https://theses.gla.ac.uk/

Theses Digitisation:

https://www.gla.ac.uk/myglasgow/research/enlighten/theses/digitisation/

This is a digitised version of the original print thesis.

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses
https://theses.gla.ac.uk/
research-enlighten@glasgow.ac.uk

A SUITE OF PROGRAMS FOR

TEXTUAL ANALYSIS ON THE KDF9 COMPUTER

a thesis presented to

The University of Glasgow

in fulfillment of the requirements

for the degree of Master of Science

by

Eleanor A. Dickie, B.Sc.

November, 1968

ACKNOWLEDGEMENTS

I would like to thank the following for their help and encouragement in the compilation and preparation of the work of this thesis:-

Prof. D.C. Gilles for his supervision throughout the work;

Mr. K. MacDonald of the Celtic Department of Glasgow University for his views on the preparation of the glossaries

and many other members of the staff of the Computing Department at Glasgow University, who gave advice and help.

ProQuest Number: 10646168

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 10646168

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

SUMMARY

This work is concerned with the preparation of glossaries and concordances for textual analysis using a computer. A historical background and introduction is given before the present work is discussed. The discussion incorporates a brief description of the programs involved. Then follows an outline of possible extensions and improvements.

INDEX

CHAPTER 1

INTRODUCTION

This work is concerned with the development of a suite of programs for textual analysis, the analysis of a text in terms of the word-forms it contains and their frequency. This analysis involves two kinds of work, firstly the preparation of word lists of all the word-forms in a piece of text. These word lists exist in a variety of types, but they belong to two main classes. These may be defined following Rev. A.Q. Morton (1) as follows:-

The first, the glossary, is an ordered list, usually the order is alphabetic, of all the word-forms in the text, with a count of the number of times each word occurs in the text. The second, the concordance, is an ordered list of all the word forms in the text with a position reference indicating where each word-form will be found in the text.

Textual analysis is also involved with the analysis of the structure and patterns of text. Examples of this are sentence length distribution, and the frequency of certain words within sentences or within groups of n words. These are more specific problems and before any analysis of this

sort can be attempted, the basic concordances must be obtained. This is the problem which has been mostly dealt with until now.

Early work on concordance preparation

The task of compiling concordances was carried out as early as the thirteenth century. In this case the Bible was the work involved. Indeed, at that time all work of this type was done on the Bible, so that, today, many people associate the word 'concordance' with the Bible. However, a number of concordances of other works do exist e.g. Horace, Prudentius.(2)

The method employed was to work through the text writing out a slip of paper for each word. The slip often included a reference and a line of context. This involved a great deal of time and labour.

A typical example of the dedication shown by scholars is that of Professor Ewald Flügel who, in 1891, took over the task, begun in 1871 by the first of his three predecessors, of compiling a Chaucer concordance. Despite much full-time labour and assistance, however, he died in 1914 having only completed the letters A-H. (2)

Another example is the report that in 1263 Cardinald Hugh of St. Cher enlisted the services of no less than 500 monks when compiling a concordance of the Bible.

Perhaps the best known concordance is Cruden's concordance to the King James Version of the Bible, London & Edinburgh, 1736

Compiling the concordance took Cruden three years. It is estimated that he worked nineteen hours per day and at the end was left "subject to intellectual infirmity, to overclouding of the mind."(1) All of the work done by hand dealt with only the significant words of the text and still there were many errors and omissions. The fact that the common words were omitted meant that the use and scope of the concordances was limited.

A considerable advance over the hand-written slip was made earlier this century in America by Lane Cooper. He avoided mistakes and excess copying by using several copies of the text involved and cutting them up so that each slip contained a printed line. Rubber stamps were used for references. A concordance of Horace (3) was produced in less than a year by this method, yet it required eighteen assistants and many other helpers to order the slips. So, although greatly reduced, the time and labour involved was not as yet insignificant.

Early application of Mechanized Techniques

Mechanized techniques of information analysis were first developed during the second world war, and since then the

construction of word lists has been entirely mechanized, firstly by punched card equipment and later by electronic digital computers. The use of a digital computer in indexing the Summa Theologiae of St. Thomas Aquinas showed how wide were the possible applications in all branches of scholarship.

Punched cards were first used for storing the words of a text. Relatively slow sorters were then used to give the word lists of the text. As one card contains eighty columns and since most words have a length less than twenty letters, each card was used to store four words. This made for complicated sorting processes.

From a computer point of view, the task of preparing these word lists is essentially the problem of sorting, and this in turn is primarily a problem of sufficient storage. In order to sort a complete piece of text, it must first be available as a whole to the machine. In the early computers with no backing store, few machines had enough space to store a complete text.

To overcome this problem it was necessary to divide the words of the text into a number of classes or groups, each of which was small enough to fit into the computer store. By making a partial listing of each class of words, the partial listings could then be combined to produce the entire

list for the text being analysed, the text thus passing through the program several times. Another method was to use the output medium of the computer as an auxiliary store. Thus, any information obtained from the text for which there was no room in the main store was punched on an "overflow" tape. When the first 'storeful' of information had been printed, the information on the overflow tape was collected and stored by treating it as a new text. The process was continued until no further overflow tape was produced. The output from the various cycles could then be combined to produce the entire list for the text.

The amount of manipulation required in either of these ways and for card equipment is very large. It was not until backing store such as magnetic tape became available that other, more efficient, methods of sorting were developed. By 1956, R. Busa and Paul Tasman of IBM were considering transferring text from cards onto magnetic tape and processing this by computer. This and the development of larger computers helped to ease the amount of work and time involved.

Significant Differences and Implications

The significance of the development of mechanized techniques is that now we have available - word lists of texts with every word being considered - not just the key

words as was the case in hand processing. The rare words reflect an author's interest, intellectual grasp and educational background: they are characteristic of the author. But it is the frequent words which he writes in all circumstances - so tending to make their use neutral to context and subject matter - and it is the rate at which he uses these frequent word-forms which most readily distinguish him from his contemporaries.

These facts have been developed into actual theories and applied to many problems of authorship.
Rev. A. Q. Morton has worked with M. Levison and W.C. Wake on the authorship of the Pauline Epistles. (4) He used conditions such as sentence length distribution, number of occurences of the common Greek words "en", "kai" and "de" on which to base his theories. Other examples of applications of textual analysis to authorship determination are the Federalist Papers, (5) Plato's 7th Letter (6) and The Junius Papers. (7)

Lack of standardisation so far

All of the work done in textual analysis until very recently has been aimed at solving specific problems and as such required data to be prepared in a specific form

often with very strict rules. Therefore, text prepared for one kind of analysis could frequently not be used as data for any other.

Objectives of present work

The main purpose of the present work was to overcome this lack of uniformity by producing a system which would be flexible enough to accept various types of input. If this were possible, libraries of texts could be stored and accessed whenever required for different forms of investigation.

We had, therefore, three main objectives. The first was, using existing facilities on the KDF9, including the Sort Program Generator and file structure, to produce a suite of programs to give word lists in forward or reverse order. Secondly, we wanted to make this suite such that by making simple alterations, it could be made to accept text in any code and in any form. Thirdly, we hoped to set up a system whereby a programmer could easily access individual items of information and carry out extra calculations in which he was particularly interested e.g. the frequency of the use of commas. This, we felt, would provide a more general system than had been produced to date. The first step was to produce a suite of programs to form the word lists accepting our own KDF9 paper tape

code and this is the project which forms the body of the work of this thesis.

CHAPTER 2

PROBLEMS OF TEXT REPRESENTATION ON A COMPUTER

General problems (or The problems in general)

Although in Chapter 1 the use of computers in textual analysis was described, no mention was made of the problems of the representation of a piece of text in a form suitable for a computer. There are three separate problems involved here. A text is represented inside a machine in the form of a file which can be accessed by means of a file name. The format of the file name forms the first problem. The second is the bibliographic description which will be of use only to the scholar and, as such, requires separate treatment. The third problem is that involving the representation of the actual text itself.

File name

The length and form of the file name depends completely on the machine for which the text is intended. This is because the file name is normally of the same length as a single machine word, so that it can be accessed by the normal fetch instructions of the machine language.

Although, in the case of poetry the end of a line is significant.

The importance of layout is illustrated by the following description of a catalogue entry given by Cox, Deus, Dolby of the University of Newcastle Upon Tyne (8):

> "The number in the top left-hand corner is the class number, whereas the numbers appearing in the collation are the number of pages and the height of the book."

The user would be expected to distinguish these. Librarians and library users are able to identify the type of information by its position in the entry and by the fact that it conforms to certain conventions familiar to libraries.

As far as preparing a bibliographic description for presentation to a computer is concerned, the problem is that one must either define the order in which the items are given and how they are separated, as in the Newcastle project, or one must have warning symbols for each item as in the Edinburgh project., For example, in the latter case, one might use <A> for author and <D> for date.

Bibliographic description

A scholar requiring to analyse a piece of text by computer and to store the text in the form of a file may wish to preserve certain extra pieces of information relating to the text. He might want to retain the whole bibliographic description or just parts of it such as the author's name, the date or place discovered or the rhymeing scheme of poetry. A particular instance of this is the work done by a group at Edinburgh University when analysing Scottish text. (9) They wanted to form slips of paper for each word of a text on which was recorded a three line context plus other relevant information.

This bibliographic description takes a form similar to that on a library card and as such sets up problems different to those involved in representation of the text itself. In its present form, a library card is such that the layout and position on the card gives information about the book. For example, the use of punctuation, newline and space imply differences in the different items of information. This is in direct contrast to the use of, for instance, newline in prose which simply means that there is no more room on this line.

Although, in the case of poetry the end of a line is significant.

The importance of layout is illustrated by the following description of a catalogue entry given by Cox, Deus, Dolby of the University of Newcastle Upon Tyne (8):

> "The number in the top left-hand corner is the class number, whereas the numbers appearing in the collation are the number of pages and the height of the book."

The user would be expected to distinguish these. Librarians and library users are able to identify the type of information by its position in the entry and by the fact that it conforms to certain conventions familiar to libraries.

As far as preparing a bibliographic description for presentation to a computer is concerned, the problem is that one must either define the order in which the items are given and how they are separated, as in the Newcastle project, or one must have warning symbols for each item as in the Edinburgh project., For example, in the latter case, one might use <A> for author and <D> for date.

The terminator for each is the end of the line, and so the layout of the information is very important.

As the suite of programs at Glasgow expands, this problem will be tackled in more detail. The system will be such that the bibliographic description, and the process of dealing with it, can be inserted later, the text being dealt with on its own at the moment.

The Text itself

Text can be prepared and stored in many forms and in many codes. For example, it can be stored in the form of punched cards, or 5, 7 or 8-hole paper tape, or on magnetic tape. Depending on the installation, these also have different codes. As a result, there are many different character sets and different sizes of sets. With 5-hole tape there are only 64^{2} different characters available, whereas with 8-hole tape there are 128^{6} characters available.

Our aim was to set up a system which could accept text in any form and in any code. This meant that we had to define several different groups of characters in text so that there would be a correspondence between a text in one code and a text in another code; and so that the programs

could deal with several languages with different numbers of letters. e.g. Russian and Greek. This will be more clearly defined in the next chapter.

To begin setting up the general system, we decided to write the programs to accept our own KDF9 8-hole paper tape code and then to extend them to accept all codes.

Paper tape code

The paper tape system which was used for the punching of the texts was the 8-hole system. This allows up to 64 different bit patterns from the six information channels, but with the addition of a 'case' character, this is expanded to allow us the use of capital and small letters and a wide variety of punctuation and other symbols.

A table of the KDF9 8-hole paper tape code is given in Appendix 1.

Code Conventions

In the early stages of writing this suite of programs, a code convention had to be stipulated. It was decided that for the punching of the text itself, the following

characters would be necessary:-

A - Z

a - z

:

;

.

,

!

?

/ \ accents

- hyphen

To allow for the inclusion of references, a 'reference warning' character would also be necessary, plus the digits 0 - 9 and a separator. The plus sign (+) was adopted as the warning character and the full stop (.) as the separator.

Thus the representation in the KDF9 paper tape character code is as follows:-

Text	KDF9 code
A - Z	A - Z
a - z	a - z
0 - 9	0 - 9
.	.
,	,
:	:
reference warning character	+

Text	KDF9 code
;	;
!	↑ (exponentiation)
?	& (pound sign)
/	< (less than)
\	> (greater than)
-	- (minus)
(	(
)	)

The bracketing (and) was included to improve the original copy of the text, but for the purposes of the analysis, they are disregarded.

We shall discuss later how this can be generalized to all codes (page 29).

Line printer code

The most usual form of output from a computer is the high speed line printer. This is a device for printing lines of information at a speed of 600 lines per minute. As one line can hold 120 characters, this means that 72,000 characters per minute can be output which is very fast compared with the paper tape punch which punches at a speed of only 110 characters per second, and this has then to be printed on a flexowriter which is still slower.

The printer at Glasgow will recognize 51 printable characters (A - Z, 0 - 9 and a selection of punctuation marks) and a space mark on the paper. A copy of the code can be found in Appendix 1.

As can be seen by a comparison of the two codes, quite a few characters are not available on the line printer, and because of this, the type of information output via the line printer is limited.

For the purpose of this thesis, the paper tape punch has been used as the output medium in order to print the results on suitable paper, but in general, it is the line printer which is used, as the amount of text being processed at any one time would be too great to cope with on paper tape.

The KDF9 computer at present installed at Glasgow University is an electronic digital computer and has a core store of 32K 48 bit registers. There are time sharing facilities in operation enabling up to four programs to be stored in the machine at once, control passing from one to another whenever time might otherwise be wasted, according to the priority grading of each program. A great deal of the time used in textual analysis is in the input and output of text and results.

If these can be transferred onto magnetic tape and printed down later at a low priority level in the machine, more efficient use of the computer will be made.

CHAPTER 3

THE SUITE OF PROGRAMS

Development

The aim of the suite of programs developed at Glasgow University was to produce word lists to solve a variety of problems. We hoped to make them useful to scholars involved in research into texts written in a variety of languages. Enquiries were made as to the requirements of several groups of people and it was decided that the following word lists would cover their needs:-

1. Alphbetical list of words of text.
2. Alphabetical list with references.
3. Alphabetical list with references and frequency.
4. Word list in order of frequency.
5. Word list in order of word length.
6. Word list in order of word endings.

Variations on these were also suggested as well as correction facilities and permanent storage of the text.

Our aim was to make the system so flexible that any other type of analysis of the text could be obtained by

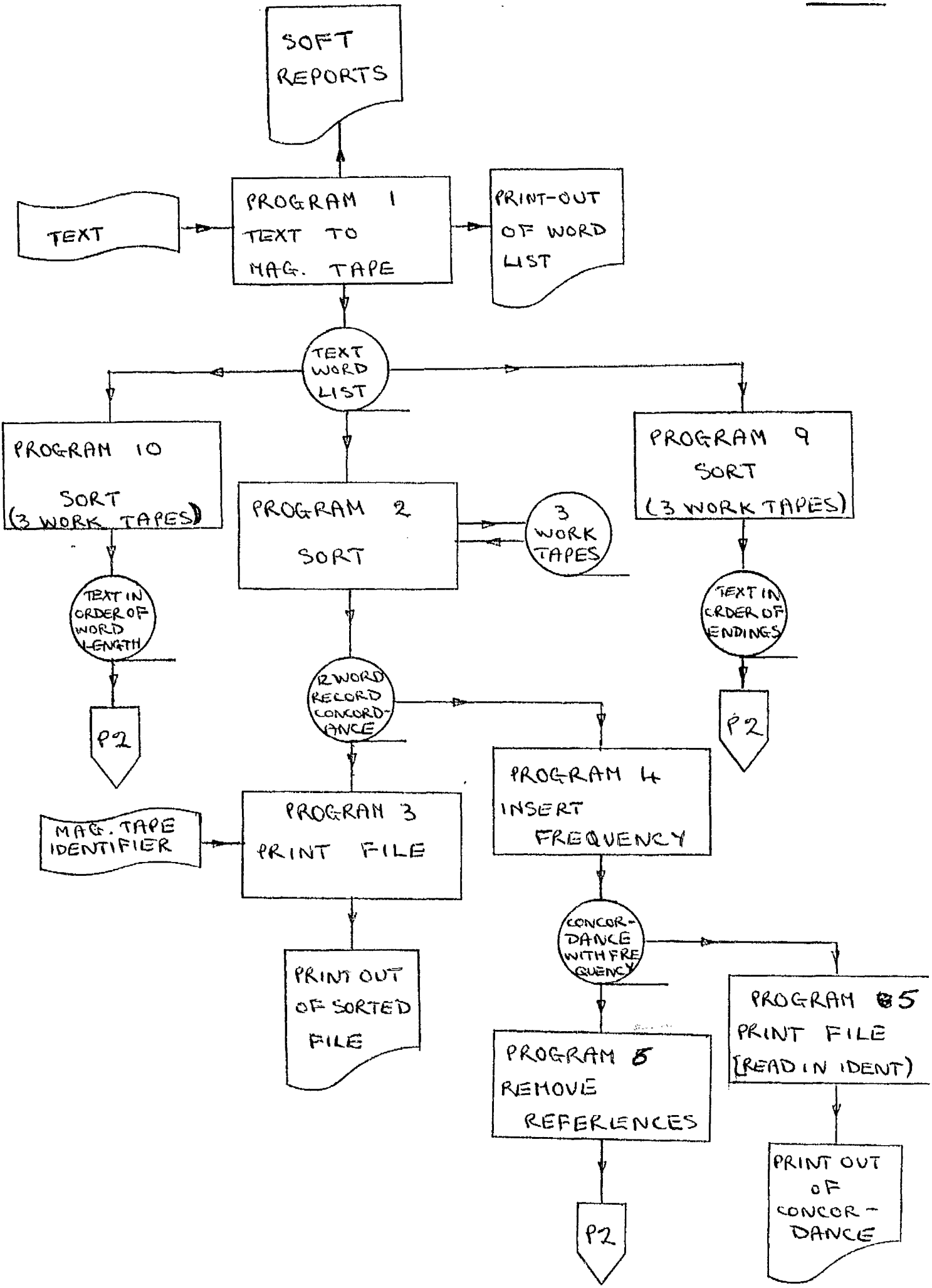
the addition of small pieces of extra programming. To do this we decided to form a version of the text consisting of one record for each word or punctuation symbol of the text. The inclusion of punctuation means that anyone interested in analysis requiring a knowledge of the frequency of the use of punctuation (e.g. distribution by sentence length) has it available in the listing of the text and does not have to go back to the original form.

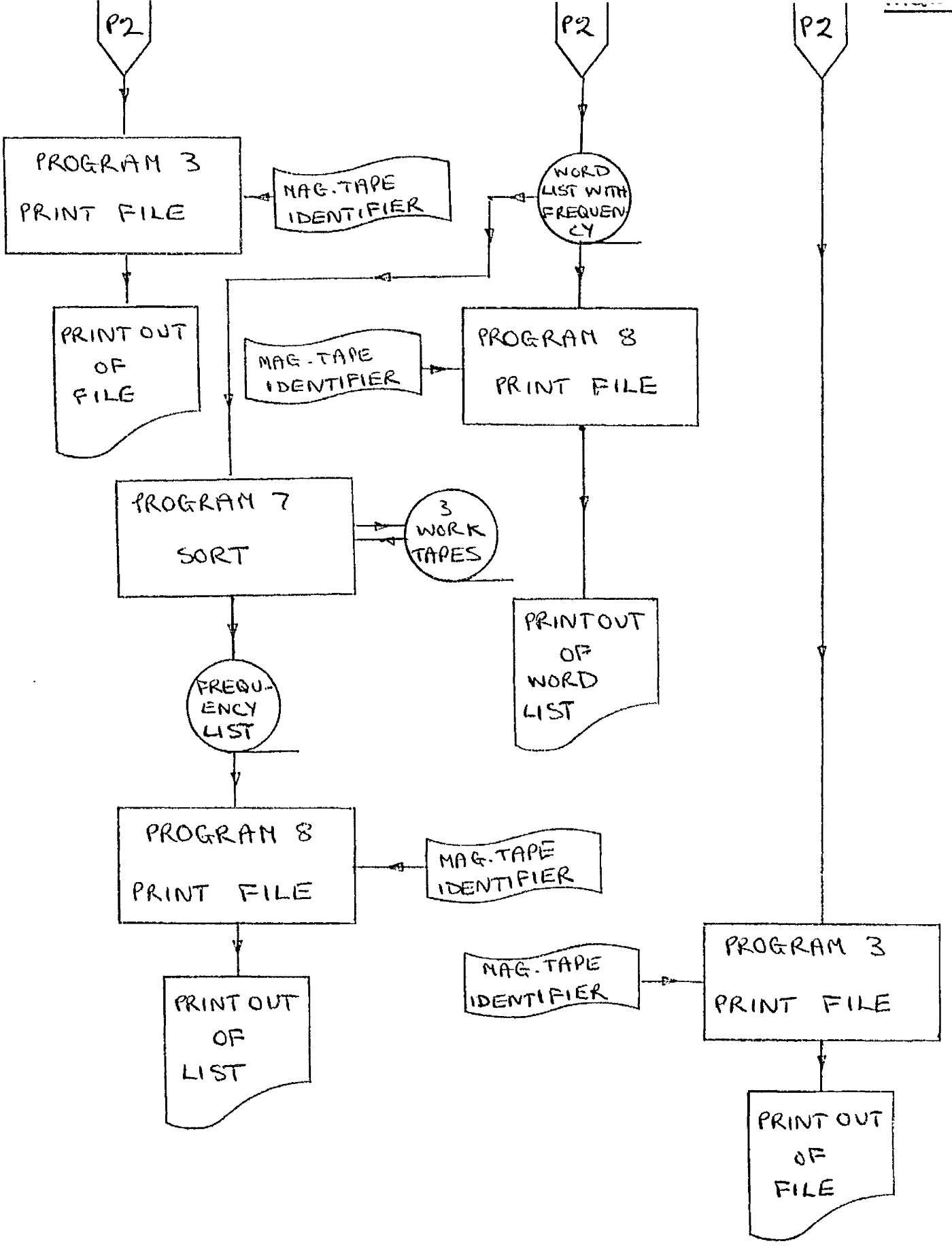
This first listing of the text would consist of a list of records of the words in order as they occur in the text with references. This could then be sorted into alphabetical order to give list 2 as given on page 20. By using another program this could be altered to give list 3, which could then be sorted to give list 4. The original list could also be sorted to give lists 5 and 6.

