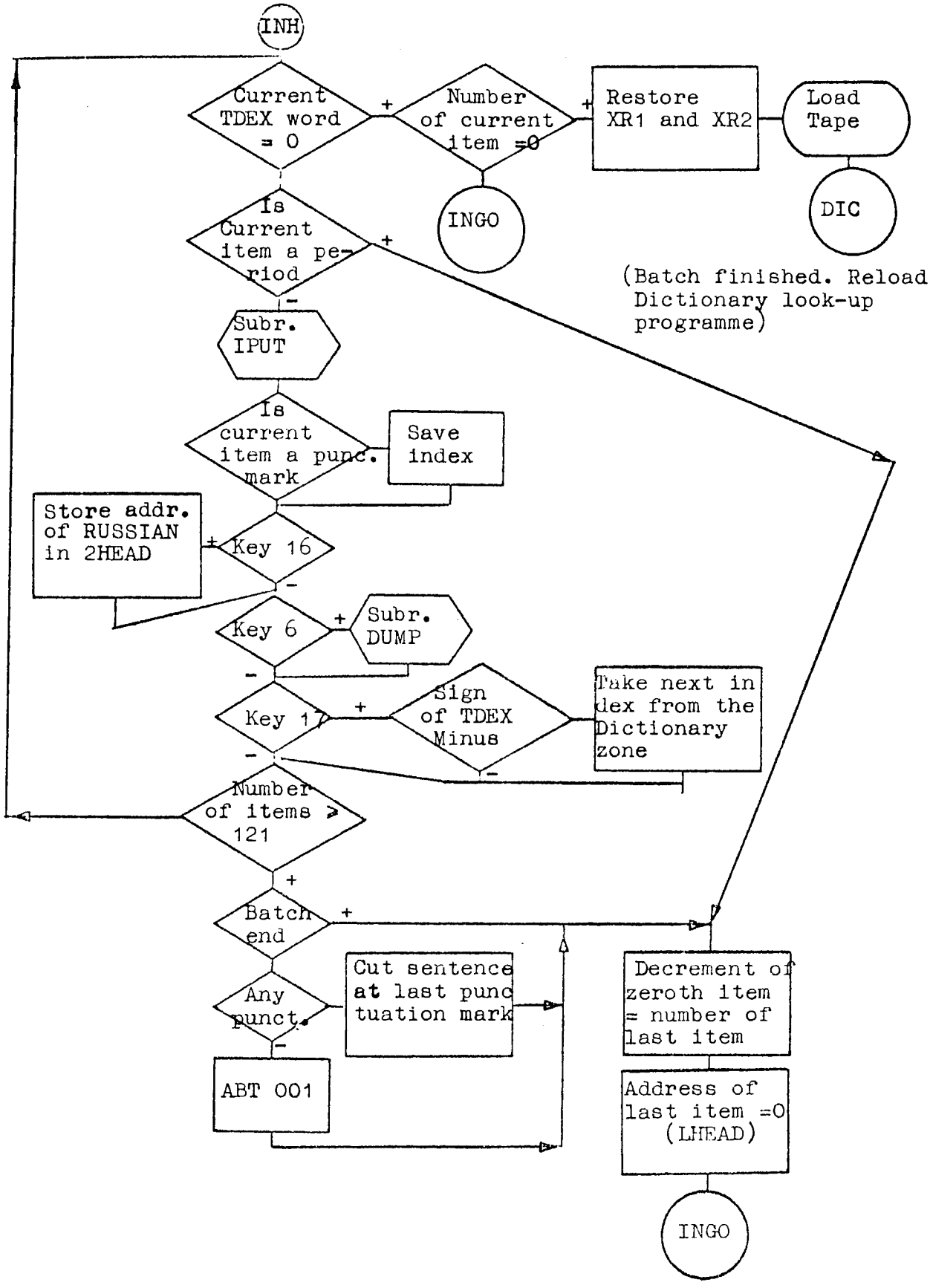


Diagramme 2 Translation Program Set-up one sentence in storage

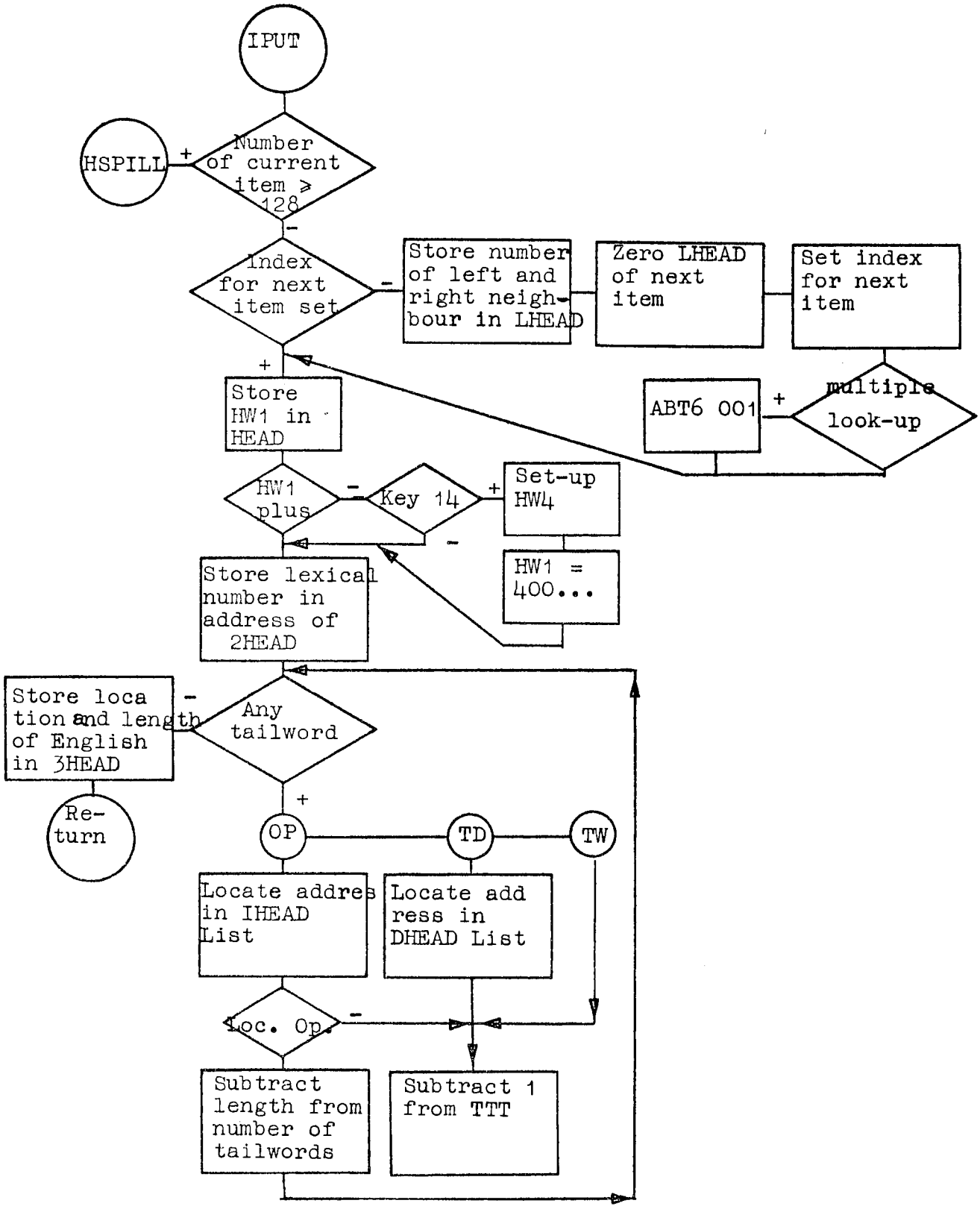


Diagramme 3 Translation Program Subroutine IPUT
Set-up item in storage

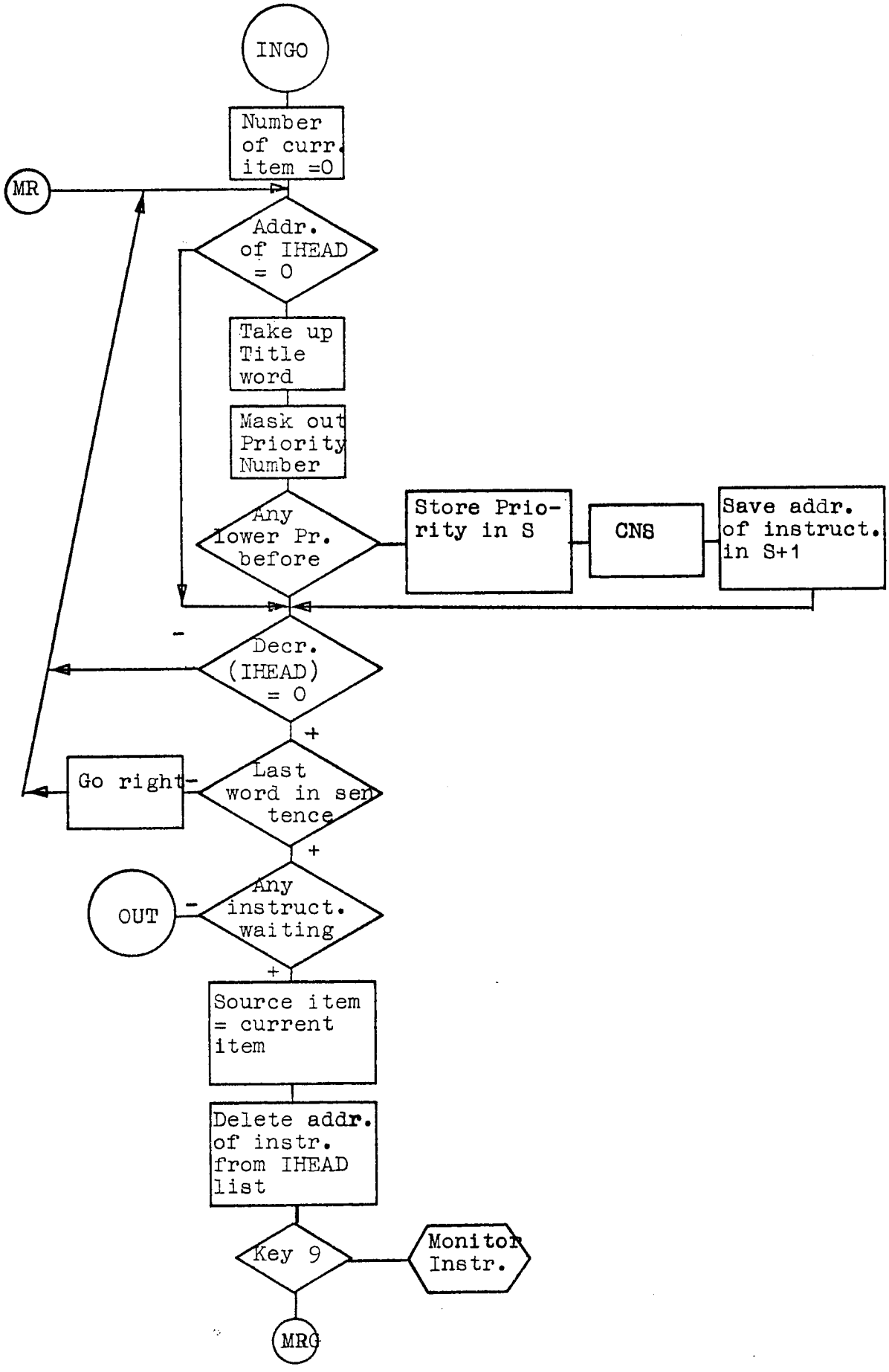


Diagramme 4 Translation Program Search for operation with lowest priority number

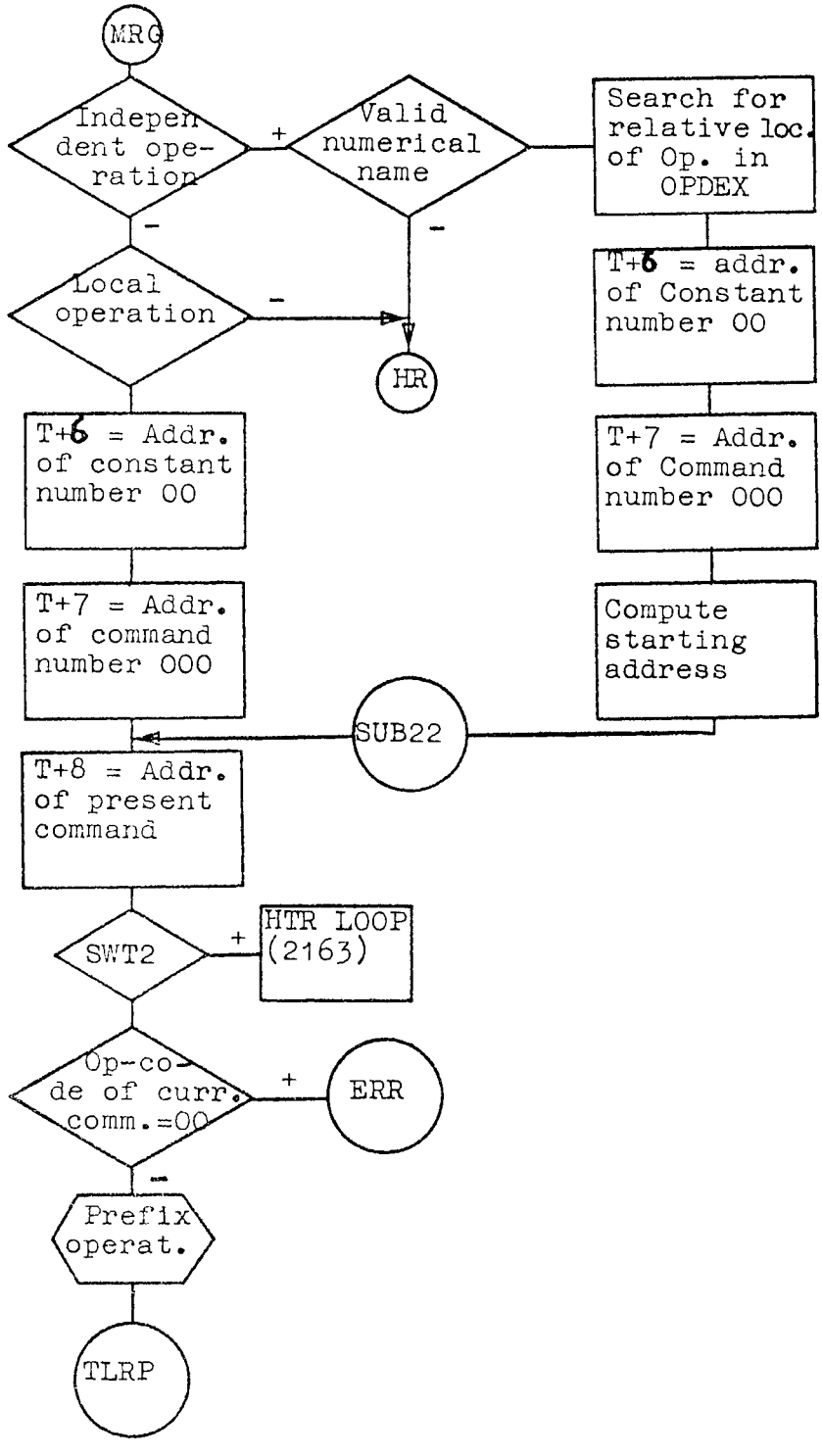


Diagramme 5 Translation Program Begin execution of operation

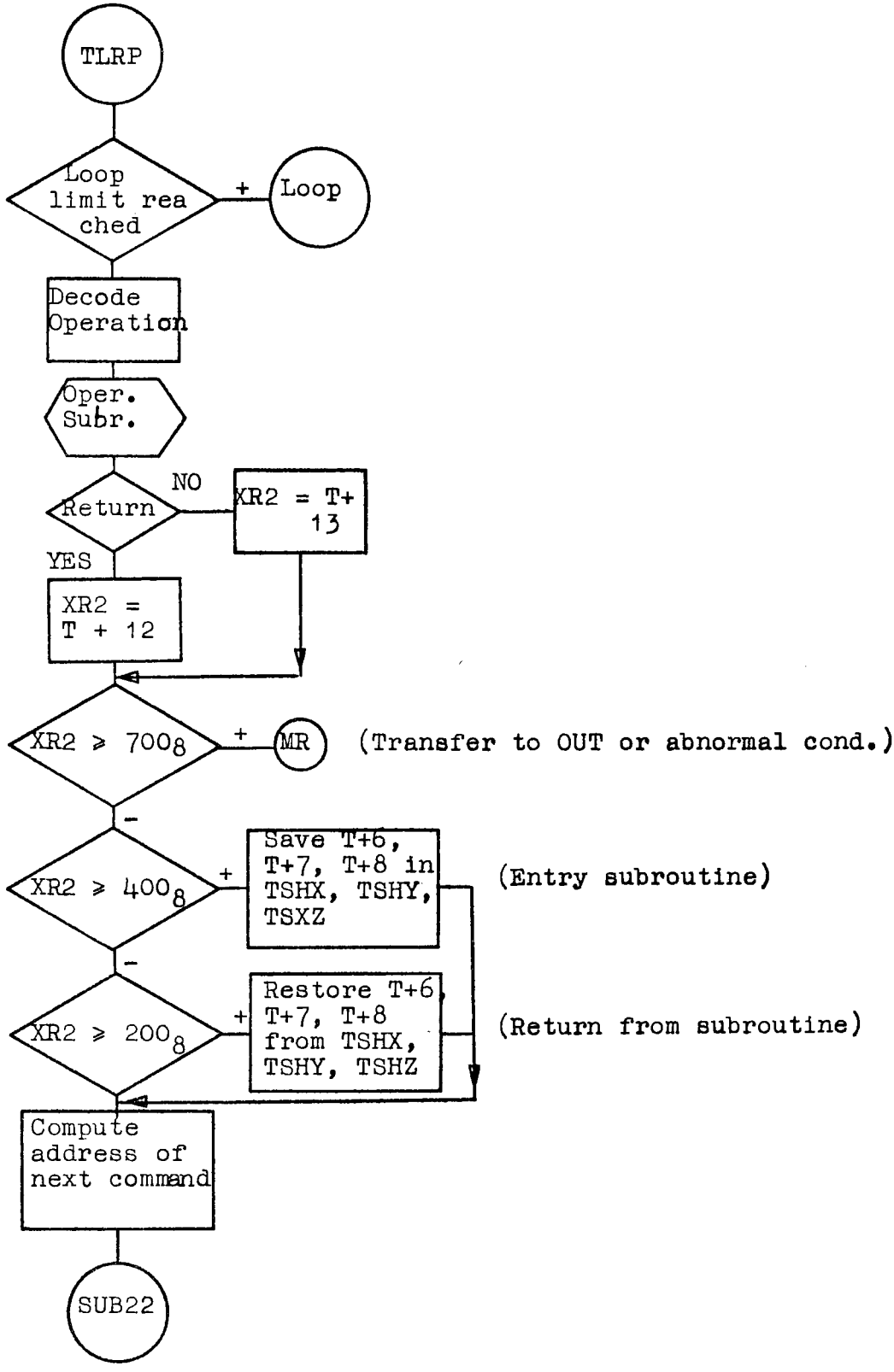


Diagramme 6 Translation Program Execute operation

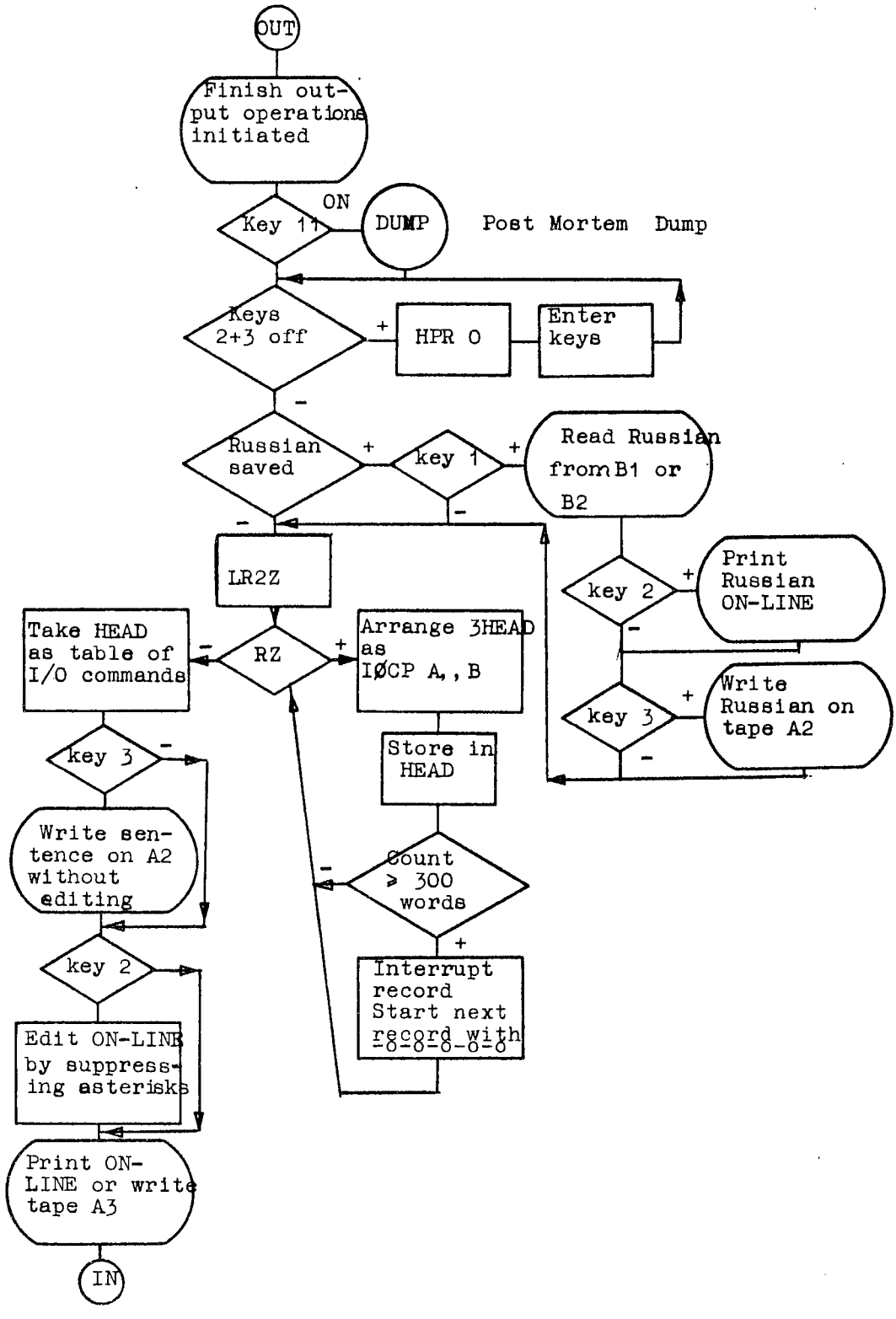
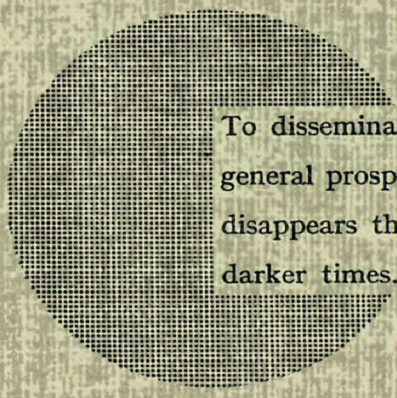


Diagramme 7 Translation program Subroutine OUT
Output translation