On examining some of the output from a typical run, we saw that a great deal of the output consisted of repetition in the cases of a word which occurs hundreds of times in the course of a large piece of text. To some people, the references are very important and at a later stage perhaps only the frequency might be important. Hence we decided to introduce variations on

the original ideas. One form of the text would give one complete record for each word followed by all the references of the different occurrences of the word without repetition of the rest of the record. Another form would be to remove all of the references but include only one record for each different word along with the frequency.

These various listings were obtained from the text by the suite of programs which will now be explained.


SOFT REPORTS
TEXT
PROGRAM 1 TEXT TO MAG. TAPE
PRINT-OUT OF WORD LIST
TEXT WORD LIST
PROGRAM 10 SORT (3 WORK TAPES)
PROGRAM 2 SORT
3 WORK TAPES
PROGRAM 9 SORT (3 WORK TAPES)
TEXT IN ORDER OF WORD LENGTH
TEXT IN ORDER OF ENDINGS
P2
P2
2 WORD RECORD CONCORD-ANCE
PROGRAM 4 INSERT FREQUENCY
MAG. TAPE IDENTIFIER
PROGRAM 3 PRINT FILE
CONCOR-DANCE WITH FREQUENCY
PRINT OUT OF SORTED FILE
PROGRAM 5 REMOVE REFERENCES
PROGRAM 5 PRINT FILE (READ IN IDENT)
P2
PRINT OUT OF CONCOR-DANCE


P2
P2
P2
PROGRAM 3
PRINT FILE
MAG. TAPE
IDENTIFIER
WORD
LIST WITH
FREQUEN
CY
PRINT OUT
OF
FILE
MAG. TAPE
IDENTIFIER
PROGRAM 8
PRINT FILE
PROGRAM 7
SORT
3
WORK
TAPES
PRINT OUT
OF
WORD
LIST
FREQU-
ENCY
LIST
PROGRAM 8
PRINT FILE
MAG. TAPE
IDENTIFIER
PRINT OUT
OF
LIST
MAG. TAPE
IDENTIFIER
PROGRAM 3
PRINT FILE
PRINT OUT
OF
FILE

The programs involved

The programs of the suite are as follows: (The numbering of the programs is as given in the flow diagram on previous page.)

PROG 1	DD026AE00UPU	-	read text and set up file.
PROG 2	SORTPROGRAM 7	-	sort original file into alphbetical order.
PROG 3	DD026AF00UPU	-	print 12 word records
PROG 4	DD026AG00UPU	-	insert frequency
PROG 5	DD026AF01UPU	-	print 13 word records
PROG 6	DD026AL00UPU	-	remove references
PROG 7	SORTPROGRAM8	-	sort frequency list into order of frequency
PROG 8	DD026AF02UPU	-	print 11 word records.
PROG 9	SORTPROGRAMO	-	sort original file onto alphabetical order of endings of words.
PROG 10	SORTPROGRAMA	-	sort original file into word length order.

A flow diagram of the complete suite can be found on the previous page. Each program will be explained briefly here although a detailed specification can be found in Appendix 3.

Program to read text and set up initial file

This is the main program in the suite. It was originally written to accept data in the form of paper tape and to output the results to the line printer. It has since been extended so that the results are output to the line printer and magnetic tape in the form of a file to be processed later. This file consists of a record of 12 words in length for each word or punctuation symbol of the input text. The details of the record will be described later in this chapter.

This program is written in Usercode and contains only original programming apart from the obligatory use of the standard magnetic tape handling routines. Their use was necessary because, in order to make use of the Sort Program Generator facilities of the KDF9, the files must be set up using these routines. (10)

Program to insert frequency

This program accepts input on magnetic tape in the form of a file of records of 12 words in length. It then inserts a thirteenth word which is the frequency count and outputs the new records onto another magnetic tape file. This is again written in machine code

with the restrictions explained for the first program.

Program to remove references

This program accepts input from one magnetic tape, processes it and then outputs the results to another tape. The input is in the form of a file of 13 word records and the output is in the form of 11 word records. The processing consists of removing the 10th and 11th words of the record, i.e. the reference as described later. This is also written in machine code with the above limitations.

Program to print a file of twelve word records (Eleven and thirteen word records.)

This is a program for printing down the contents of a magnetic tape file written by one of the other programs of this suite. The actual tape from which reading has to take place is specified at run time by meansof a data tape. The tape consists simply of the following:-

CN CRLF
<8 character identifier> →→

e.g. CN CRLF
DG010052 →→

The size of the record is 12 words, 13 words or 11 words

for DD026AF00UPU, DD026AF01UPU and DD026AF02UPU respectively.

Sort programs

There are four sorts used as part of this suite of programs. They are generated using a standard SORT PROGRAM GENERATOR (S.P.G.) the full description of which is available. (10) In order to generate a sort program, a parameter tape must be prepared and fed as data to the S.P.G. This data gives information as to what type of sort is required.

Alphabetical sort

This accepts as input the file set up by DD026AE00UPU, i.e. a file of records of 12 words in length. It then sorts this into alphabetical order according to the beginnings of the words. If these are the same, other keys are given in order i.e. hyphen marker, accent marker, word endings and finally reference so that the records of the same word or punctuation symbol are given in order of occurrence.

Frequency sort

This sort accepts as input the file set up by the program DD026ALOOUPU, i.e. a file of 11 word records. It then sorts the file into order of frequency and alphabetically within frequency so that all those words occurring once are listed alphabetically followed by those occurring twice and so on.

Sort into order of endings of words

This sort takes as input the original file set up by DD026AEOOUPU. This is similar to SORTPROGRAM 7 except that in this case the sort is in order of the endings of the words.

Word length sort

This sort takes as input the original file set up by DD026AEOOUPU. It is again similar to SORTPROGRAM 7 except that the order is now word length, but alphabetical within word length.

The record itself

At the beginning of this chapter we mentioned that the lists would consist of records. Once the type of lists required had been defined, it was then necessary to decide upon the contents of the record.

Firstly, of course, the word itself had to be included and because of the interest in word endings shown by certain groups, it was also thought necessary to include the word spelt backwards. Other important items are

(i) a count of the number of letters in the word, and

(ii) its reference in the text. To allow for extra items such as hyphens and accents, some indication of their occurrence would have to be included. A frequency count was to be inserted into the record at a later stage.

Once the contents of the record had been decided, there then arose further problems. We had to decide whether to use a variable length record or whether to fix each item at a certain size. We chose fixed length record because we wanted to make quite extensive use of the standard facilities of the KDF9 system (e.g. Sort Program Generator and Report Program Generator). This led yet again to a problem: if the record is fixed in length, then the word length would have to be limited. To allow for languages with long words, an upper limit of 24 letters was set. This, however, does not mean that longer words would cause a failure, but that the extra letters would be lost in the word spelt forwards and the first few letters lost

in the word spelt backwards. For the reference, it was decided that one consisting of four parts would be sufficient. This would start with two variable references. By variable we mean that they could be any subdivisions of the text that the user might require. The last two parts would be line number and word number.

When all of these are combined, a twelve word record emerges as detailed later in this chapter.

The text itself

One of our main aims was to make the system such that it could accept text with different character sets and conventions. Thus, in defining our original character set, we tried to allow for peculiarities which might arise in languages other than English. For instance, French and Gaelic, as well as other languages, have accents. Also, some letters in Greek are represented differently at the end of a word from their representation when at the beginning or in the middle of a word. For instance, there are two forms of s and two forms of omega.

In English, we are only concerned with 26 letters,

punctuation symbols and hyphens. However, other features do occur in other languages and of course, English text can contain a number of foreign words and expressions with accents etc. Needless to say, it would be "impossible" to account for each one separately so we decided to define several groups of characters such as letters, digits and punctuation but also allow extra groups such as hyphen or two accents which would allow users involved with other languages to fit their own symbols into one or other of the categories. For example, in the Greek alphabet, there are fewer distinct letters, but, as already mentioned, some have different spellings depending on the position in the word. This can be accounted for by assigning another of the letter set to the second form of the letter.

To illustrate the definition of the English character set, there follows a Backus Normal Form description of that set. (11)

Text Specification

A file consists of a number of lines which are numbered in sequence within any one page, (or other subdivision of the text).

⟨typical line⟩ ::= ⟨word⟩⟨word separator⟩⟨word⟩ ....
....⟨word⟩ ⟨word separator⟩
⟨end of line⟩ ⟨newline⟩

⟨end of line⟩ ::= ⟨word⟩/
⟨word⟩ ⟨word separator⟩ ⟨reference⟩/
⟨part word⟩ ⟨word continuation symbol⟩/
⟨part word⟩ ⟨hyphen⟩ ⟨reference⟩

⟨reference⟩ ::= ⟨major⟩⟨separator⟩⟨minor⟩⟨separator⟩
⟨line number⟩

⟨major⟩ ::= ⟨reference warning character⟩ ⟨decimal integer⟩

⟨minor⟩ ::= ⟨decimal integer⟩

⟨line number⟩ ::= ⟨decimal integer⟩

⟨decimal integer⟩ ::= ⟨digit⟩/⟨digit⟩⟨decimal integer⟩

⟨digit⟩ ::= 0/1/2/3/4/5/6/7/8/9

⟨separator⟩ ::= .

⟨reference warning character⟩ ::= +

⟨word⟩ ::= ⟨letter⟩/⟨word⟩⟨letter⟩/⟨word⟩⟨word
continuation symbol⟩⟨word⟩

⟨letter⟩ ::= a/b/c/.../y/z /⟨accent⟩⟨letter⟩

⟨word continuation symbol⟩ ::= -

⟨word separator⟩::= ⟨space⟩ / ⟨sentence marker⟩ /
⟨other punctuation⟩/
⟨word separator⟩ ⟨word separator⟩/
⟨newline⟩

<sentence marker> ::= . / ? / !

<other punctuation> ::= , / : / ; /) / (

<space> ::= ␣

<accent> ::= null (There are no accents in English)

<part word> ::= <letter string><word continuation symbol>

<letter string> ::= <letter> / <letter><letter string>

<word> ::= <letter string> / <part word><letter string> /
<part word><part word><letter string>

Record description

From the text as described in the previous section the basic record is obtained. The exact format of this record will now be explained and illustrated.

The twelve word record is made up as follows:-

words 1 - 3 = word forwards
words 4 - 6 = word backwards
word 7 = letter count
word 8 = hyphen marker
word 9 = accent marker
words 10 & 11= reference
word 12 = CRLF

In the case of a punctuation symbol, the record takes the form :-

words 1 - 3 = decimal value of character
words 4 - 6 = punctuation character
word 7 = zero
word 8 = not used
word 9 = not used
words 10 & 11= reference of previous word
word 12 = CRLF

To illustrate the following description an example of each type is given below.

Examples for "nevertheless" occurring as the first

word and followed by a comma in the fourth line of the second page of the ninth chapter of a book:

Sample record for a word

FORWARDS	BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	R2	LINE	WD
NEVERTHELESS	SSELEHTREVEN	12			9	2	4	1

Sample record for a punctuation symbol

31	,	0			9	2	4	1

Word forwards

This is simply the letters of the word set out in words 1 to 3 of the record, with the first letter in the top end of word 1.

In the case of punctuation, the decimal value of the symbol is set in the lower end of word 1.

Word backwards

This is similar to above with the last letter of the word in the top end of word 4.

For punctuation, the actual symbol is set in the lower end of word 4.

Letter count

This consists of the decimal value of the count of the number of letters in the word, or, in the case of punctuation, the value zero.

Hyphen marker

This consists of the decimal value of the count of the number of letters which precedes the hyphen. Up to two hyphens are allowed and the hyphen marker will then consist of two integers. A hyphen can occur under two circumstances. The first is when a natural break occurs as in the case of forget-me-not, or, secondly, when a word is split between two lines because the line is too short to take the whole word. A distinction is made between the two different causes of a hyphen by including a minus sign before the count referring to an example of the second type.

e.g. for forget-me-not, word 8 of the record would be set out as follows:-

␣␣␣6␣␣8

and in the case of : .... NEVERTHE-
LESS ...

it would be :-

␣␣␣␣␣-8

Of course, a mixture of these might occur.

e.g.

ULTRA-VIO-

LET

in which case word 8 would take the following form:-

␣␣␣5 ␣␣ -8

A further example of what might happen is the case of an actual hyphen occurring at the end of a line,

e.g. forget- me-

not.

This would appear in the record as

␣␣␣6 ␣␣ -8

Examples like this would have to be dealt with later by someone working through the word list and noting any such occurrences. This could be dealt with by a program only if there was a dictionary of all such legitimate words and a search of this dictionary would be required. This is a linguistic problem and it was felt that, because of the small probability of such things happening, it need not be accounted for in the program. It might be observed that in some tape-punching conventions, split words are allowed.

At a later stage, once individual records can be accessed in an on-line system, problems like this can

be overcome by being able to delete the hyphen marker and insert the correct one.

Accent marker

This also takes the form of the decimal value of a count, but in this case it is the count of the letter to which the accent refers.

In the preparation of the text, the accent is printed before the letter to which it refers. Also, in this case, up to two accents in one word are allowed, each being represented by the letter count of the letter to which it refers. If it happens, as in Greek, that a word contains three accents, then the count of the first will be lost and the other two retained. However, this occurs only very rarely and is not really relevant. Another possibility is that there may be two accents on the one letter. Again, this occurs, as far as I know, only in Greek and is not accounted for in the programs.

An example of the use of accents is:

"élève" would be printed as <EL>EVE and word 9 would be set out as follows: ␣␣␣1␣␣␣3

No distinction is made between the two alternative types of accents.

Reference

Words 10 and 11 contain the reference of the word or punctuation symbol and is in the form of a four part reference:

word 10		word 11	
REF.1	REF.2	LINE	WORD

The line count is updated when a CRLF is encountered, and the word count is updated at the start of a new word. When a CRLF is found, the word count is reset at zero. The first two references must be updated in the text itself using the reference warning character followed by a reference (as described in P25 on page 74). The values in REF1 and REF2 and LINE are replaced by those being read in and WORD is reset at zero because the reference occurs at the end of a line. Further CRLF's are ignored until the start of a new word.

Finally, word 12 contains a CRLF to complete the record so that on output a newline will be taken.

Soft reports

During the initial processing of the text, there might occur fault conditions of which warning must be given. Those which are accounted for are as follows:-

1) If an invalid character is detected.
In this case the message

INVALID CHARACTER FOUND IN WORD

is sent to an output stream to be printed down at a later stage. This is followed by the word in which the character has been found (if it occurs in a word)
Then the message

AT REFERENCE,

followed by the reference, is also output providing an indication of the location of the error.

2) If a long word (i.e. a word of more than 24 letters) is read. In this case, the message

LONG WORD FOUND,

followed by the complete word, is output. Thus although the complete word is not retained on the file, a copy of it is available after input.
In the current version of the program when such a word is encountered no reference is given as in the other cases. However, this would be a very simple alteration.

3) If an invalid character has been found in a reference i.e. a character which is not a digit, a separator or a terminator.

In this case, the message

INVALID CHARACTER IN LINE REFERENCE

is printed.

This is actually carried out by the reference reading routine. (i.e. F25 on page 74)

CHAPTER 4

CONCLUSIONS AND IMPLEMENTATIONS

Results

In order to illustrate the use of the suite of programs, they were applied to the poem "The Daffodils" by William Wordsworth. A copy of this can be found on page 85. A poem was chosen because it includes a number of verses which involves the use of three of the four possible references. The poem was punched on paper tape in accordance with the conventions defined in appendix 3. A print out of the poem obtained in this manner can be found on page 86. This data tape was then used as input to the suite as shown in fig.2 and the results given in appendix 5 were obtained.

List 1 consists of the basic list of the records of the words and punctuation symbols of the text as they are read in. A heading is output at the beginning of each new page of line printer paper and this defines what each item is. This is obtained as

a result of PROGRAM 1.

List 2 is the concordance obtained by sorting List 1 into alphabetical order. The punctuation symbols are separated off from the words and are printed at the beginning.

List 3 is the concordance plus a frequency count. The repeated records of "same words" have been removed except for the references.

List 4 is a list of the words and punctuation symbols of the text in alphabetical order with frequency but without references. This, it can be seen, is a much less bulky list.

List 5 is the list 4 sorted into order of frequency given in alphabetical order within frequency. This could be useful in a larger text for locating mis-spellings and rare words, although in a short text, such as used here, most of the words occur only once.

List 6 is the original word list sorted into order of the endings of the words. This is useful to those interested in the endings rather than the beginnings of words. This could also be extended to include frequency lists etc. as for the word forwards lists by making slight alterations to the programs and sorts.

List 7 is again the original word list sorted, but this time into order of word length and alphabetically within this.

As can be seen from all of these lists, the references are given in order through the text so that it is simple to work through the text and locate each occurrence.

Present use of system

At the moment, the suite of programs is being used by the Celtic Department at Glasgow University. This Department is carrying out work on the compilation of an Historical Dictionary of Scottish Gaelic. This involves locating every occurrence of most words from many works. This will then permit them to deduce meanings in the normal manner. The use of computers is new to them and it has taken several months to work out what information they required. A great deal of time has been spent in the punching of the text and they are at present (September 1968) examining the output they have received.

Effectiveness of system

We have now obtained all of the lists we aimed at:

1. Alphabetical list of words
2. Alphabetical list with references
3. Alphabetical list with references and frequency
4. Word list in order of frequency
5. Word list in order of word length
6. Word list in order of word endings

This was sufficient to carry out the work required for the Gaelic research.

Because we have arranged the contents of a typical record, (ref. page 33), so that the items word forwards, word backwards, letter count, and frequency can be used as keys for sorting processes of the Sort Program facilities, we may also readily obtain other word lists should they be required.

Various types of input

As the programs stand at the moment, they can accept text prepared in the KDF9 8-hole paper tape code. Another of our aims was to make the system such that it could accept different codes. This

flexibility can be achieved in several ways. One is to have a small program to carry out a code conversion before submitting the text to the suite. This could be done in a general way by feeding the new character set in as data to a standard conversion package, or it could be done by writing a different program for each different code. Although this would appear wasteful, in fact the programs would all be more or less the same with a few changes being made for each code. The alternative to this code conversion is to replace the jump table in the main subroutine of PROGRAM 1 (i.e. P22) by another table such that the instructions, which cause reference to be made to the table, will cause control to pass to the correct reference in the program. For instance, for the Russian language, there are more than 26 letters required. Hence, the necessary number of extra characters will be chosen and treated as letters. This will be done by assigning to the appropriate position in the table the address of reference 32 which deals with letters. Similarly if there are less than 26 letters required, those which are not in use will be treated as invalid

characters and control will pass to reference 14.

At the moment, however, due to a slight oversight in planning, there are two parts of the program which do not lend to generality. In order to permit the extension to the programs to be made, these will now be described.

Referring to printouts of the programs in appendix 4, at reference 22 of P22, there is an independent fetch of a character which is then tested for the values 0 and 2 which are space and CRLF respectively. This would not work with a code in which the values for space and CRLF are different. Hence, these two tests would also need to be altered.

The second place where generality is lost is in P25. In this case, the values of the digits 0 - 9 are assumed to be as in the KDF9 code (appendix 1.) Again, the characters . (full stop), space, CRLF are taken as having the values in the KDF9 code. In order to make this subroutine general, it would either have to be replaced for codes in which these characters are different, or it would have to be rewritten with a jump table (as in P22) which could be replaced for different codes as in P22.

Another aid to generality would be to set up two separate tables for the two separate 'cases' (i.e. CASE SHIFT and CASE NORMAL in KDF9 code.) This would then make the transformation to accept the Greek 5-hole tape quite simple. This code consists of 64 characters, 32 in LETTER SHIFT and 32 in FIGURE SHIFT. To replace the single table by two tables, a test would have to be made on the 'case' of the character by means of a marker which would have the value 1 or 0. This would then determine which table was to be referred to. Again this can be quite easily done.

Greek 5-hole tape

To illustrate the changes required in the jump table for P22, a copy of the Greek 5-hole paper tape code is given on Page 49 with the corresponding tables which would be required. As in the present program, each address is set into one half of a V-store. The marker (V2) which is set at references 9 and 10 would be used to indicate which table is to be used for the character. In order to impliment this code, the other alterations already mentioned would have to be made.

A problem arises as regards what one does with CRLF in this code. The trouble is that whereas in the KDF9 code, one character is used to indicate a carriage return and a line feed, in the 5-hole paper tape code two separate characters are used. If both were treated as CRLF, this would cause faulty updating of the line number. Another alternative is to use one as a word ~~terminator~~ separator (i.e. CR), and the other (i.e. line feed) as meaning CRLF. This fits in effectively with the B.N.F. description text (page 31).

Another way of dealing with it would be to write a small piece of program which would test for a CR being followed by a LF. If this were the case, then the two would be treated as one character and therefore control would pass to reference 20 in P22. If, however, this was not the case, the function of CR would be as a word terminator and then perhaps treated as an invalid character. If an isolated LF was detected, similar action could be taken. Alternatively, it could be assumed that the person punching the text always punched both characters. In this case, it would be simplest to ignore one of the characters and treat the other as the CRLF character.

Greek paper tape code

Tape	Value	Letters	Figures
		FIGURE	SHIFT
.	0	α	1
. 0	1	β	2
.0	2	γ	(
.00	3	δ	4
0.	4	ε	)
0. 0	5	ϛ	– (not used)
0.0	6	ζ	7
0.00	7	θ	8
0 .	8	ι	;
0 . 0	9	κ	,
0 .0	10	λ	– (minus)
0 .00	11	μ	/
00.	12	ν	L.F.
00. 0	13	ξ	.
00.0	14	ο	– (not used)
00.00	15	π	0
0 .	16	ρ	– (not used)
0 . 0	17	ϟ	– (not used)
0 .0	18	τ	3
0 .00	19	υ	– (not used)
0 0.	20	φ	5
0 0. 0	21	χ	6
0 0.0	22	ψ	– (not used)
0 0.00	23	ω	– (not used)
00 .	24	ϡ	9
00 . 0	25	(not used)–	+
00 .0	26	LETTER	SHIFT
00 .00	27	.	.
000.	28	space	space
000. 0	29	(not used)–	C.R.
000.0	30		ERASE.
000.00	31		

Jump Table 1		(letter shift)	2 (figure shift)
0	-	AR 10	AR 10
1	-	AR 32	AR 14
2	-	AR 32	AR 14
3	-	AR 32	AR 12
4	-	AR 32	AR 14
5	-	AR 32	AR 12
6	-	AR 32	AR 14
7	-	AR 32	AR 14
8	-	AR 32	AR 14
9	-	AR 32	AR 12
10	-	AR 32	AR 12
11	-	AR 32	AR 19
12	-	AR 32	AR 14
13	-	AR 32	AR 20
14	-	AR 32	AR 12
15	-	AR 32	AR 14
16	-	AR 32	AR 14
17	-	AR 32	AR 14
18	-	AR 32	AR 14
19	-	AR 32	AR 14
20	-	AR 32	AR 14
21	-	AR 32	AR 14
22	-	AR 32	AR 14
23	-	AR 32	AR 14
24	-	AR 32	AR 14
25	-	AR 32	AR 14
26	-	AR 14	AR 18
27	-	AR 9	AR 9
28	-	AR 12	AR 12
29	-	AR 6	AR 6
30	-	AR 14	AR 12
31	-	AR 14	AR 14

File structure

It was also our aim to set up a file structure which would allow a user to manipulate the text to his own requirements.

The file structure is now available but the system associated with it and required to manipulate it is not yet complete. This is however accounted for in the scheme since the scheme has been developed so that once the on-line system at Glasgow (COTAN) is available, our file system can be incorporated. During the period that the programs were being developed there was no on-line desk system available and hence we have been to date unable to make use of such a facility.

We envisage the situation where various departments have on-line connections with the computer through which they can carry out any particular manipulations of their texts. Certain facilities involved in the suite of programs will be available direct to the user as standard routines. This will avoid the necessity for an intermediate programmer to carry out the minor pieces of programming involved.

REFERENCES

(1) "The Computer in Literary Studies". A.Q. Morton Proceedings of the IFIP Conference at Edinburgh, August, 1968.

(2) Modern Language Review 57-1962, Wisbey R. Page 161 "Concordance making by electronic computer."

(3) "A concordance to the works of Horace", compiled and edited by Lane Cooper (The Carnegie Institution, Washington, 1916).

(4) "The Philosophical Journal" Vol.3, No.2 pp 129-148, 1966. "On Certain Statistical Features of the Pauline Epistles" M. Levison, A.Q. Morton and W.C. Wake.

and, (4) The Journal of the Royal Statistical Society Series A (General). Vol.128, Part 2, 1965, pp 169 - 233. "The Authorship of Greek Prose."

(5) Inference and Disputed Authorship: The Federalist Mosteller, Frederick and Wallace, David L. Addison-Wesley, Reading, Mass., 1964.

(6) "Computers & the Humanities" 1, 3 (Jan. 1967) The Computer & Plato's 7th letter." Morton, A.Q.; and Winspear, A.D.

(7) "A Statistical Method of Determining Authorship" Ellegard, A. Göteborg, 1962.

(8) "The Computer and the Library" - W.S.M. Cox, J.D. Deus, and J.L. Dolby. University of Newcastle upon Tyne Library, 1966.

(9) Edinburgh University Project on preparing an "Dictionary of the Older Scottish Tang." Editor Jack Aitken.

(10) E.E.L.M. KDF9 Sort Generator Users Manual.

(11) "Readings in Automatic Language Processing" Editor, D.G. Hays, Chap. 2 "Specification Languages for Mechanical Languages and their Processors - A Baker's Dozen" (Saul Gom) Sec. 4.

(12) Davis, G.M. "The English Electric KDF9 Computer System," Computer Bulletin 4, 3, pp 119-120.

APPENDIX 1

Code Tables.

KDF9 P.T. CODE

Decimal Value	Function	Decimal Value	Symbol Normal/Shift	
0	SPACE	32		
1		33	A	a
2	CRLF	34	B	b
3	PAGE THROW	35	C	c
4	TAB	36	D	d
5		37	E	e
6	CASE SHIFT	38	F	f
7	CASE NORMAL	39	G	g
8		40	H	h
9		41	I	i
10		42	J	j
11		43	K	k
12		44	L	l
13		45	M	m
14		46	N	n

Decimal Value	Symbol Normal/Shift		Decimal Value	Symbol Normal/Shift	
15	/	:	47	O	o
16	0	↑	48	P	p
17	1	[	49	Q	q
18	2	]	50	R	r
19	3	<	51	S	s
20	4	>	52	T	t
21	5	=	53	U	u
22	6	×	54	V	v
23	7	÷	55	W	w
24	8	(	56	X	x
25	9	)	57	Y	y
26			58	Z	z
27	10	£	59		
28	;	;	60		
29	+	≠	61	→	→
30	-	*	62		
31	.	,	63	DELETE	

KDF9 PRINTER CODE

Octal	Decimal	Printer	Octal	Decimal	Printer
00	0	space	40	32	not used
01	1	not used	41	33	A
02	2	LS	42	34	B
03	3	PC	43	35	C
04	4	not used	44	36	D
05	5	not used	45	37	E
06	6	%	46	38	F
07	7	!	47	39	G
10	8	:	50	40	H
11	9	=	51	41	I
12	10	(	52	42	J
13	11	)	53	43	K
14	12	£	54	44	L
15	13	*	55	45	M
16	14	,	56	46	N
17	15	/	57	47	O
20	16	0	60	48	P
21	17	1	61	49	Q
22	18	2	62	50	R
23	19	3	63	51	S
24	20	4	64	52	T
25	21	5	65	53	U
26	22	6	66	54	V
27	23	7	67	55	W
30	24	8	70	56	X
31	25	9	71	57	Y
32	26	not used	72	58	Z
33	27	₩	73	59	not used
34	28	;	74	60	not used
35	29	+	75	61	end message
36	30	-	76	62	start message
37	31	.	77	63	ignored

APPENDIX 2

THE KDF9 COMPUTER.

As an introduction to the detailed specification of the suite of programs, it will be helpful to give a brief description of some of the more important features of the hardware and logic of the KDF9 computer, although a full description is available (ref.12).

The English-Electric KDF9 Computer is a 48-bit, parallel arithmetic machine, with a 6μsec core cycle time.

Its particular features are:-

The Nesting Store

Arithmetic and logical operations are carried out in the nesting store, or 'push-down' store, which is a stack of sixteen 48-bit words. The mode of operation is analogous to that used for bullets in the magazine of a gun. The rule is, therefore, 'first in-last out', except that there are a few instructions deliberately designed to rearrange items in the nesting store. Automatic tests inside the machine check that no more than 16 words have been fetched into the nesting store,

and also that a program does not attempt to remove more words than have previously been put in. A contravention of either of these restrictions leads to the immediate failure of a program.

The Subroutine Jump Nesting Store (SJNS)

The SJNS, a similar "push-down" store, is used automatically by the machine to store the return address whenever a subroutine is entered. Since second or higher order subroutines are quite often needed and since the return address for the last one entered is required first, a nesting store is ideal for this purpose. Sixteen cells are provided in the SJNS but programmers are recommended to restrict their use to fourteen cells, leaving the other two for use by the control program of the machine.

A subroutine return address is referred to as the 'LINK' and this can be accessed from the nesting store.

Q-stores

Connected to the top of the nesting store is a set of sixteen Q stores numbered Q0 to Q15, Q0 having the permanent value of zero. Each of the remaining fifteen consists of a 48 bit fast access register.

They may be used for a variety of purposes such as temporary storage of data or results when their presence in the nest would be inconvenient, and as index registers for modifying main store addresses. A Q-store can be regarded as holding three independent 16-bit binary integers, known as

(a) Counter (C)

(b) Increment (I)

(c) Modifier (M)

This format is used in input and output operations.

Store area locations

The main store is used to accommodate the instructions and the data of the programs. Normally, the first word of the data storage area is referred to in User Code by the symbolic form Y0 and subsequent words are Y1, Y2, ... Other areas can be referred to as YA0, YA1, ... and YB0, YB1 ... and up to YZ0, YZ1,...

It is possible that an area of store may be required as working space by a large subroutine. For this purpose, stores known as W-stores are provided.

It is often necessary for a program to have certain constants during the execution of the program. User Code provides a set of V-stores for this purpose.

APPENDIX 3

DATA SPECIFICATION

In order that a piece of text should be acceptable to the suite of programs as they are at the moment, it must be prepared to the following specification.

Input, can be made in the form of paper tape (8, 7 or 5 hole) or in the form of cards if notice is given to the machine operator. The text itself should be punched according to the specification given in Chapter 3. The first character on the data tape is used as an indication of what the terminating condition will be. This means that when the program detects 6 successive copies of this character, it terminates. e.g. if Z is to be the end of data character, then the first character will be a Z and the text will be terminated by ZZZZZZZ → → . The two end messages are necessary for the input routine.

APPENDIX 4

THE PROGRAMS OF THE SUITE

The use of the programs of the suite has already been discussed in general terms. We give here a more detailed specification with the flowcharts and a print out of the coding. A description, flowchart and the coding of each program is given.

PROGRAM 1 DDO26AEOOUPU

The write file is first opened on a magnetic tape with identifier DG010050. The program then claims the line printer and the paper tape reader. The heading for the soft reports is then output to stream 70 to be printed down later. This heading is:- TEXT ANALYSIS SOFT REPORTS. The first non-blank character is then read from the paper tape and set as the end of data character. From this a word of three copies of this character is set up as the end of data word.

YC1, which contains the counters for references one and two is initially set to contain the characters 00010001, so that the references have the initial value of 1.

This is the preparation necessary before entry is made to P22. Entry is now made to P22 and after each entry the record is stored in a buffer until 720 words (i.e. 60 records) have been set up ready for output to the line printer. At the same time, each record is written onto the magnetic tape file using P31. When 60 records have been set up in the buffer the heading

WORD FORWARDS WORD BACKWARDS LETTER HYPHEN ACCENT REFERENCE COUNT MARKER MARKER R1 R2 LINE WORD

is output to the line printer followed by the 60 records. This occupies one page of the line printer paper.

When exit is made from P22 at EXIT 2, this indicates that the end of the data has been reached. At this point, a new heading followed by the contents of the buffer is output to the line printer. The line printer is then deallocated, the magnetic tape file is closed (an end of file marker being written) and the tape rewound; the paper tape reader is deallocated and the program is finally terminated.

This program uses P20, 21, 22, 23, 24, 30, 31, 32, 55
L2, 37, 61, 78, 79, 89.

Computer requirements:

Nesting store:		3 cells
SJNS	:	1 cell
Q-stores	:	Q1, 4, 5
V stores	:	V0 - 38
W stores	:	W0 - 30
YA stores	:	YA0 - 12
YB stores	:	YB0 - 13
YC stores	:	YC0 - 3
YZ stores	:	YZ0 - 127

PROGRAM 4 DD026AG0OUPU

The file formed by program DD026AE0OUPU has since been sorted into alphabetical order by SORTPROGRAM 7 and the new file is to be found on the magnetic tape with identifier OUTPUT MT. This is then the input to DD026AG0OUPU.

The program firstly opens the read file on magnetic tape OUTPUT MT and the write file on magnetic tape DG01.0064. The first file is a file of 12 words records and the second a file of 13 word records.

The first record is then read into the buffer in YA1 - 12 and word 12 is copied into YA 13. YA 12 is now free to hold the frequency count and is initially set at unity. The next record is then read from magnetic tape and copied into YB1 - 12. YB 12 is copied into YB 13 and YB 12 is set at zero. A comparison is then made between YA1 - 9 and YB1 - 9 i.e. the whole record except the reference. If they are the same, the counter in YA12 is updated. YB1 - 9 are then made zero so that the record in YB1 - 13 consists only of the reference. This is then copied into a Y-store buffer. The process is then repeated by reading another record and comparing with the original.

If, however, the comparison fails, the frequency count is converted to a decimal number using P20, and the 13 word record in YA1 - 13 is copied into the first 13 words of the Y-store buffer (i.e. Y1 - 13). The upper address of this buffer is then set in N1 and the buffer written onto the output tape using P50. The contents of YB1 - 13 (i.e. the new record) are then copied into YA1 - 13 and YA 12 set at unity. The next record is read into YB1 - 13 and compared with the record now in YA1 - 13.

This process is repeated until the end of file marker is detected on the input tape. At this point the read file is closed and the tape rewound. The end of file marker is written onto the output tape, the file closed and the tape rewound. The program then terminates.

This program uses P20, P30, 31, P40, 41, 42, P50 L2, 34, 37, 61, 78, 79, 89.

Computer requirements

Nesting store	:	2 cells
SJNS	:	1 cell
Q-stores	:	Q1, 2, 3, 4, 5.
V stores	:	V0 - 5
W stores	:	W0 - 30
YA stores	:	YA0 - 13
YB stores	:	YB0 - 13.

PROGRAM 6 DD026AL00UPU

This program accepts input from one magnetic tape, processes it and then outputs the results to another tape. The input tape is DG010064 and the output tape is DG010068.

The program removes the references from the records of the file set up by DD026AGOOUPU, so that the output file consists of records in the format:-

WORD FORWARDS WORD BACKWARDS LETTER HYPHEN ACCENT FREQUENCY COUNT MARKER MARKER

i.e. an eleven word record.

The program opens a read file on the input tape (the record size being 13 words) and opens a write file on the output tape (the record size being 11 words). The first record is read and copied into YA1 - 13. YA1 is tested to see whether it is empty or not. If it is, the record is ignored and the next one read in.

If YA1 contains either part of a word or a number corresponding to a punctuation symbol, YA12 is copied into YA 10 (i.e. the frequency) and YA 13 is copied into YA 11 (i.e. the CRLF). The eleven word record in YA1 - 11 is then written onto the output file and the process repeated until the end of file marker is detected on the input tape. At this point the read file is closed and the tape rewound. The end of file marker is written onto the output tape, the file closed and the tape rewound. The program is then terminated.

This program uses P30, 31, 32, P40, 41, 42

L2, 34, 37, 61, 78, 79, 89

Computer requirements

Nesting store	:	2 cells
SJNS	:	1 cell
V stores	:	V0 - V2
W stores	:	W0 - W30
XA stores	:	XA1 - 13

PROGRAM 3 DD026AF00UPU (including
PROGRAM 5 DD026AF01UPU, and
PROGRAM 8 DD026AF02UPU.)

This is a program for printing down the contents of a magnetic tape file written by one of the programs of this suite. The actual tape from which reading has to be taken is specified at run time by means of a data tape. The tape consists simply of the following:-

CN CRLF
<8 character identifier> →→

e.g. CNCRLF
DF010052 →→

The size of the record is 12 words. This, however, can be varied by using a different version of the program. For example, DD026AF01UPU copies down a file

of 13 word records and DDO26AFO2UFU prints a file of 11 word records. In this case, we will deal only with the 12 word record, the only difference being the size of the record in the TDT for P40 and the size of the buffer used for output.

The program firstly claims the paper tape reader and reads in the data. This is then used to claim the particular magnetic tape and to open the read file. The line printer is also claimed.

The first record is then read and copied into a buffer of 1500 words (i.e. 125 records). The next record is read and copied into the next 12 words of the buffer and the process repeated until 125 records have been read. When this has been done, the buffer is output to the line printer. (No heading is output but this would be a very simple alteration.)

This whole process is then repeated until the end of file marker is detected at which point the file is closed, the tape rewound and the line printer deallocated. The paper tape reader is then deallocated and the program terminated by P55.

This program uses P21, P40, 41, 42, P51, 55.
L2, L34, 78, 79, 89.

Computer requirements

Nesting store	:	2 cells
SJNS	:	1 cell
Q-stores	:	Q1 and Q2
V stores	:	V0 and V1
W stores	:	W0 and - 30
YZ stores	:	YZ0 - 127.

P20

This is a subroutine for converting a word in character form into binary or vice-versa. When converting from binary into character form, any of the groups of 6 binary bits which is blank will be converted into the character zero. This will appear as 0 on the output stream if word is output. To avoid left-hand zeroes, as a result of this, the subroutine suppresses these in the binary to character conversion.

Entries 1. To convert to binary from decimal:

entry is at reference 1 (JSIP20)

N1 = word to be converted.

The routine removes the XS 16 bit and using the instruction TOB; converts the word into the binary equivalent.

2. To convert to decimal from binary.

entry is at reference 2 (JS2P20).

N1 = word to be converted.

The routine converts the contents of N1 into character form using the instruction FRB; (from binary). It then inserts the XS 16 bit and suppresses left-hand zeroes.

Exits.

The only exit is by EXIT 1, with the converted word in N1.

Computer Requirements

Nesting store	:	4 cells
SJNS	:	0 cells
Q-stores	:	Q15 (preserved and reset before exit).

P21

This is a subroutine for extracting the next 6-bit character into the buffer, (as described on Page 82) or for restoring a character into the buffer. If the buffer becomes empty, the subroutine refills it.

Entries 1. Initial entry : entry is at reference 0 (JSP21). The subroutine claims the paper tape reader and fills the buffers initially.

2. Read next character:

entry is at reference 1. (JSIP21).

The subroutine tests to see if the buffer is empty. If so, it fills the next and jumps back to read the next character. If not, the next character is removed and set in the top cell of the nesting store.

3. Restore character: entry is at reference 7 (JS7P21)

N1 = character to be restored.

The subroutine replaces in its correct position the character in N1. The counter is altered to account for the extra character now in the buffer.

Exits

There is only one exit and this is by EXIT 1. The nest is empty except in the case of entry via JSIP21 in which case N1 contains the next character.

Computer requirements

Nesting store	:	3 cells
SJNS	:	2 cells
Q-stores	:	Q14, 15 (preserved & reset)
YZ stores	:	YZ0 to YZ127

P22

This subroutine reads the next word or punctuation symbol from the input buffer and sets up a twelve word record as described in chapter 3.

P22

Before entry to the routine, at the start of the main program, the following must be set up:-

V17 P22 must be set equal to 6/1/AYA1 (=V0P22)

Entry must be made to P21 to initialize the paper tape reader i.e. via JSP21.

The first non-blank character from the data tape must be entered in V21 P22, and V20P22 must be set as the end of data word i.e. = ␣␣␣␣␣chchch

e.g. if the end of data character is z,

V21 P22 would be set at ␣␣␣␣␣␣␣z

and V20 P22 ␣␣␣␣␣zzz

Also, V01 must be set as ␣␣␣1 ␣␣␣1

for the first two references, as described under [illegible]. on pages 59 and 60

Entries 1. The only entry is at reference 0 (JSP22)

The subroutine sets up the twelve word record, as described above, in YA1 to YA12 for transferring over to the main program.

Exits 1. The normal exit is via EXIT 1 with the record set up as described.

2. EXIT 2 is a special exit when the end of the data has been detected. The last item whether word or punctuation, has been dealt with before this and

YA1 - 12 is empty.

This subroutine uses P20, P21, P23, P24, P25, P55.

Computer requirements

Nesting store	:	7 cells
SJNS	:	4 cells
Q-stores	:	Q1 - 5 (preserved & reset)
YA stores	:	YA 1 - 12
YB stores	:	YB 1 - 13
YC stores	:	YC 1 - 3
YZ stores	:	YZ 0 - 127

P23

This is a subroutine for reversing a word of less than or equal to 24 letters. The word itself is contained in YA1, 2, and 3, and the word is reversed into YA 4, 5 and 6. It is required that the modifier part of Q1 (i.e. M1) should contain the address of that register which contains the last letter of the word.

Entries 1. Only entry is at reference 0 (JSP23).

The subroutine takes the word contained in registers YA1 to YA3 and sets the reverse copy of the word into YA4 to YA6, both versions being left justified.

For example, if the word NEVERTHELESS is to be reversed, the following would be the result:-

Before	YA1	YA2	[illegible]	YA3
	NEVERTHELESS			
After	YA1	YA2	[illegible]	YA3
	NEVERTHELESS			
	YA4	YA5	[illegible]	YA6
	SSELHTREVEN			

Exits

Normal exit is via EXIT 1 with the nesting store empty.

Computer requirements

Nesting store	:	3 cells
SJNS	:	0 cells
Q-stores	:	Q1, 2, 3, 4 (preserved & reset)
YA stores	:	YA 1 - 6
YB stores	:	YB 1 - 3

P24

This subroutine reverses a word of greater than 24 letters. The word itself will be contained in YA1 onwards. Only the first 24 letters will be preserved in YA1, 2 and 3 and only the last 24 letters will be preserved in YA4, 5 and 6. As for P23, M1 must

contain the address of the register containing the last part of the word.

Entries 1. The only entry is at reference 0 (JSP24).

The whole word is copied into YB1 - 6 and the address of that YB-store which contains the last part of the word is set in M1. The subroutine then enters P23 and the last 24 letters are set into YA4, 5 and 6. Thus the following would be the result:-

Before YA1 YA2 YA3 YA4

ANTIDISESTABLISHMENTARIANISM

After YA1 YA2 YA3

ANTIDISESTABLISHMENTARIA

YA4 YA5 YA6

MSINAIRATNEMHSILBATSESID

Computer requirements

Nesting store	:	3 cells
SJNS	:	1 cell
Q-stores	:	Q13, 14, 15 (preserved & reset)
YA stores	:	YA 1 - 6
YB stores	:	YB 1 - 6

P23 is also required by this subroutine.

P25

This subroutine reads a reference. The current reference is held in YC1 and YC2 and is in four parts which are described earlier. When a reference warning character is detected, the contents of YC1 and the first half of YC2 are altered according to what is in the reference which follows the reference warning character.