To disseminate knowledge is to disseminate prosperity — I mean general prosperity and not individual riches — and with prosperity disappears the greater part of the evil which is our heritage from darker times.

Alfred Nobel

SALES OFFICES

All Euratom reports are on sale at the offices listed below, at the prices given on the back of the cover (when ordering, specify clearly the EUR number and the title of the report, which are shown on the cover).

PRESSES ACADEMIQUES EUROPEENNES
98, Chaussée de Charleroi, Bruxelles 6

Banque de la Société Générale - Bruxelles
compte N° 964.558,

Banque Belgo Congolaise - Bruxelles
compte N° 2444.141,

Compte chèque postal - Bruxelles - N° 167.37,

Belgian American Bank and Trust Company - New York
compte No. 22.186,

Lloyds Bank (Europe) Ltd. - 10 Moorgate, London E.C.2,

Postcheckkonto - Köln - Nr. 160.861.

OFFICE CENTRAL DE VENTE DES PUBLICATIONS DES COMMUNAUTES EUROPÉENNES
2, place de Metz, Luxembourg (Compte chèque postal N° 191-90)

BELGIQUE — BELGIË
MONITEUR BELGE
40-42, rue de Louvain - Bruxelles
BELGISCH STAATSBLAD
Leuvenseweg 40-42 - Brussel

DEUTSCHLAND
BUNDESANZEIGER
Postfach - Köln 1

FRANCE
SERVICE DE VENTE EN FRANCE
DES PUBLICATIONS DES
COMMUNAUTES EUROPEENNES
26, rue Desaix - Paris 15e

GRAND-DUCHE DE LUXEMBOURG
OFFICE CENTRAL DE VENTE
DES PUBLICATIONS DES
COMMUNAUTES EUROPEENNES
9, rue Goethe - Luxembourg

ITALIA
LIBRERIA DELLO STATO
Piazza G. Verdi, 10 - Roma

NEDERLAND
STAATSDRUKKERIJ
Christoffel Plantijnstraat - Den Haag

EURATOM — C.I.D.
51-53, rue Belliard
Bruxelles (Belgique)