Entries 1. The only entry is at reference 0 (JSP25) At this point, a reference warning character has been detected and a reference is expected. Only digits are acceptable as reference characters and if any other character is detected, a soft report is written and the character ignored. The digits are read in and stored and the X3 16 bit is then removed and the corresponding decimal number formed. This is continued until a separator or terminator is detected at which point the numbers are converted to binary and stored in the appropriate parts of YC1 and YC2.

Exits The only exit is via EXIT 1 with YC1 and YC2 containing the references. An example is as follows:-

input + R1. R2. R3 CRLF

gives:- YC1 = R1, R2
YC2 = R3, 0

P25

Computer requirements

Nesting store	:	7 cells
SJNS	:	3 cells
Q-stores	:	Q15 (preserved & reset)
YC stores	:	YC1 and 2

This subroutine also uses P21 and P55.

P30 Open write magnetic tape file.

Entries 1. The only entry is at reference 0 (JSP30)

N1 = 8 character magnetic tape identifier.

N2 = message length in words.

This subroutine opens a write file on the magnetic tape defined by the eight character identifier in N1. As this routine uses L37, the label on the magnetic tape must be written in the standard 16 word form defined for the KDF9 Magnetic Tape Handling Suite of Subroutines. (See SRLM Vol.3 section 16. 3).

The routine sets up the tape data table required by L37 using the contents of N1 and N2. Entry is then made to L37 which opens the write file.

Exits 1. Normal exit is via EXIT 2. This exit means that the write file has been successfully opened.

2. EXIT 1 is a special exit which occurs in the case of a failure in opening the file. The message L37 ERROR →

d →

will be typed on the monitor typewriter, where d is a failure number, the meaning of which can be found in SRLM Vol. 3. Sec. 16.4.

Subroutine uses: L37, L61, L78, L79, L89.

Computer requirements

Nesting store	:	9 cells
SJNS	:	9 cells
Q-stores	:	Q-12 - 15 (not preserved on entry)
W- stores	:	W0 - 30

Note: If this subroutine is being used, the W-stores must be declared at the start of the program. If Q12-15 are being used before entry to this subroutine and the contents are required later, they must be preserved before entry to P30. (This also applies to P31 and P32.)

P31 Write record.

This subroutine writes a record onto a magnetic tape file which has been opened using P30.

Entries. 1. The only entry is at reference 0. (JSP13).

N1 = base address of source area.

The routine writes the record (as defined by the tape data table of P30) onto the magnetic tape defined in P30. The routine again uses L37 and N1 must contain the base address of the T.D.T. and N2 the base address of the source area before entry to L37.

Exits 1. Normal exit is via EXIT 2

2. EXIT 1 is a special exit similar to EXIT 1 for P30.

This subroutine uses P30, L37, L61, L78, L79, L89.

Computer requirements:- see P30

Note: see P30.

P32 Close write file

This subroutine closes a write file opened using P30.

Entries 1. The only entry is at reference 0 (JSP32)

The routine closes a write file as defined by P30, again making use of L37.

Exits: as for P31.

Computer requirements: see P30.

P40 Open read magnetic tape file.

Entries 1. The only entry is at reference 0 (JSP40).

N1 = 8 character magnetic tape identifier.

N2 = message length in words.

The subroutine opens a read file on the magnetic tape defined by the 8 character identifier in N1. As this routine uses L34, the tape label must be set up in the standard 16 word form defined by the KDF9 Magnetic Tape Handling Suite of Subroutines. (see SRLM Vol.3 Section 16.3).

The routine sets up the Tape Data Table required by L34 using the contents of N1 and N2. Entry is then made to L34 by which the read file is opened and the first record read.

Exits 1. Normal exit is via EXIT 2.

N1 contains the base address of the next message.

2. EXIT 1 is a special failure exit resulting from a failure in L34.

The message L34 ERROR →

d →

will be typed on the monitor typewriter, where d is a fail number, the meaning of which can be found in SRLM Vol.3. sec. 16.5.

P40

Subroutine uses : L34, L78, L79, L89.

Computer requirements

Nesting store	:	7 cells
SJNS	:	6 cells
Q-stores	:	Q11 - 15 (not preserved on entry)
W-stores	:	W 0 - 30

Note: See notes for P30. (Also applies to P41 & P42)

P41 Read record.

This subroutine reads a record from a magnetic tape file which has been opened by P40.

Entries 1. The only entry is at reference 0 (JSP41).
N1 = base address of last record read.
(i.e. result of previous entry to P41 or of entry to P40.)

The routine reads the next record, as defined by the T.D.T. of P40, into core and sets the address of this record in N1.

Exits 1. EXIT 2 is normal exit
N1 = base address of next record.
2. EXIT 1 is a fail exit as defined by P40.

Computer requirements. See P40.

P42 Close read file.

Entries 1. The only entry is at reference 0 (JSP42)

The subroutine closes a read file as defined by T.D.T. of P40. Once the file is closed, the tape is rewound to B.T.C.

Exits Exit is via EXIT 1, although a failure may have occurred in L34, in which case a message as for P40 is typed on the monitor typewriter.

Computer requirements see P40.

P50

This subroutine stores the contents of an area of Y-stores from Y1 onwards onto a magnetic tape.

Entries 1. The only entry is at reference 0 (JSP50).

N1 = upper address of source area.

The routine writes onto a magnetic tape file, which has been opened by P30 and which is defined by the T.D.T. of P30, the area of store between Y1 and the address specified in N1. It is assumed that the record length in this case is 13 words, but this could be easily altered for records of different lengths.

Exits 1. EXIT 2 is the normal exit

2. EXIT 1 is a failure exit as a result of a failure in L37 (see P31).

P50

This subroutine uses P30, P31, L37, L61, L78, L79, L89.

Computer requirements

Nesting store	:	10 cells
SJNS	:	10 cells
Q-stores	:	Q10 (preserved & reset)
W-stores	:	W 0 - 30.

This routine preserves the contents of Q11 - 15 before entry to P31.

P51 Read magnetic tape identifier.

This subroutine reads an 8 character identifier from paper tape. It can only be used when the paper tape reader is not already being used as it claims the reader and fills the input buffer using P21.

Entries 1 The only entry is at reference 0. (JSP51)

The routine reads the first eight printable characters (i.e. all those on paper tape code except O.S., C.N., CRLF, and SPACE).

Exits 1. The only exit is via EXIT 1.

with N1 = identifier.

The subroutine uses P21, P55.

P51

Computer requirements

Nesting store	:	3 cells
SJNS	:	3 cells
Q-stores	:	Q15 (preserved & reset).

P55

This subroutine reads from paper tape into a buffer. Double buffering is used to enable processing to continue while input is in progress.

Entries 1 Initial entry: entry is at reference 0 (JSP55).

The subroutine claims the paper tape reader and fills the buffers initially.

2. Fill next buffer: entry is at reference 1 (JSIP55).

The subroutine tests a marker to see which buffer is to be filled next and then reads into this area while processing of the other buffer is continuing. A check is made as to whether a parity failure has occurred on input. If this is the case, a message:-

PARITY FAIL ON INPUT

is typed on the monitor typewriter and the program terminates.

P55

Exits 1. The only exit is via EXIT 1 with Q15 set up in the format 64/1/6a of source area, ready for the character fetch via P21.

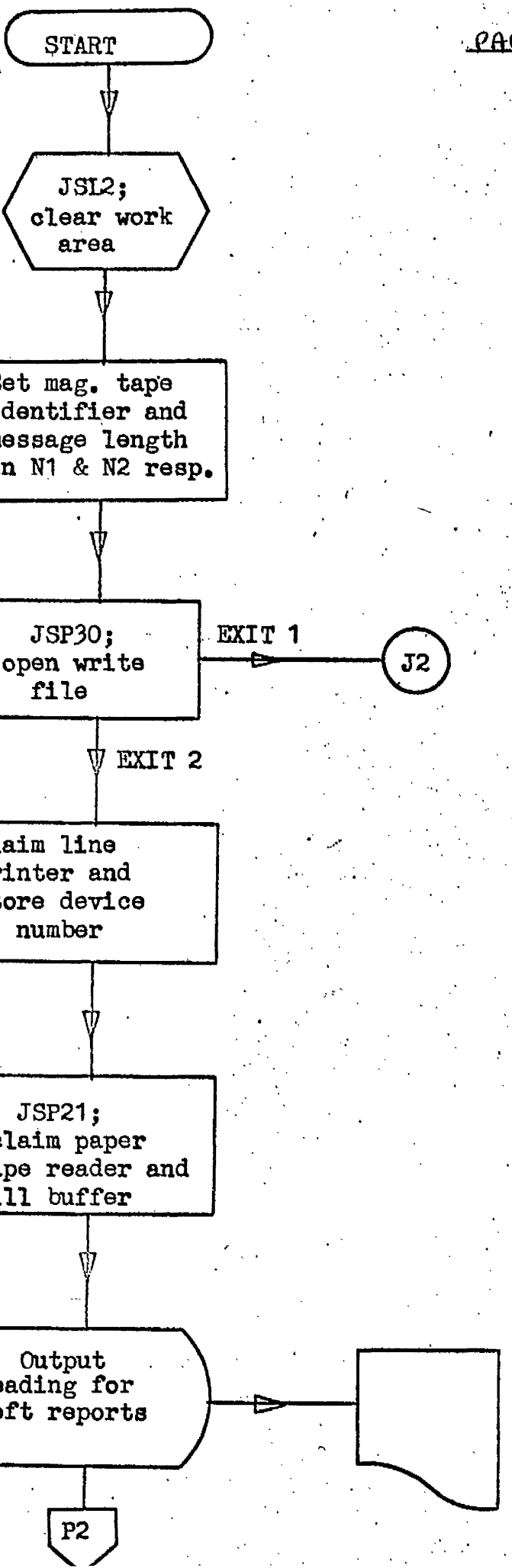
Computer requirements

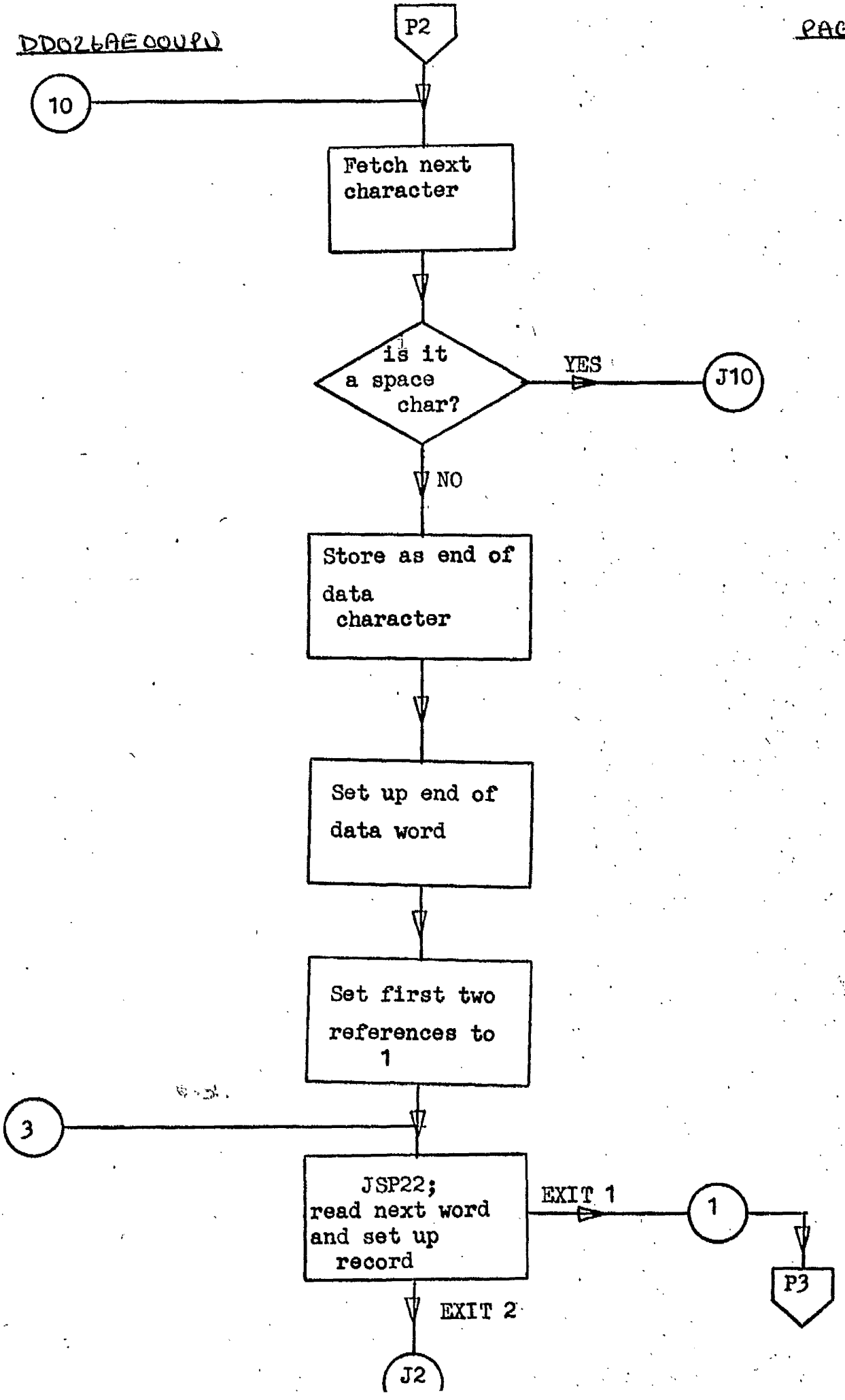
Nesting store	:	3 cells
SJNS	:	1 cell
Q-stores	:	Q15
YZ-stores	:	YZ 0 - 127

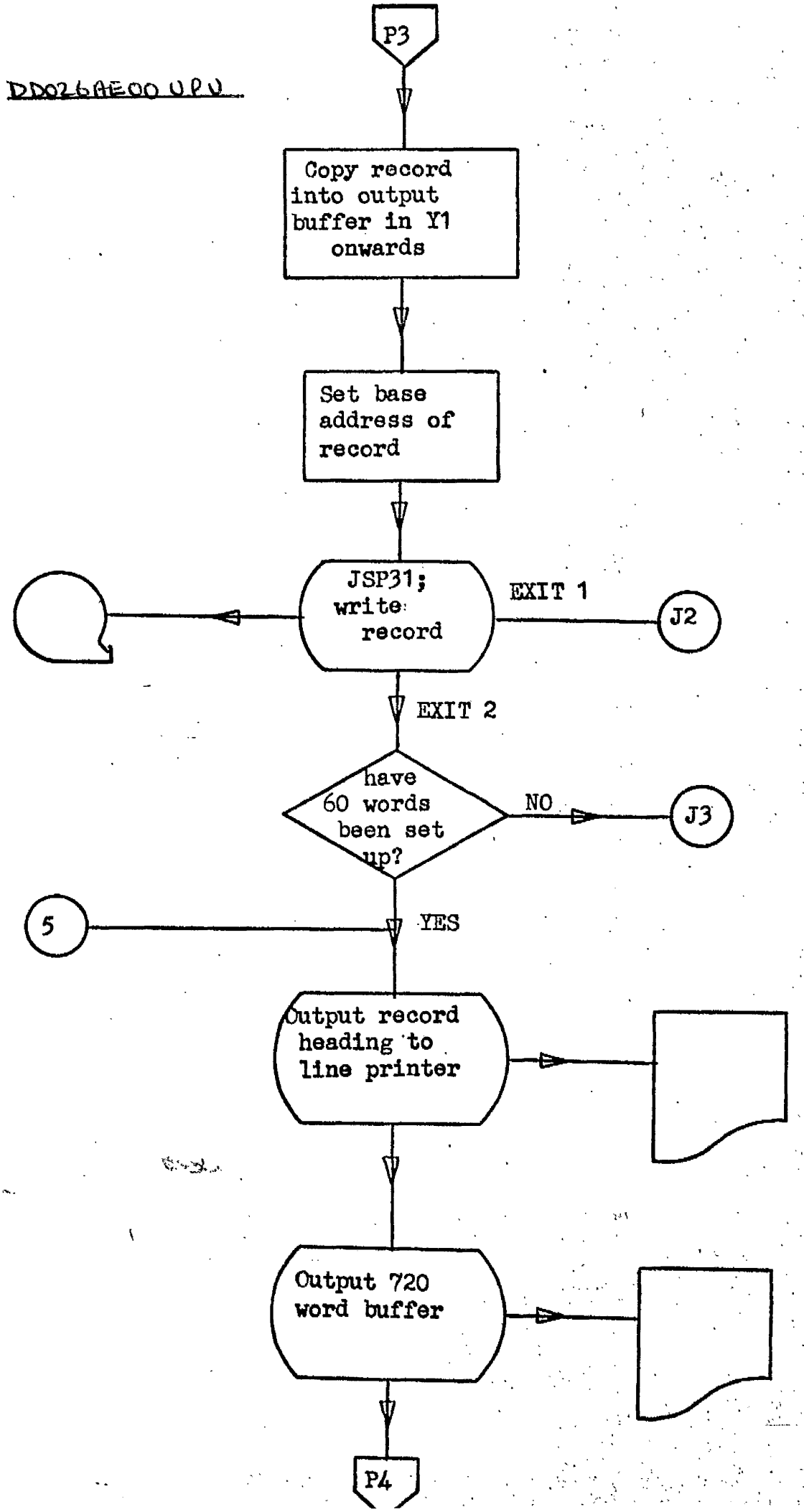
TR used.

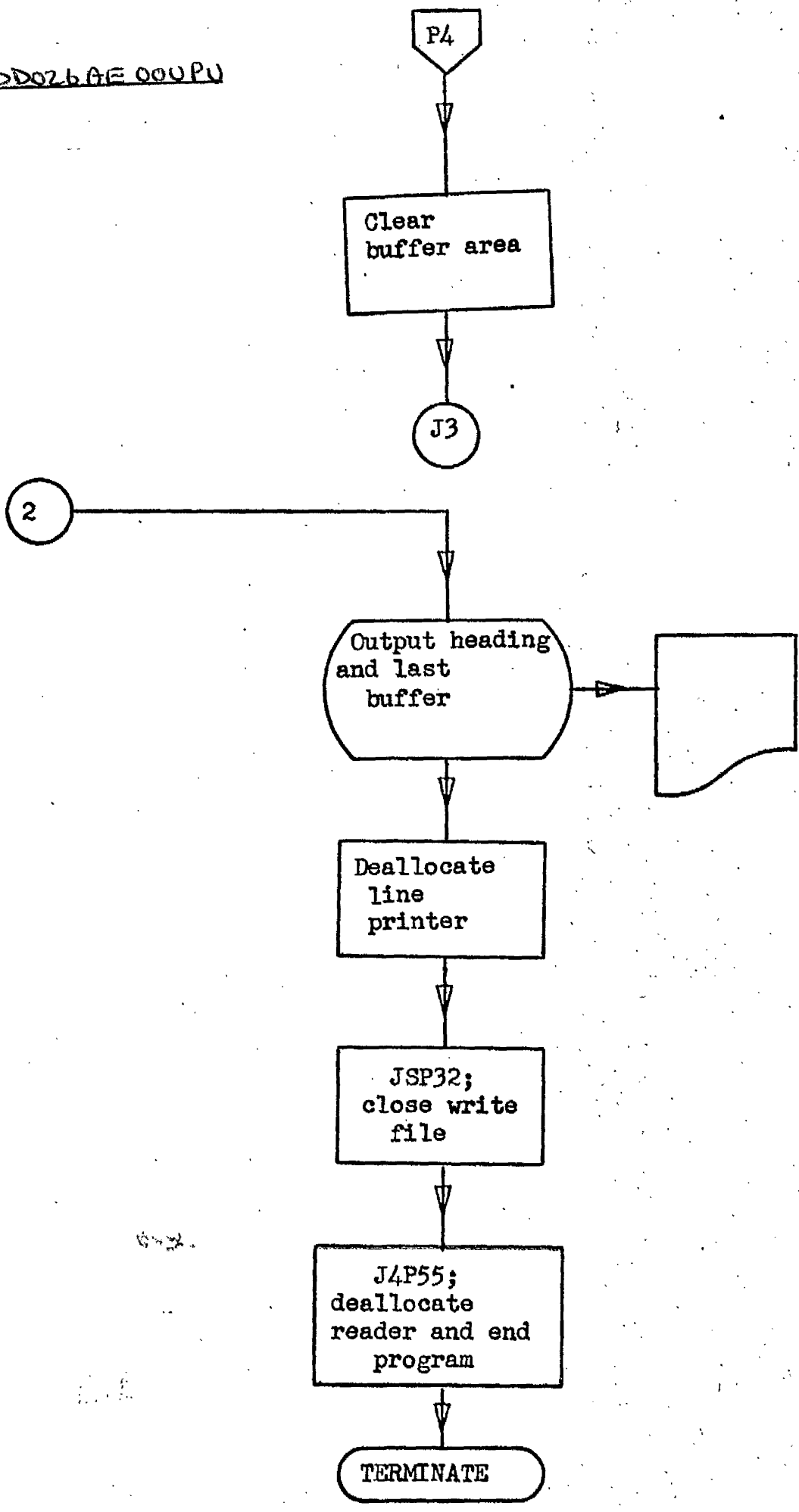
Entry 3 Terminate Program: entry is at reference 4. (JS4P55).

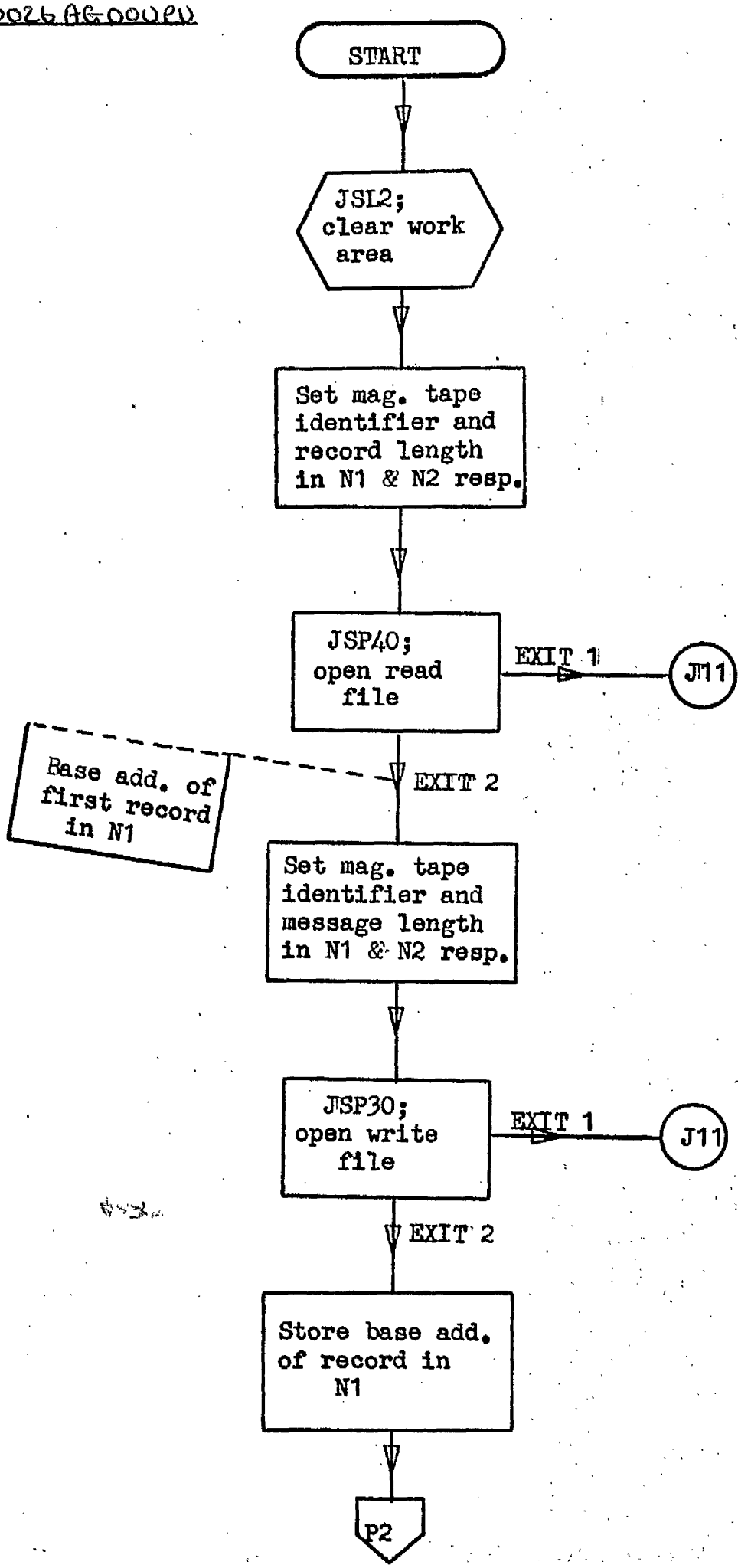
The subroutine deallocates the paper tape reader and terminates the program.

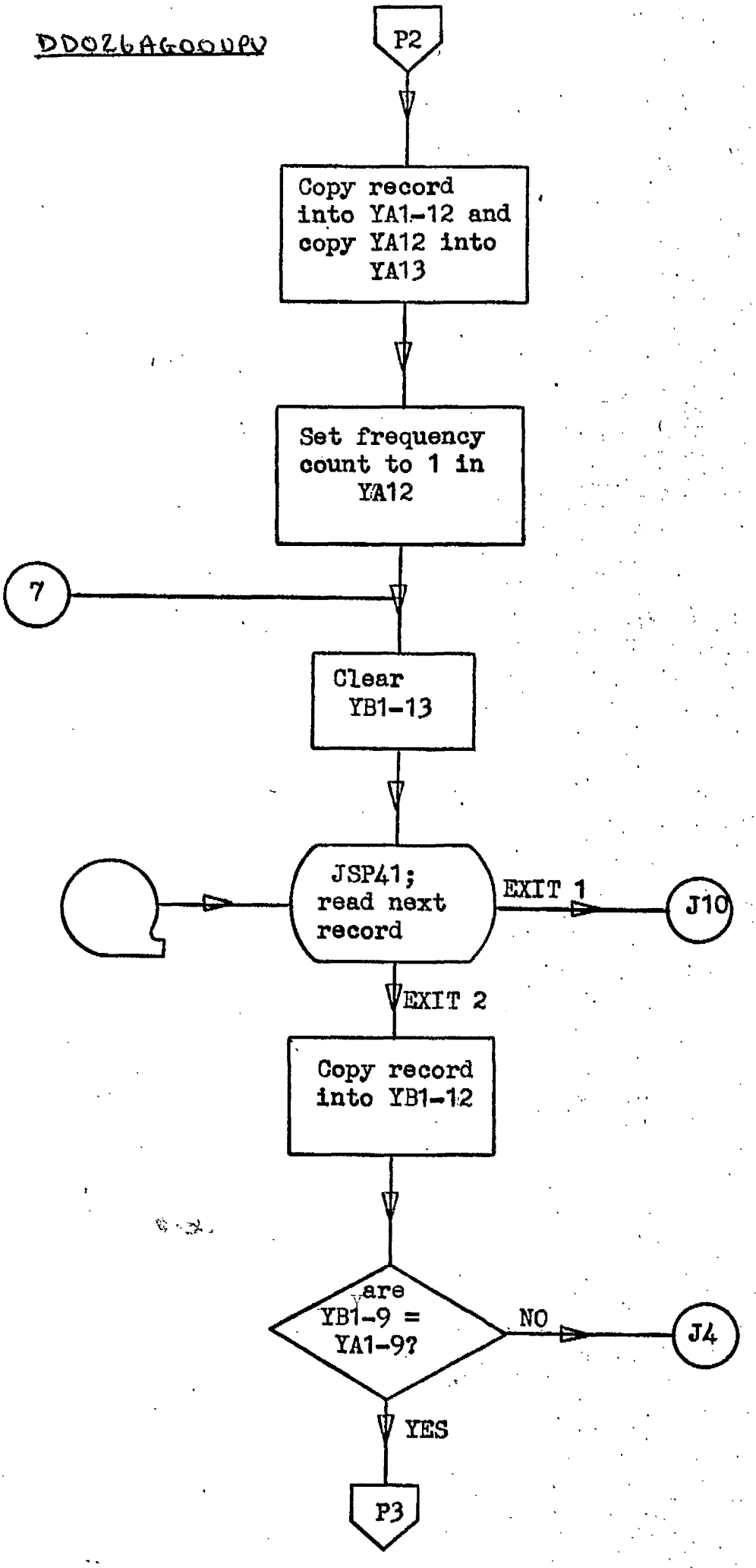

START
JSL2; clear work area
Set mag. tape identifier and message length in N1 & N2 resp.
JSP30; open write file
EXIT 1
J2
EXIT 2
Claim line printer and store device number
JSP21; claim paper tape reader and fill buffer
Output heading for soft reports
P2

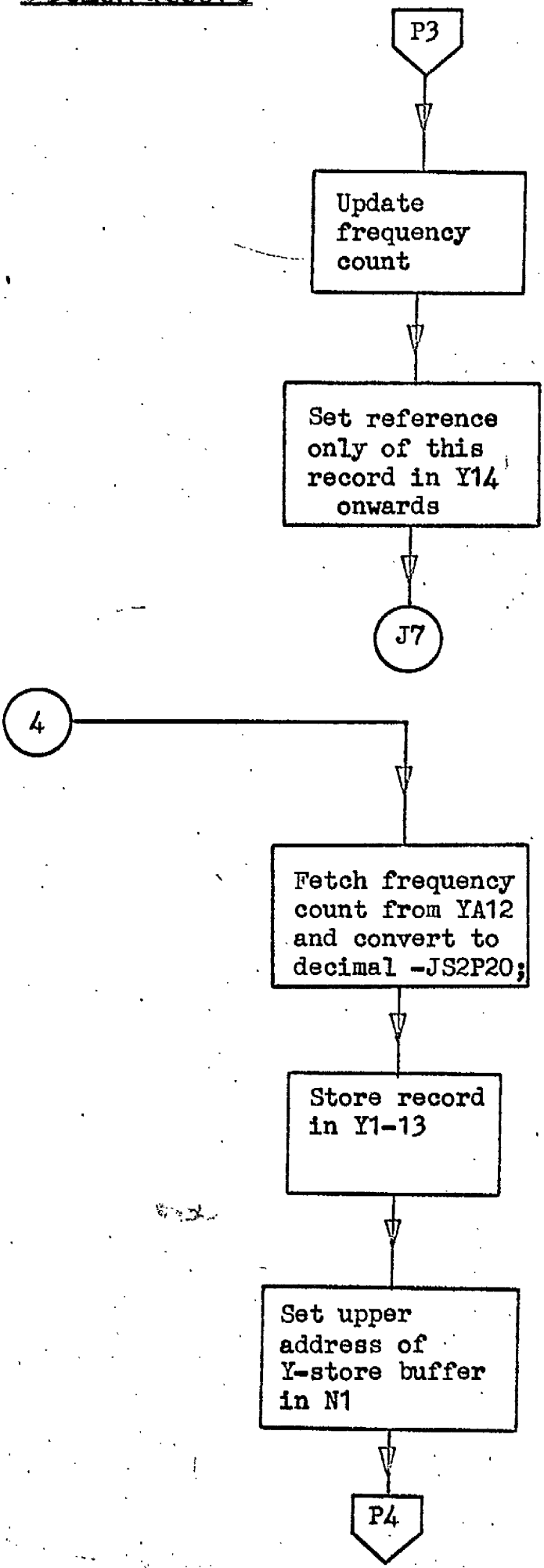

P2
10
Fetch next character
is it a space char?
YES
J10
NO
Store as end of data character
Set up end of data word
Set first two references to 1
3
JSP22; read next word and set up record
EXIT 1
1
P3
EXIT 2
J2

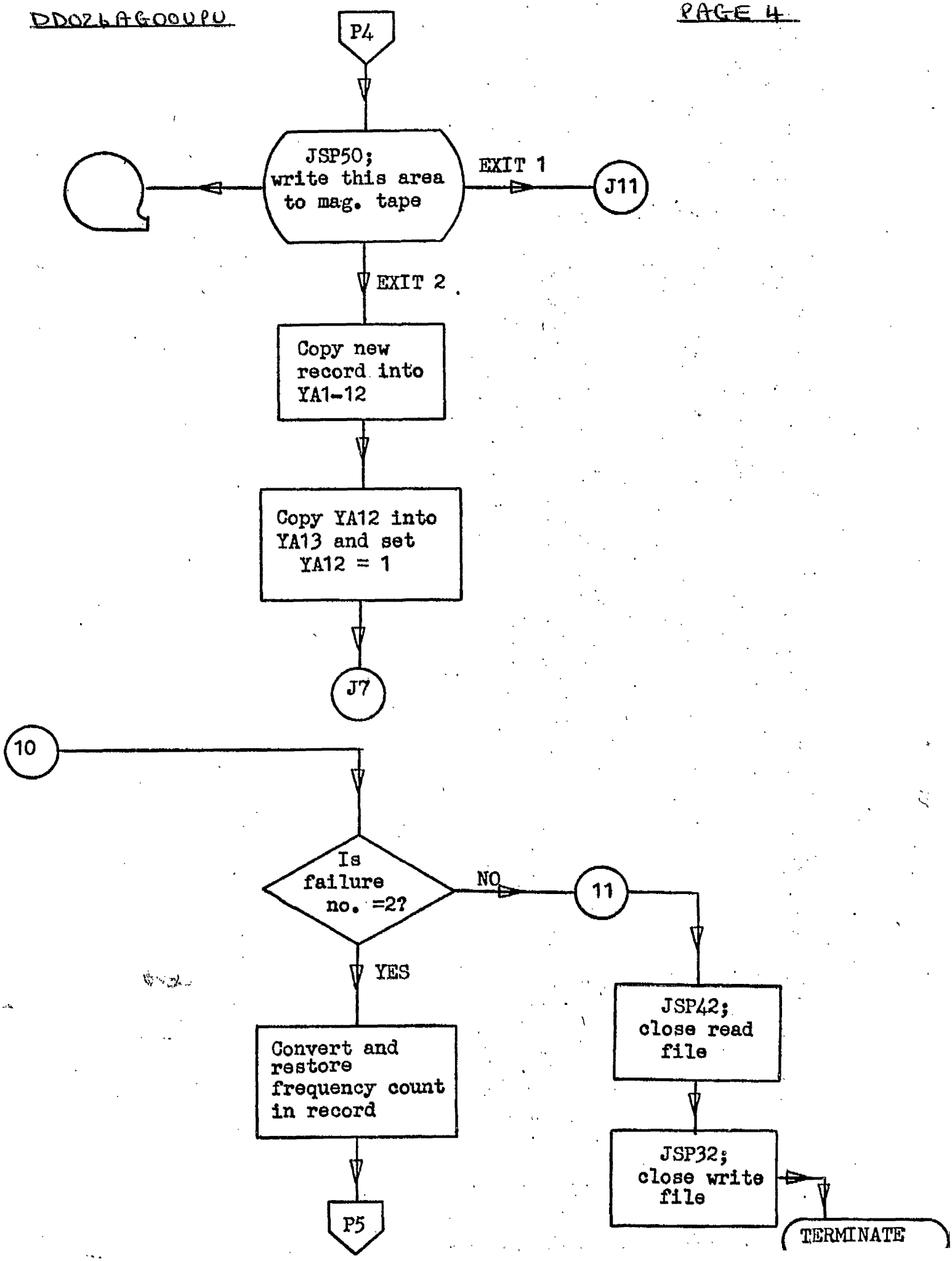

P3
Copy record into output buffer in Y1 onwards
Set base address of record
JSP31; write record
EXIT 1
J2
EXIT 2
have 60 words been set up?
NO
J3
5
YES
Output record heading to line printer
Output 720 word buffer
P4

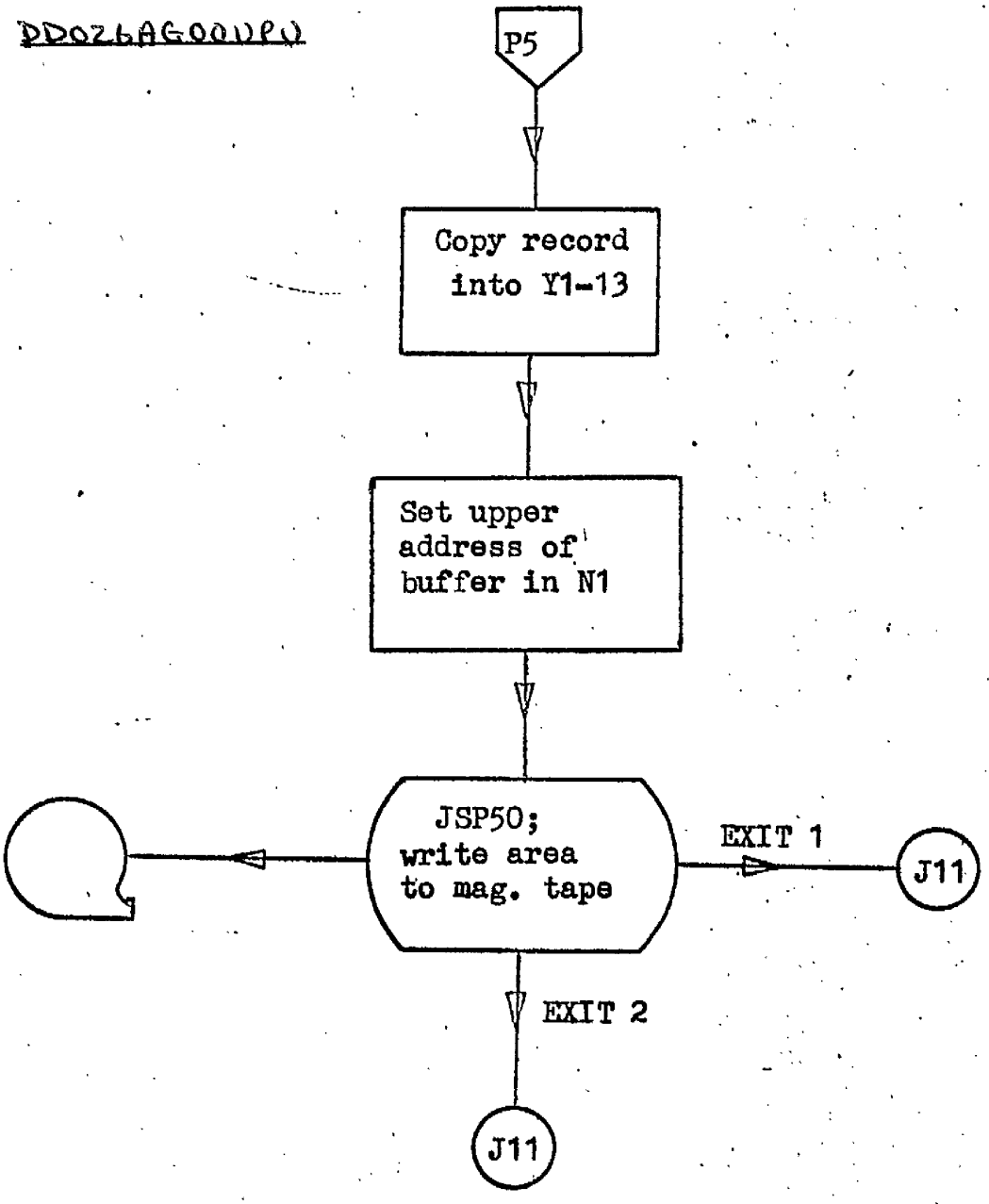

P4
Clear buffer area
J3
2
Output heading and last buffer
Deallocate line printer
JSP32; close write file
J4P55; deallocate reader and end program
TERMINATE

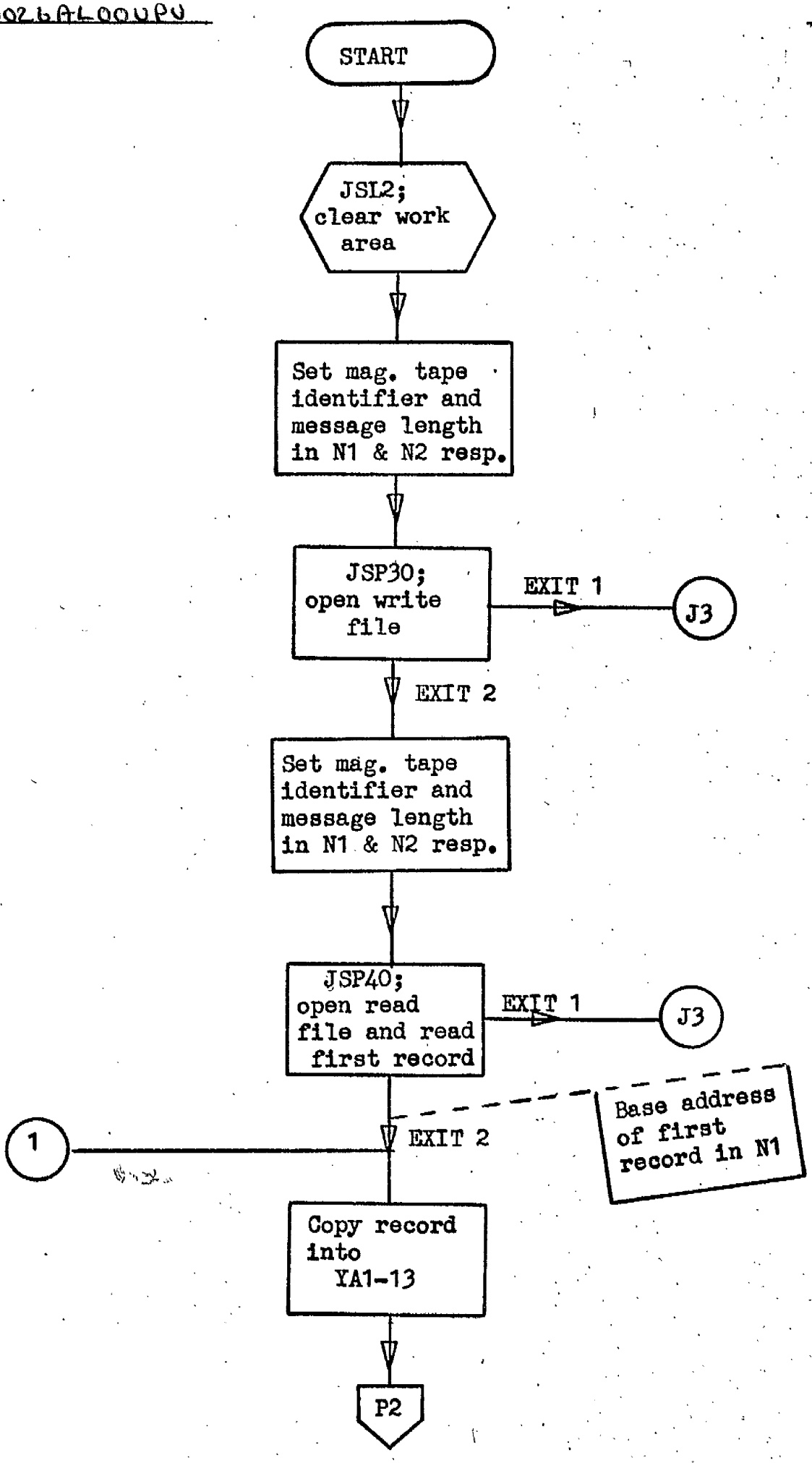

START
JSL2; clear work area
Set mag. tape identifier and record length in N1 & N2 resp.
JSP40; open read file
EXIT 1
J11
Base add. of first record in N1
EXIT 2
Set mag. tape identifier and message length in N1 & N2 resp.
JSP30; open write file
EXIT 1
J11
EXIT 2
Store base add. of record in N1
P2

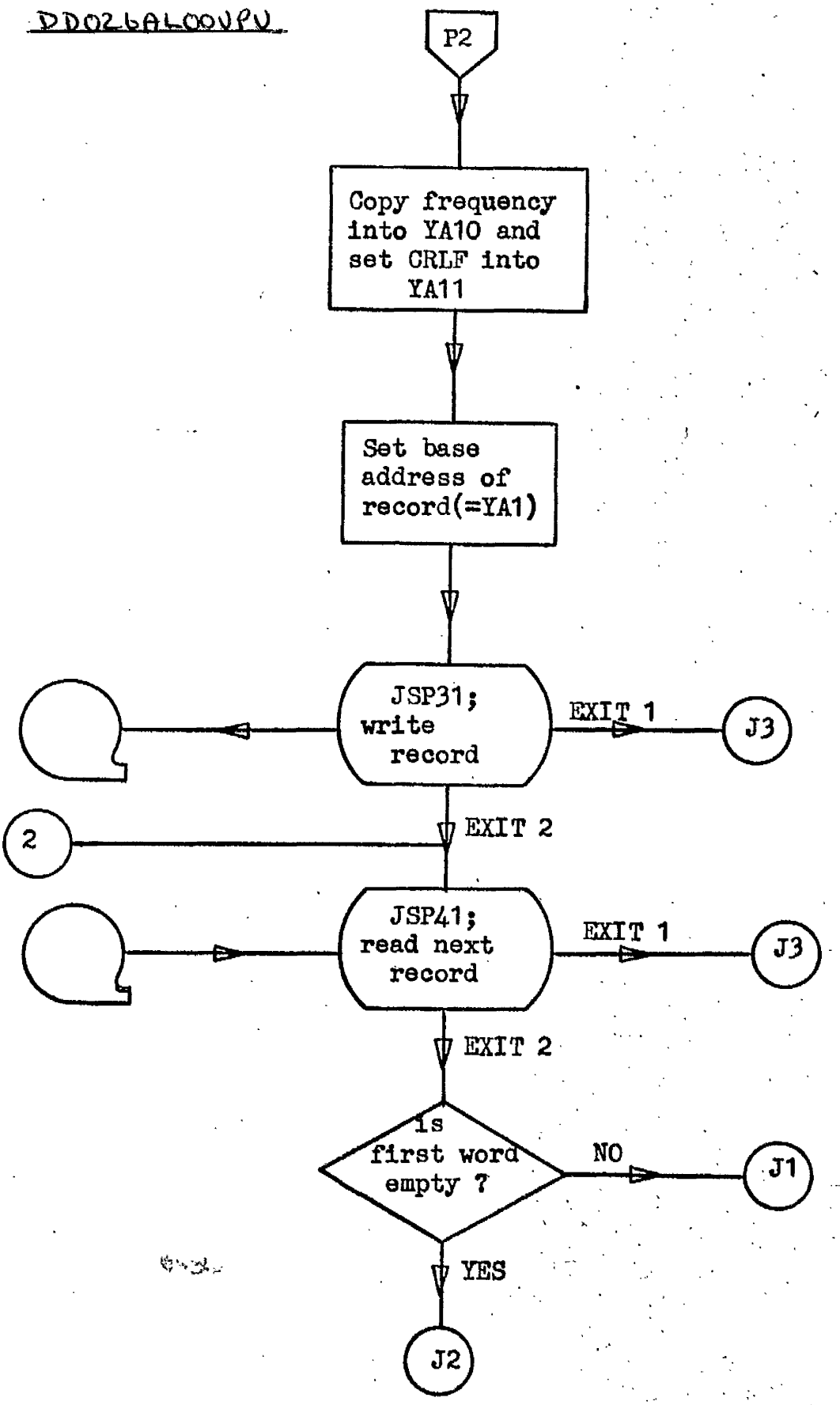

P2
Copy record into YA1-12 and copy YA12 into YA13
Set frequency count to 1 in YA12
7
Clear YB1-13
JSP41; read next record
EXIT 1
J10
EXIT 2
Copy record into YB1-12
are YB1-9 = YA1-9?
NO
J4
YES
P3

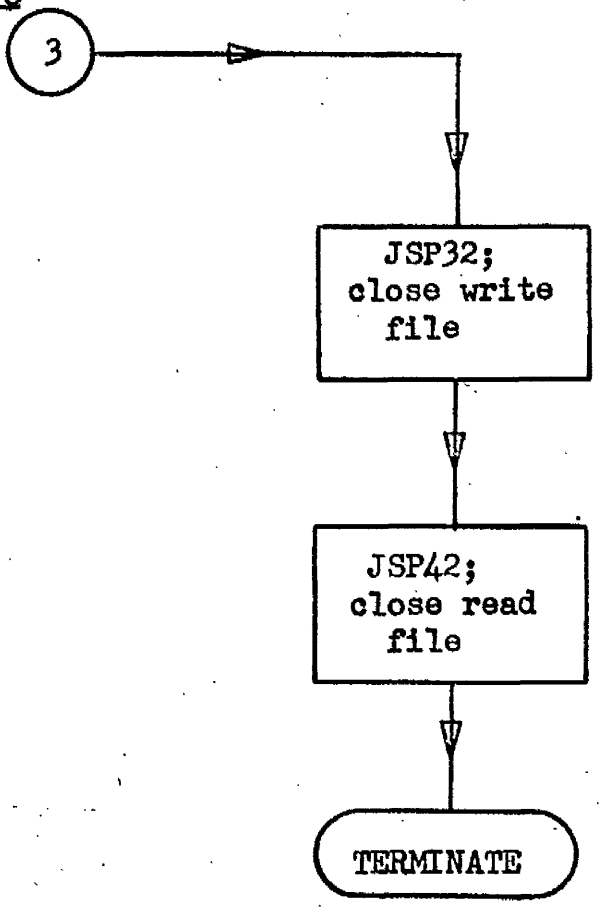

P3
Update frequency count
Set reference only of this record in Y14 onwards
J7
4
Fetch frequency count from YA12 and convert to decimal -JS2P20;
Store record in Y1-13
Set upper address of Y-store buffer in N1
P4


P4
JSP50;
write this area
to mag. tape
EXIT 1
J11
EXIT 2
Copy new
record into
YA1-12
Copy YA12 into
YA13 and set
YA12 = 1
J7
10
Is
failure
no. =2?
NO
11
YES
JSP42;
close read
file
Convert and
restore
frequency count
in record
JSP32;
close write
file
P5
TERMINATE

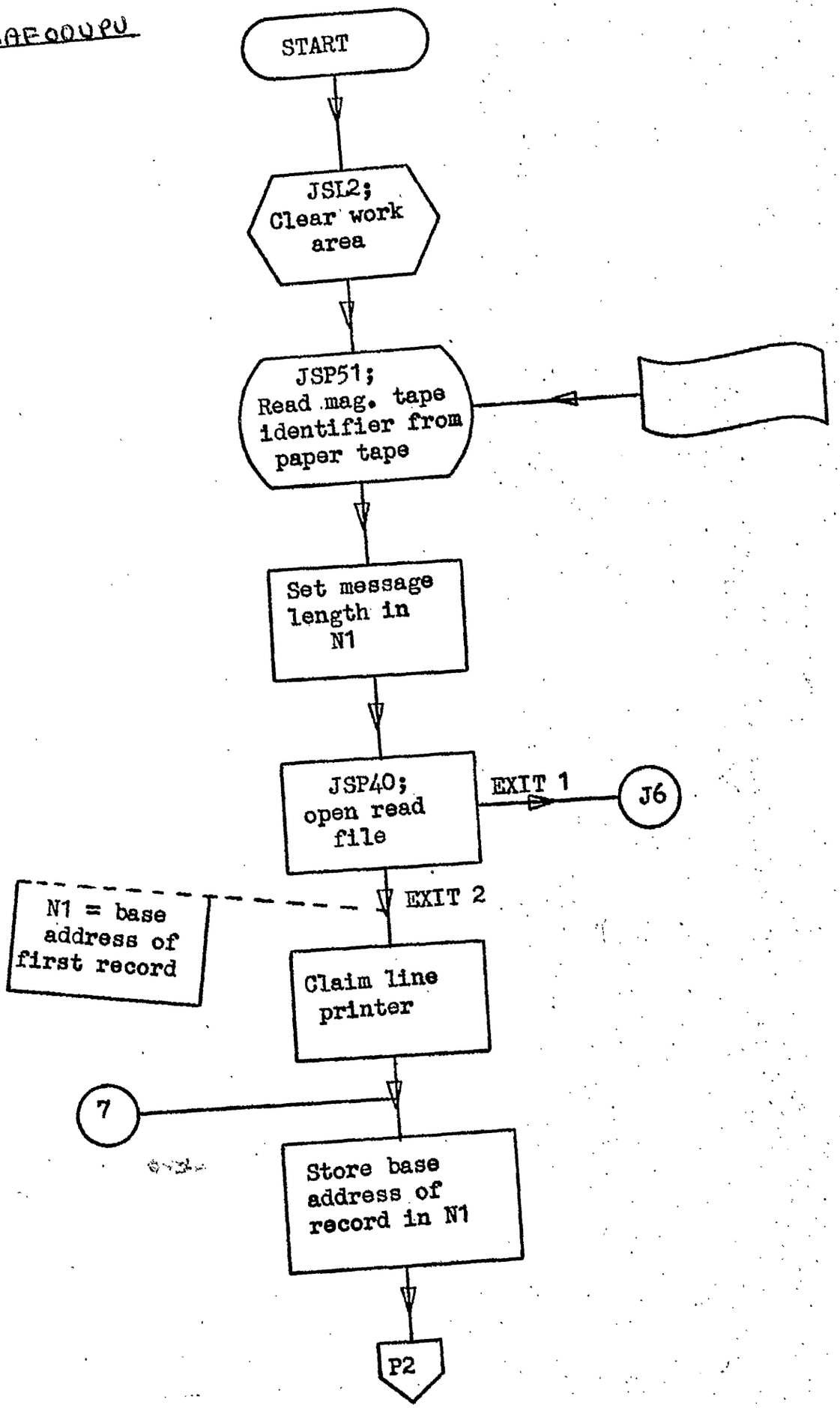

P5
Copy record into Y1-13
Set upper address of buffer in N1
JSP50; write area to mag. tape
EXIT 1
J11
EXIT 2
J11

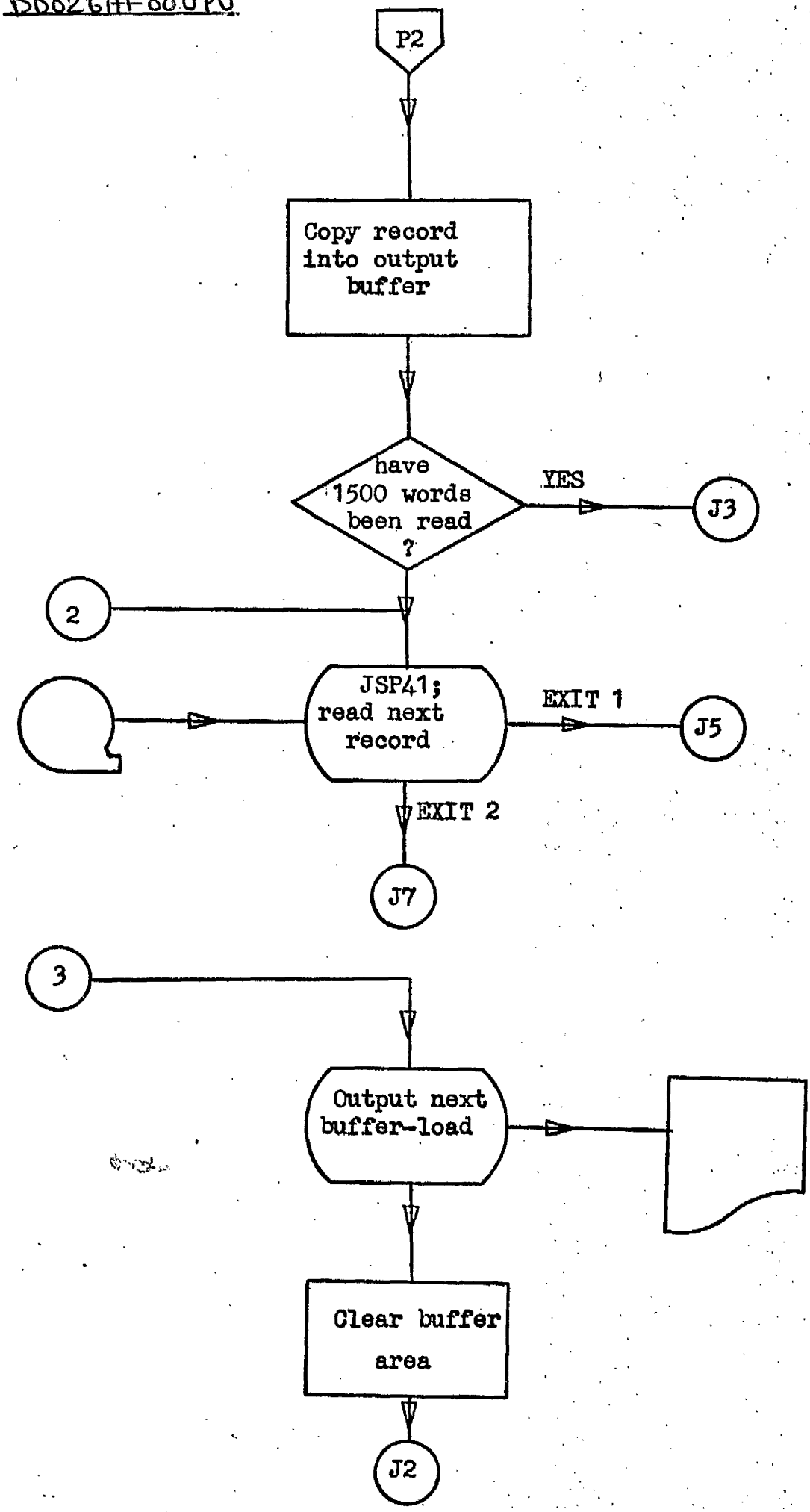

START
JSL2; clear work area
Set mag. tape identifier and message length in N1 & N2 resp.
JSP30; open write file
EXIT 1
J3
EXIT 2
Set mag. tape identifier and message length in N1 & N2 resp.
JSP40; open read file and read first record
EXIT 1
J3
Base address of first record in N1
1
EXIT 2
Copy record into YA1-13
P2

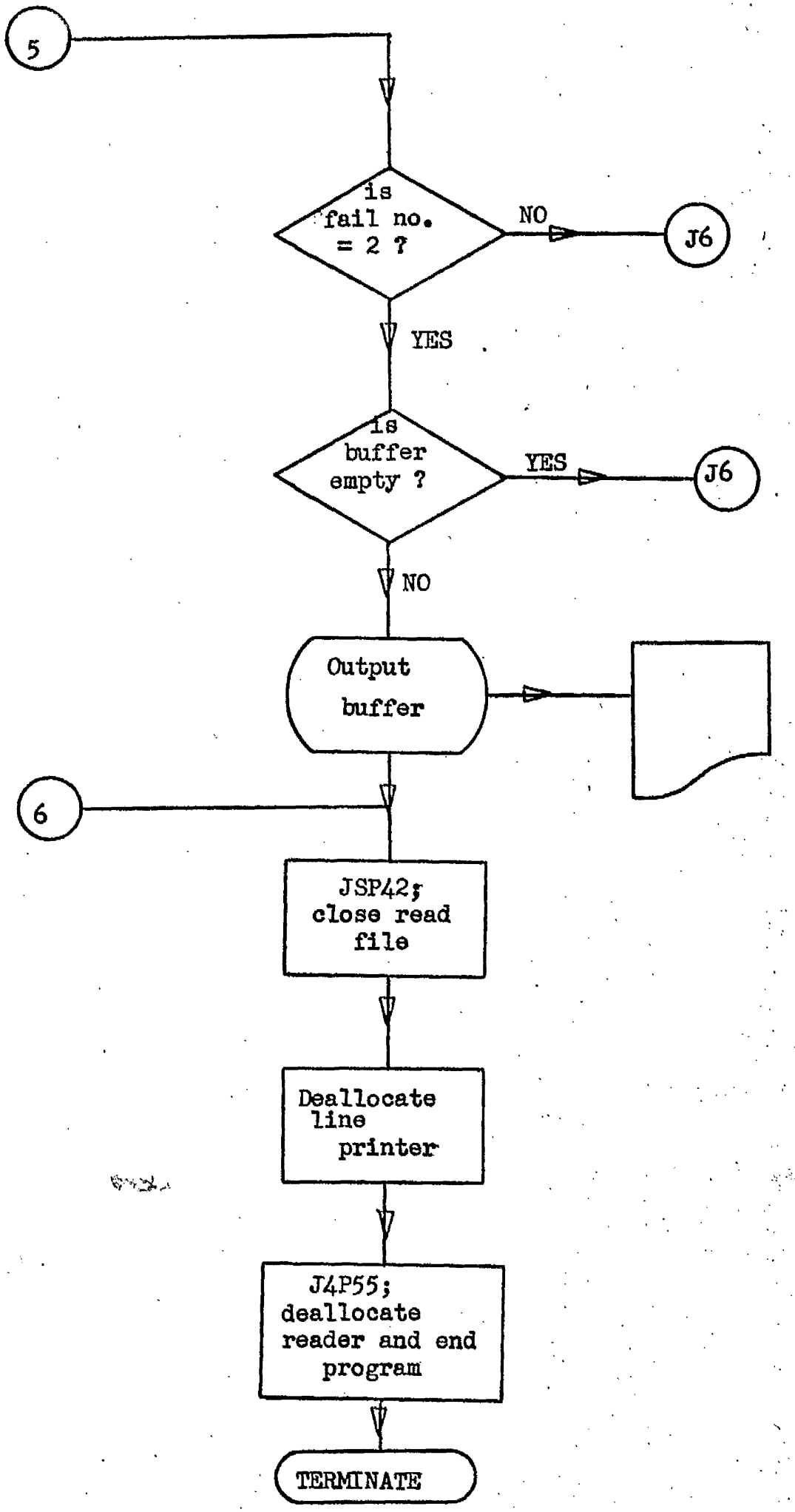

P2
Copy frequency into YA10 and set CRLF into YA11
Set base address of record(=YA1)
JSP31; write record
EXIT 1
J3
2
EXIT 2
JSP41; read next record
EXIT 1
J3
EXIT 2
Is first word empty ?
NO
J1
YES
J2

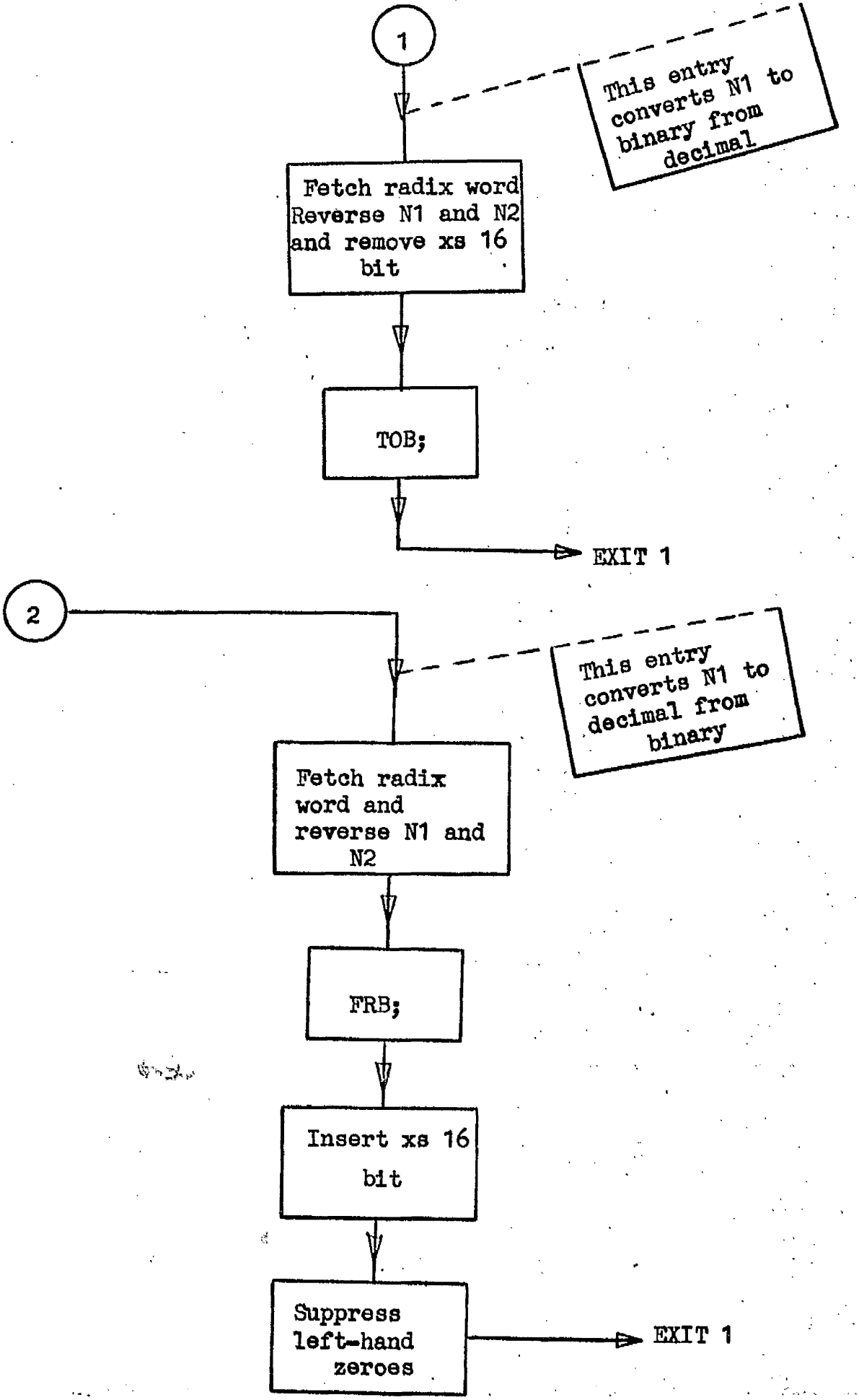

3
JSP32;
close write
file
JSP42;
close read
file
TERMINATE

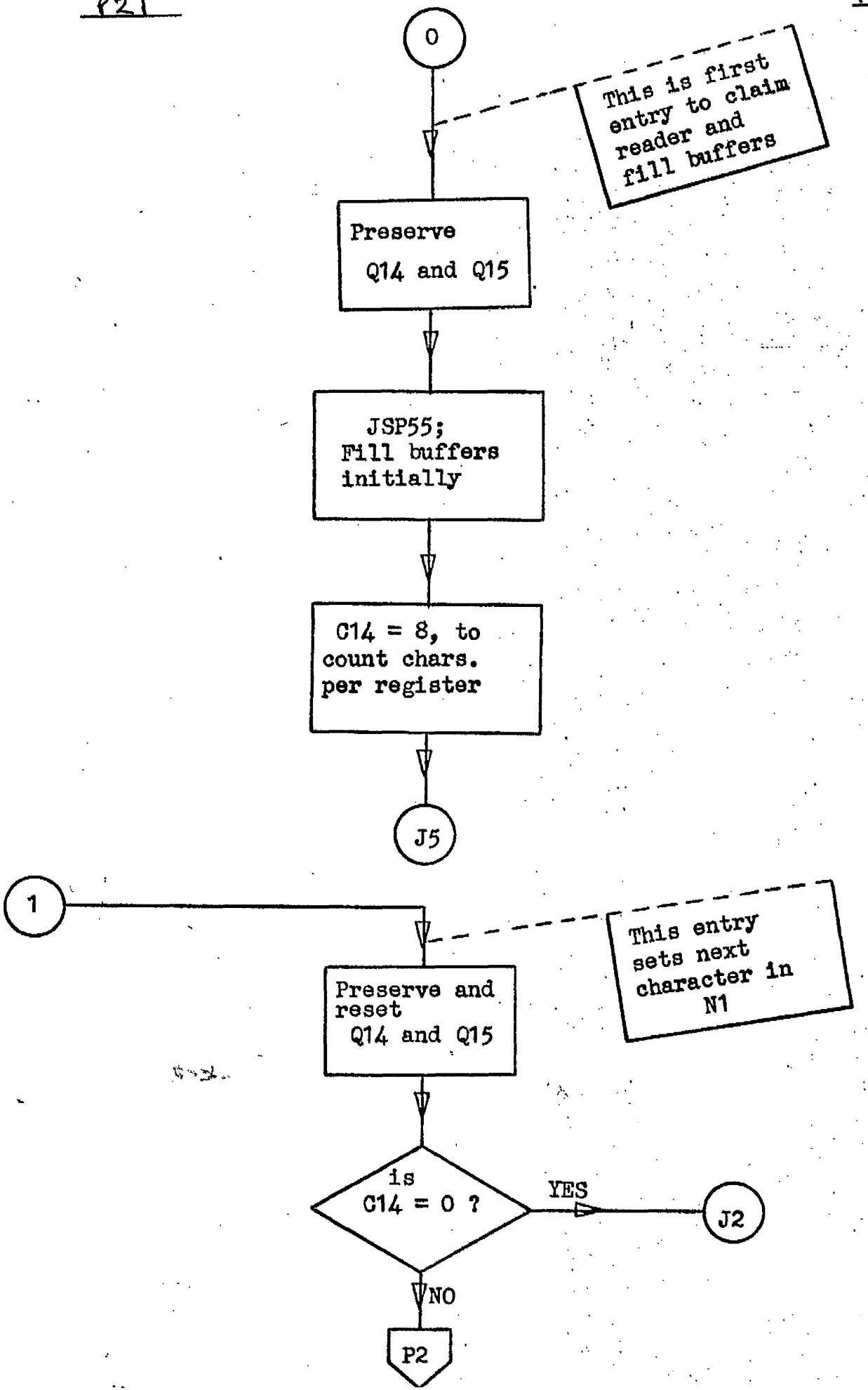
DD026AF00UPU

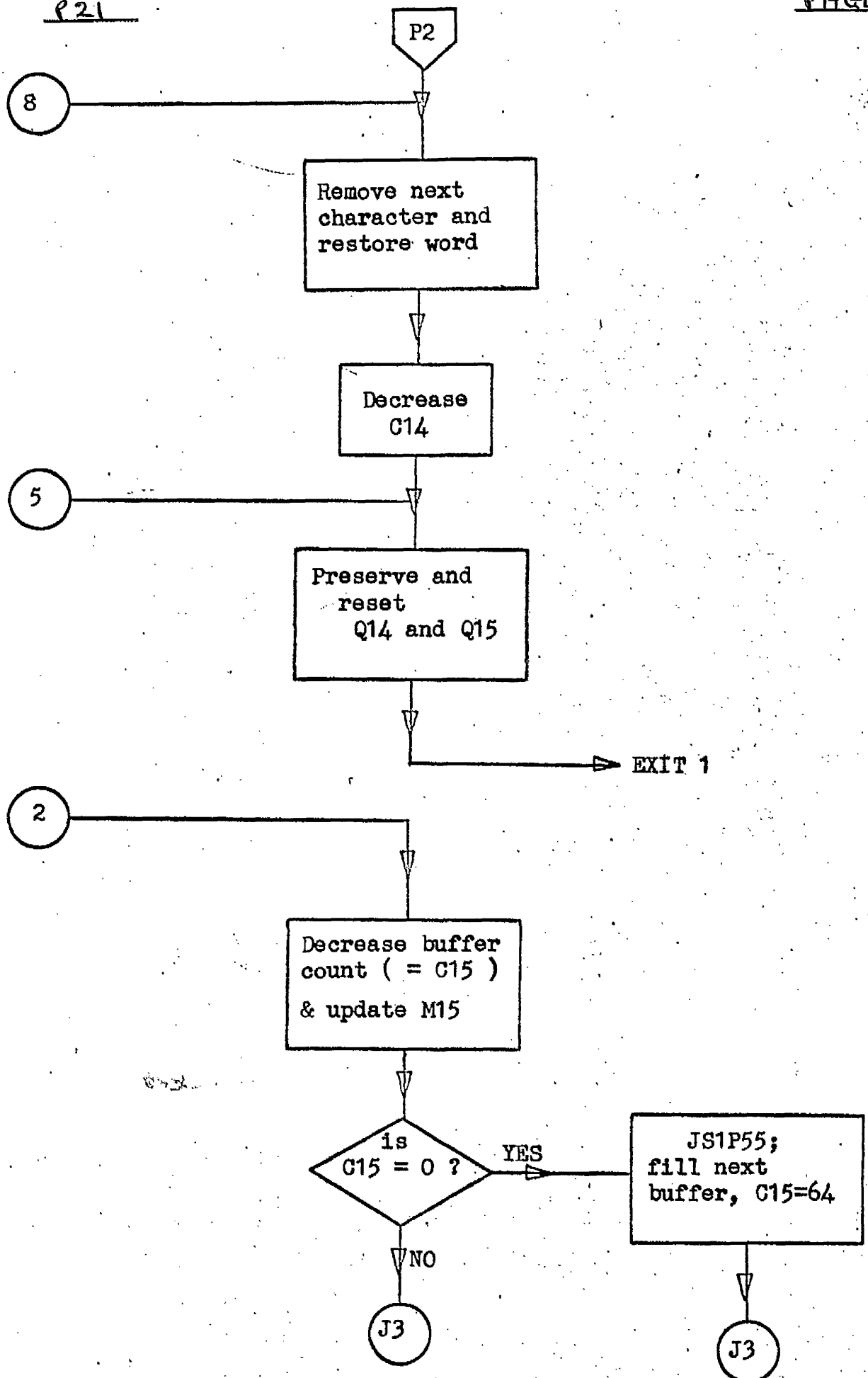


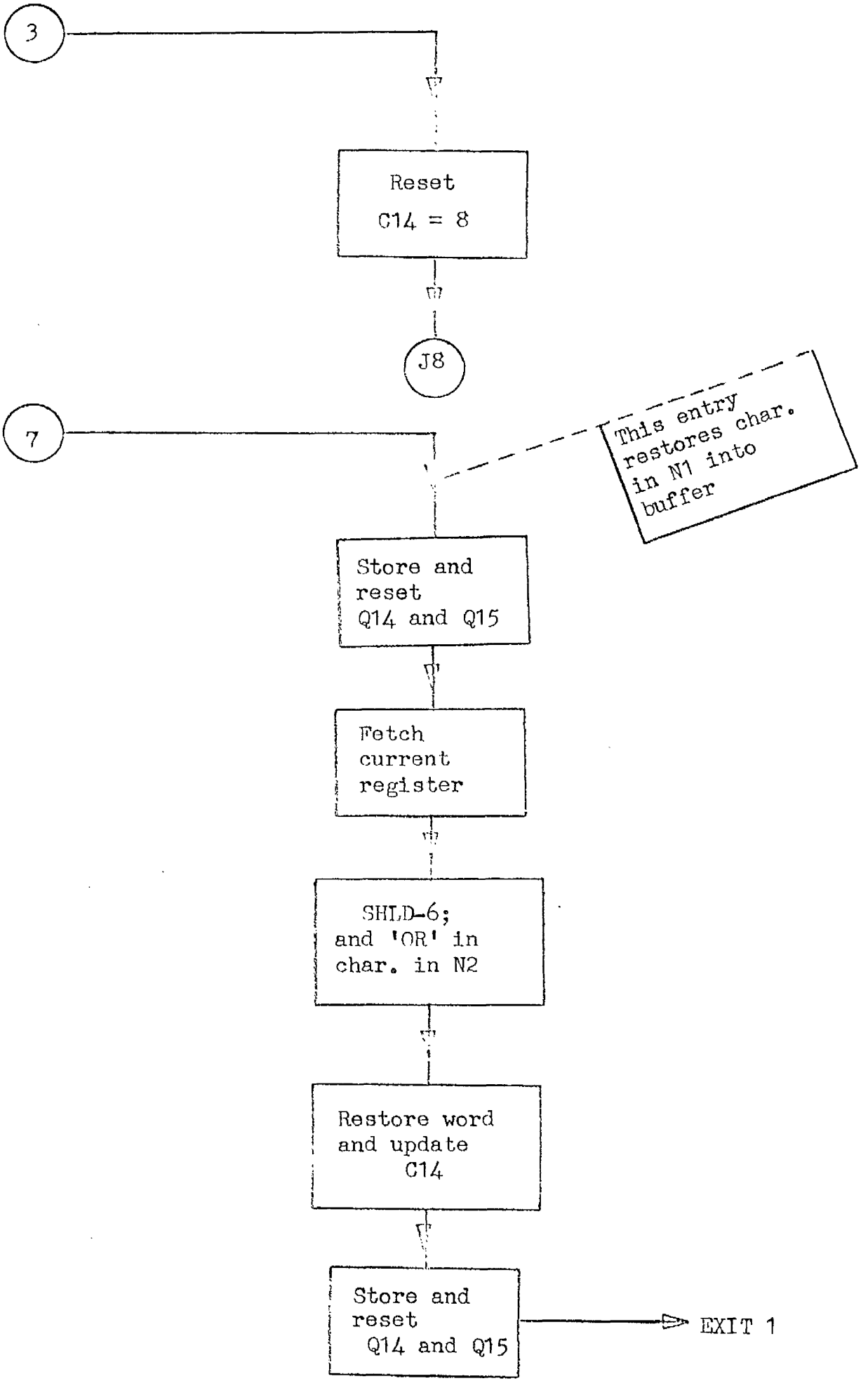

P2
Copy record into output buffer
have 1500 words been read ?
YES
J3
2
JSP41; read next record
EXIT 1
J5
EXIT 2
J7
3
Output next buffer-load
Clear buffer area
J2

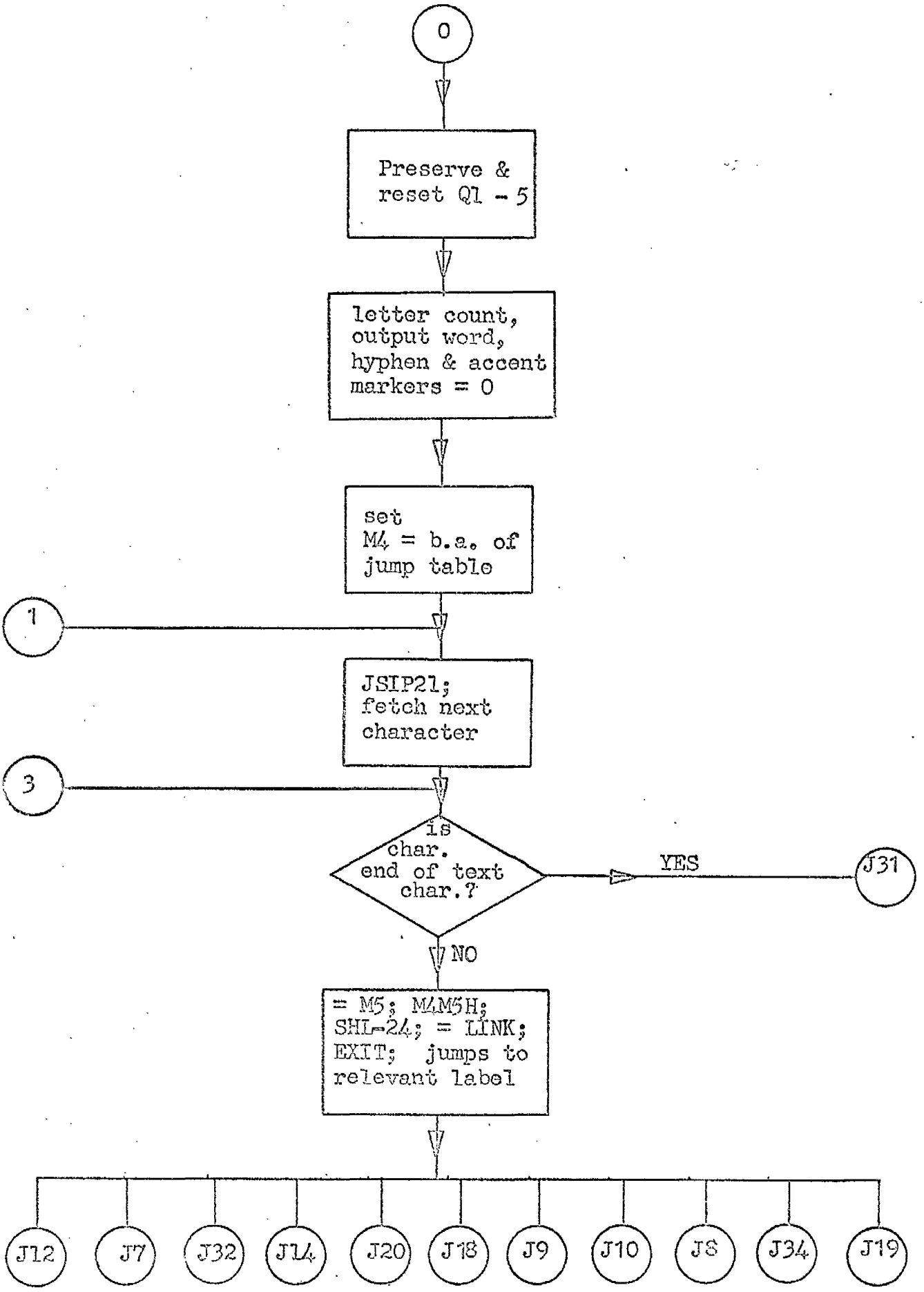

5
is fail no. = 2 ?
NO
J6
YES
is buffer empty ?
YES
J6
NO
Output buffer
6
JSP42; close read file
Deallocate line printer
J4P55; deallocate reader and end program
TERMINATE

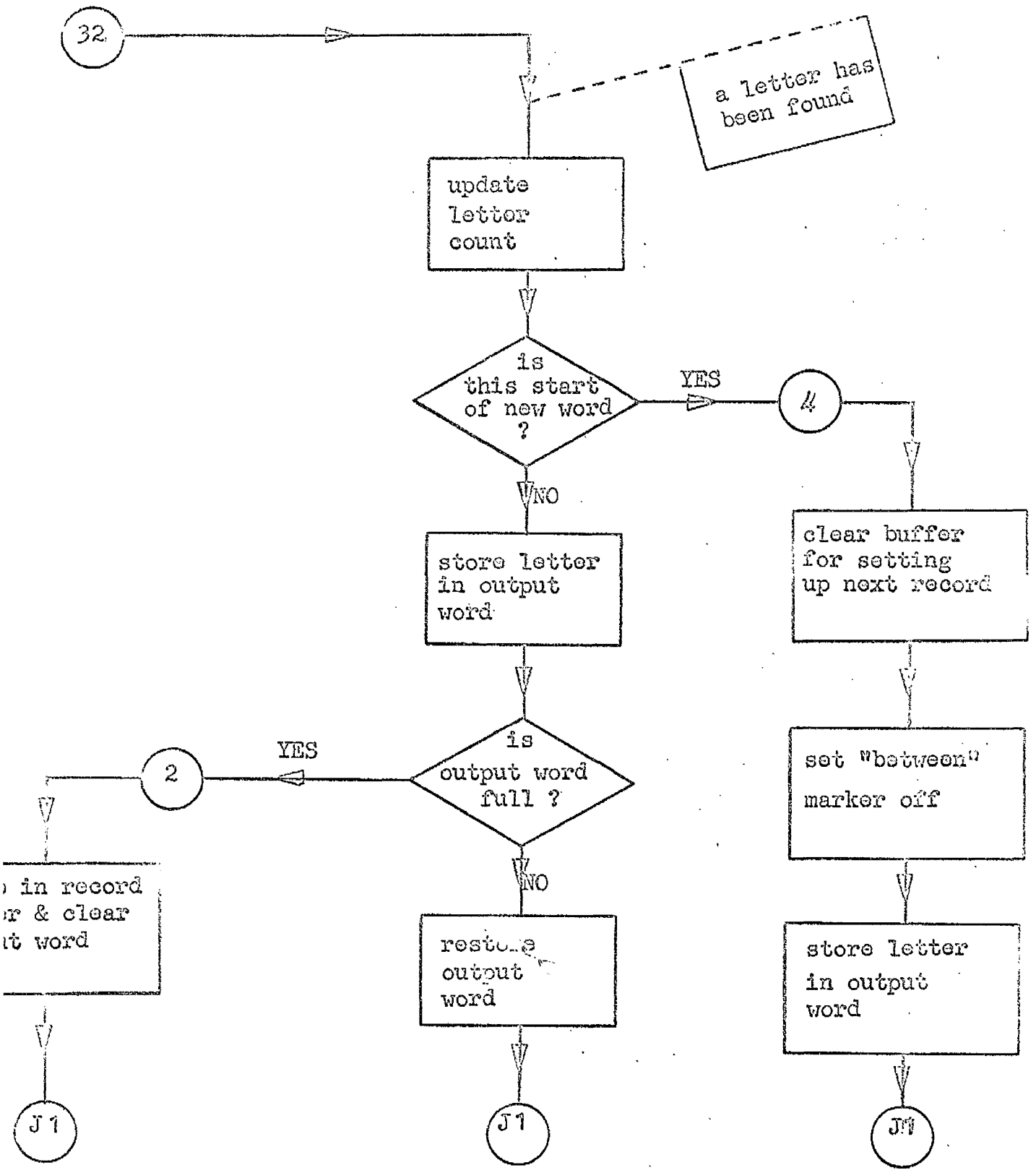

1
This entry converts N1 to binary from decimal
Fetch radix word Reverse N1 and N2 and remove xs 16 bit
TOB;
EXIT 1
2
This entry converts N1 to decimal from binary
Fetch radix word and reverse N1 and N2
FRB;
Insert xs 16 bit
Suppress left-hand zeroes
EXIT 1

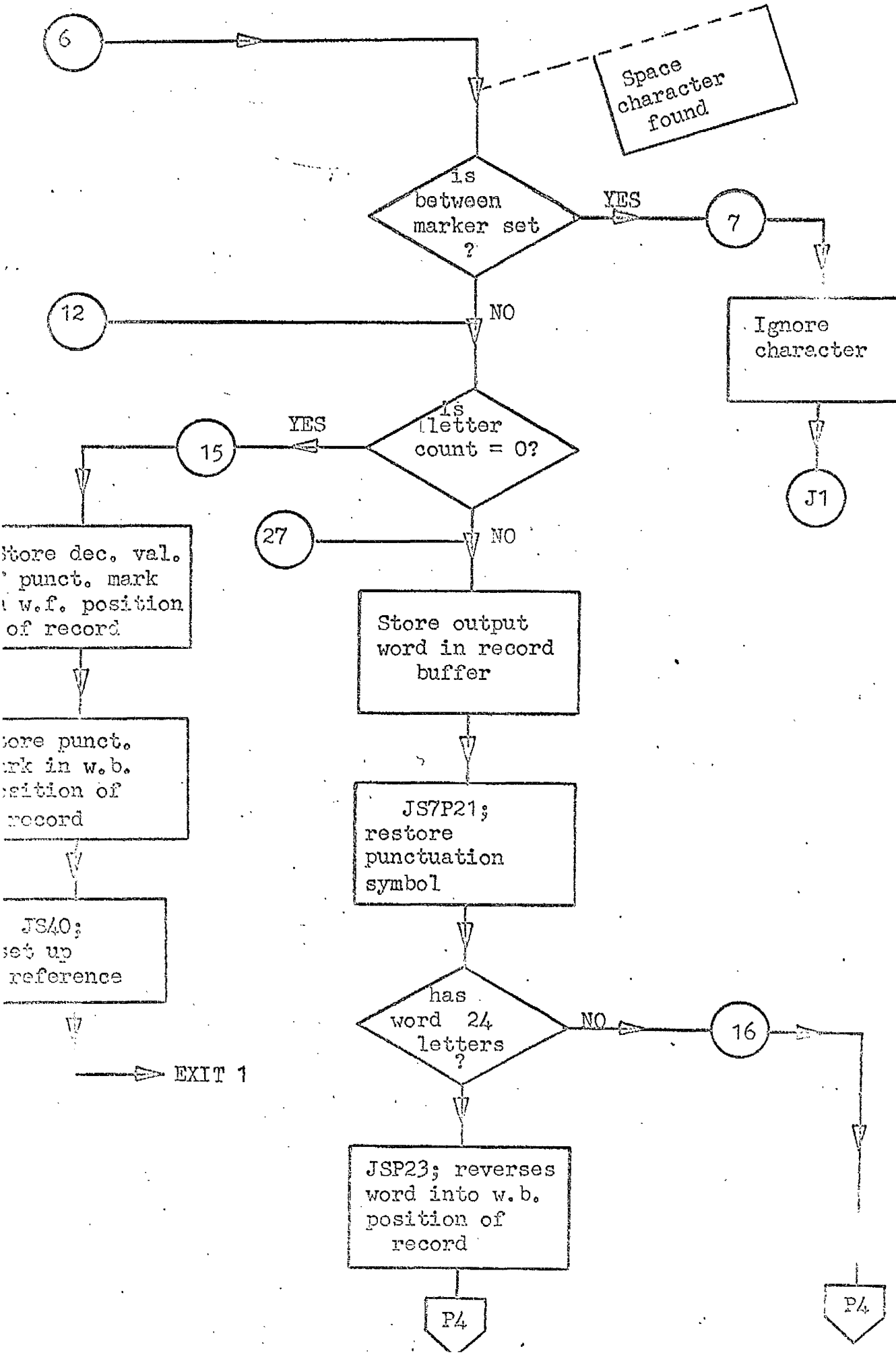

0
This is first entry to claim reader and fill buffers
Preserve Q14 and Q15
JSP55; Fill buffers initially
C14 = 8, to count chars. per register
J5
1
This entry sets next character in N1
Preserve and reset Q14 and Q15
is C14 = 0 ?
YES
J2
NO
P2

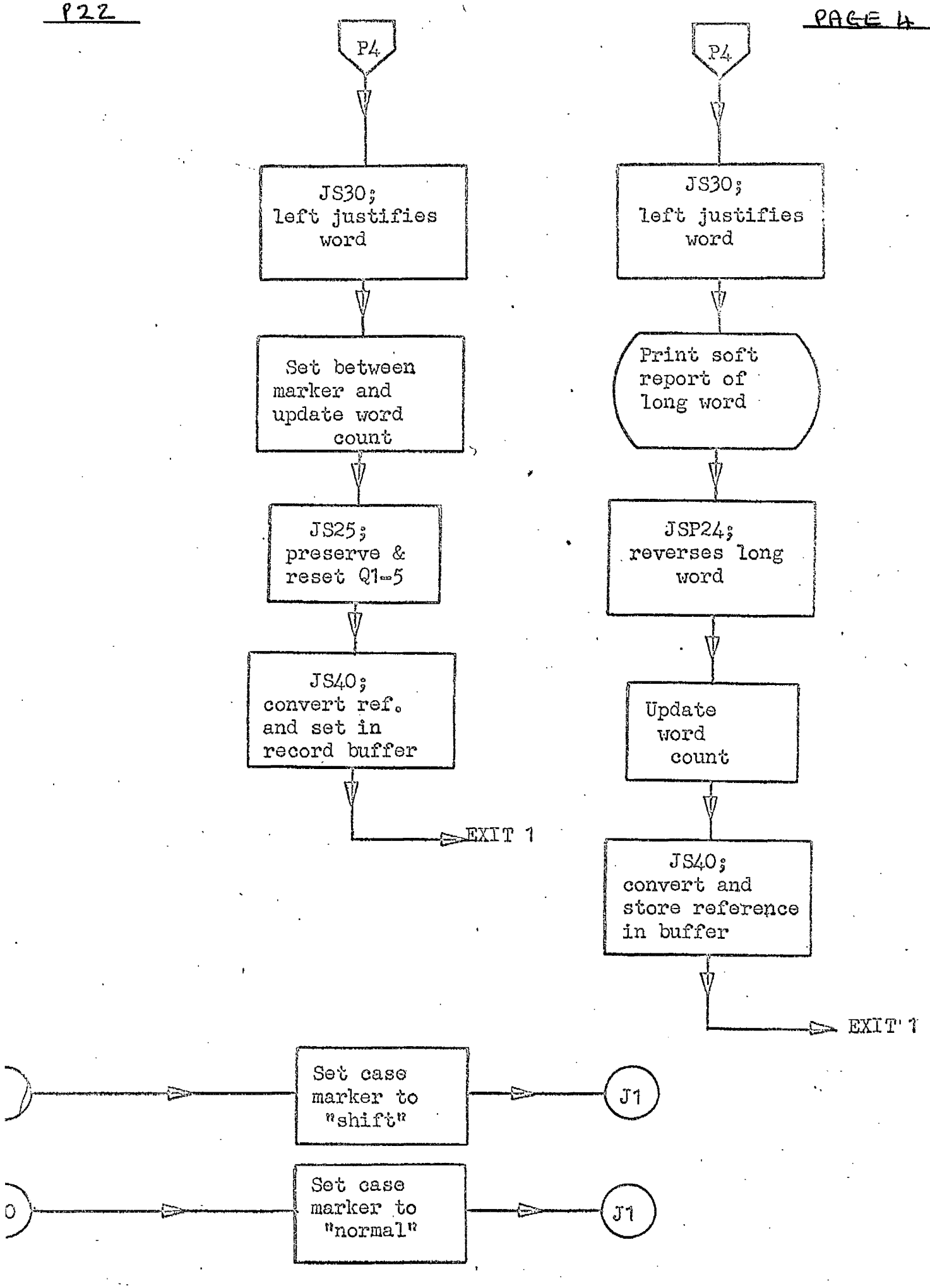

P2
8
Remove next character and restore word
Decrease C14
5
Preserve and reset Q14 and Q15
EXIT 1
2
Decrease buffer count (= C15)
& update M15
is C15 = 0 ?
YES
JS1P55; fill next buffer, C15=64
NO
J3
J3

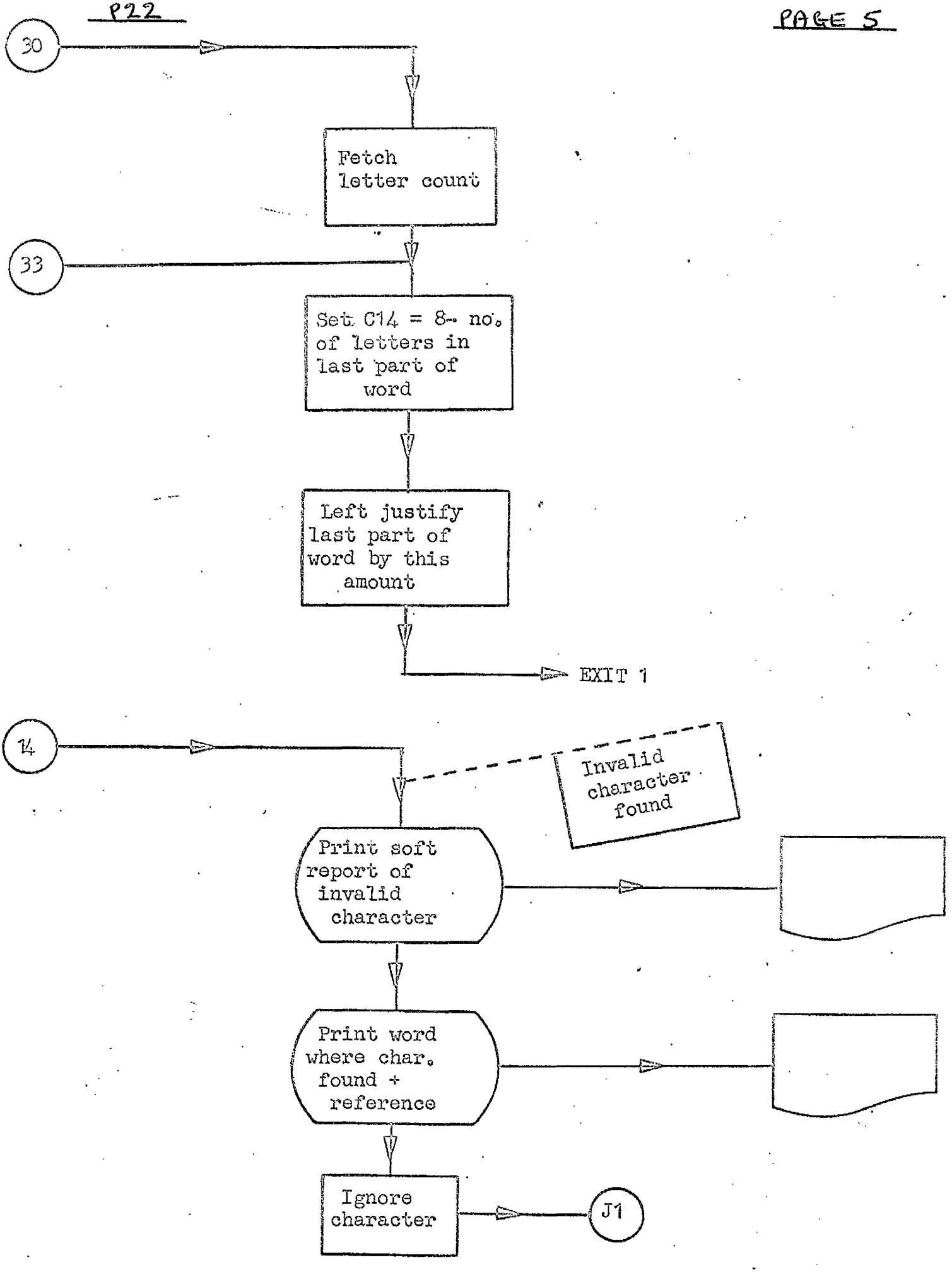

3
Reset
C14 = 8
J8
7
This entry restores char. in N1 into buffer
Store and reset Q14 and Q15
Fetch current register
SHLD-6; and 'OR' in char. in N2
Restore word and update C14
Store and reset Q14 and Q15
EXIT 1

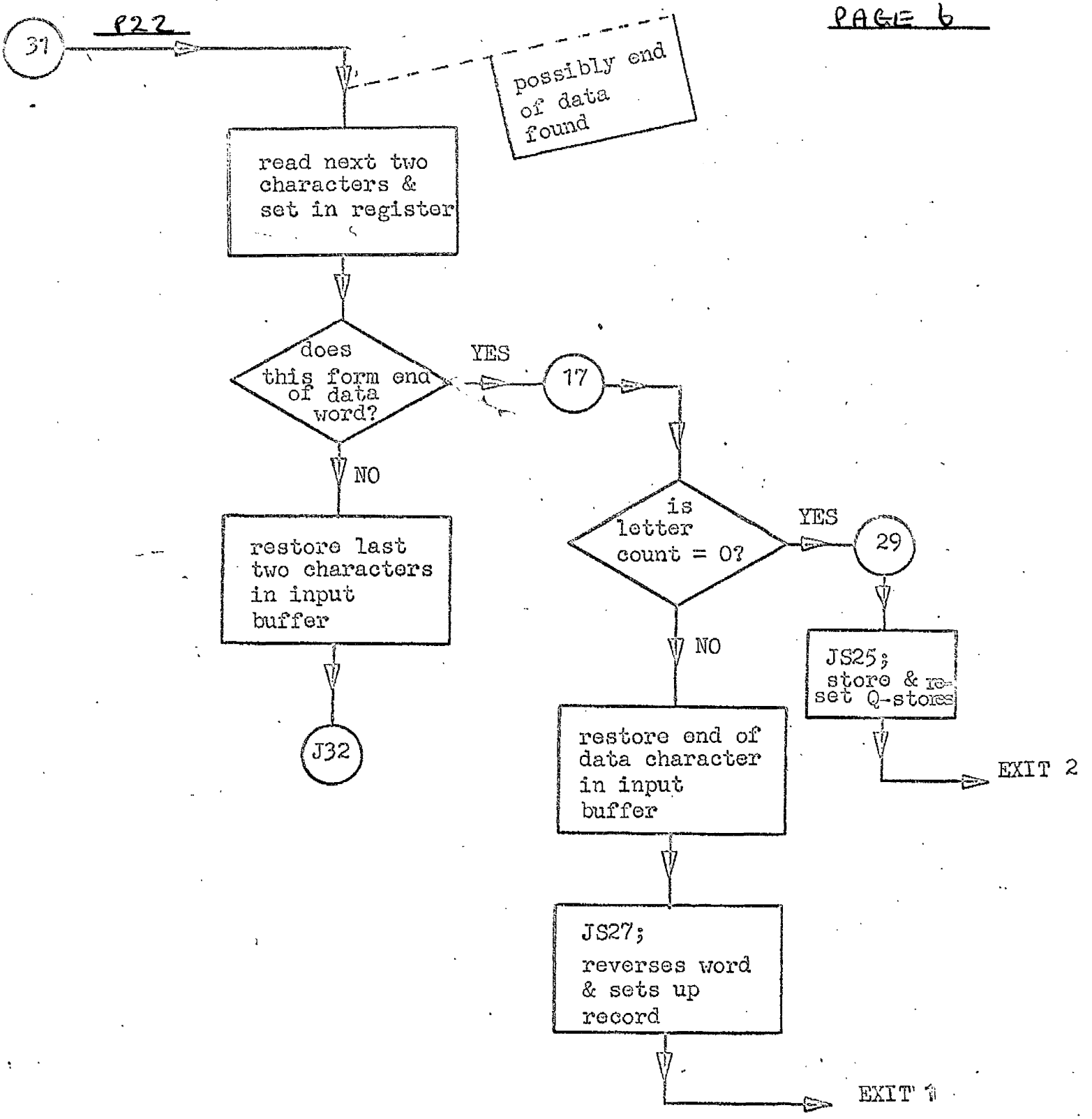

0
Preserve & reset Q1 - 5
letter count, output word, hyphen & accent markers = 0
set M4 = b.a. of jump table
1
JSIP21; fetch next character
3
is char. end of text char.?
YES
J31
NO
= M5; M4M5H; SHL-24; = LINK; EXIT; jumps to relevant label
J12
J7
J32
J14
J20
J18
J9
J10
J8
J34
J19

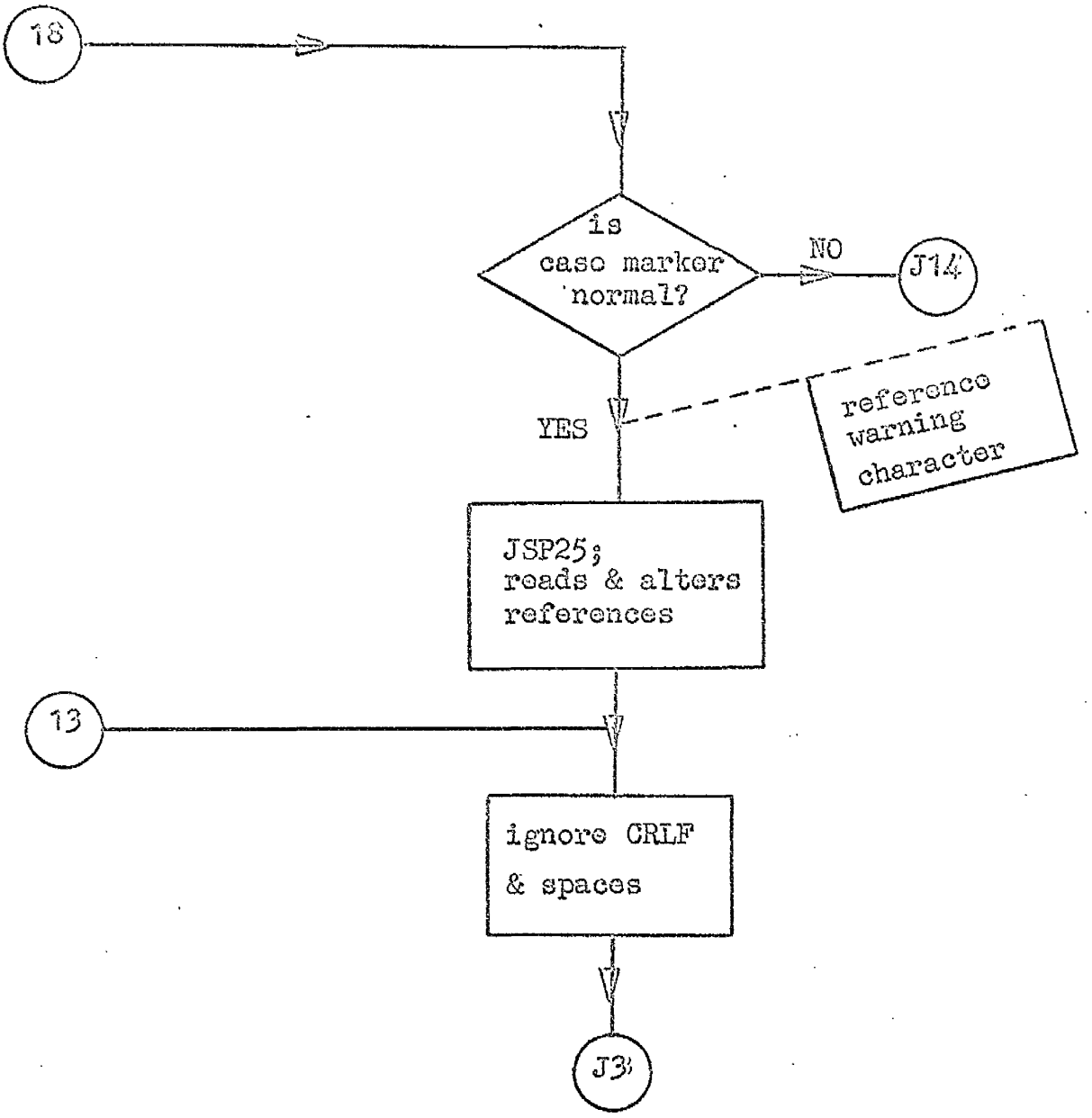

32
a letter has been found
update letter count
is this start of new word ?
YES
4
NO
store letter in output word
clear buffer for setting up next record
is output word full ?
YES
2
NO
set "between" marker off
in record
& clear
t word
restore output word
store letter in output word
J1
J1
J1

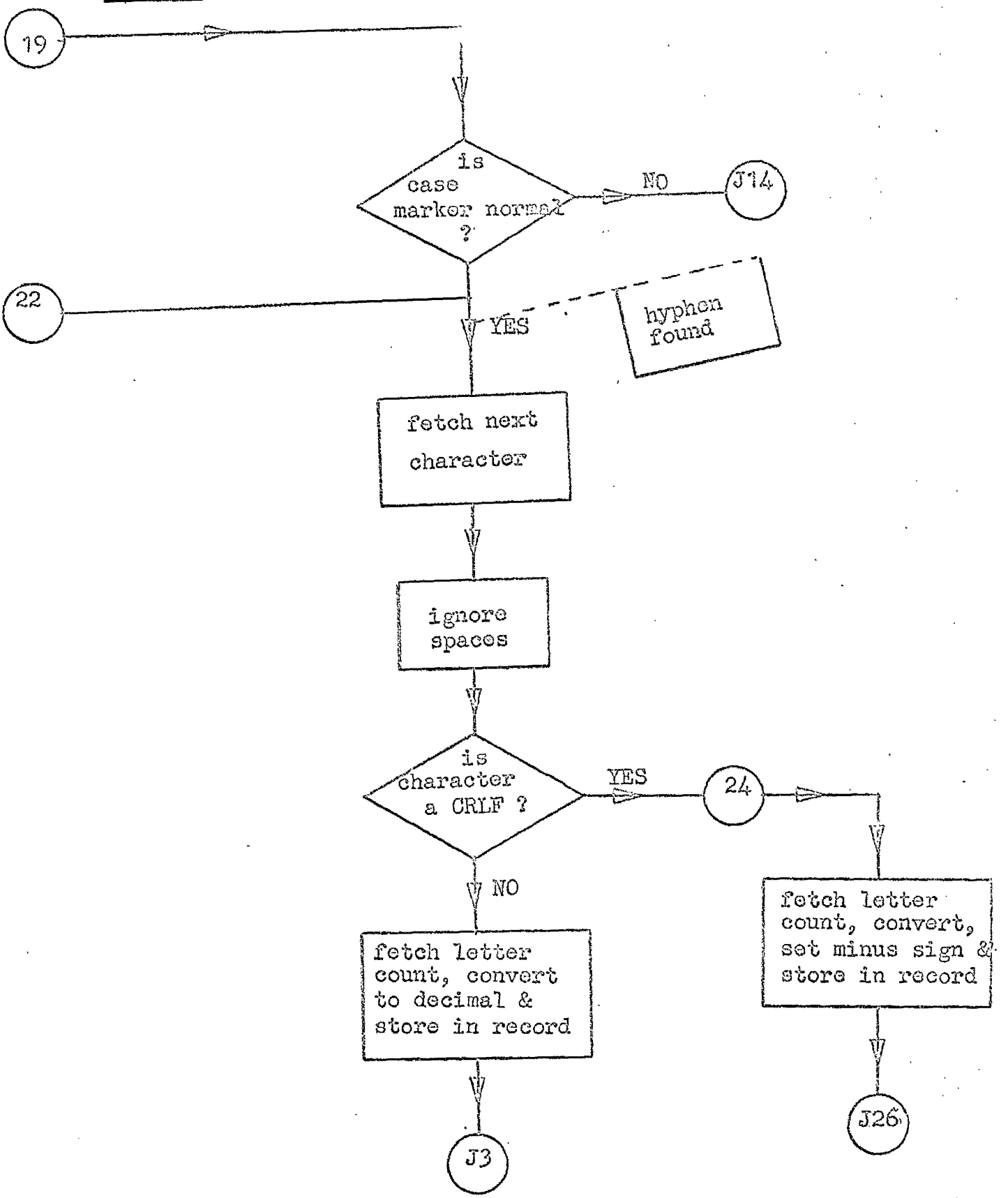

6
Space character found
is between marker set ?
YES
7
NO
12
Ignore character
J1
is letter count = 0?
YES
15
NO
27
Store dec. val. punct. mark w.f. position of record
Store output word in record buffer
ore punct. rk in w.b. sition of record
JS7P21; restore punctuation symbol
JS40; set up reference
has word 24 letters ?
NO
16
EXIT 1
JSP23; reverses word into w.b. position of record
P4
P4

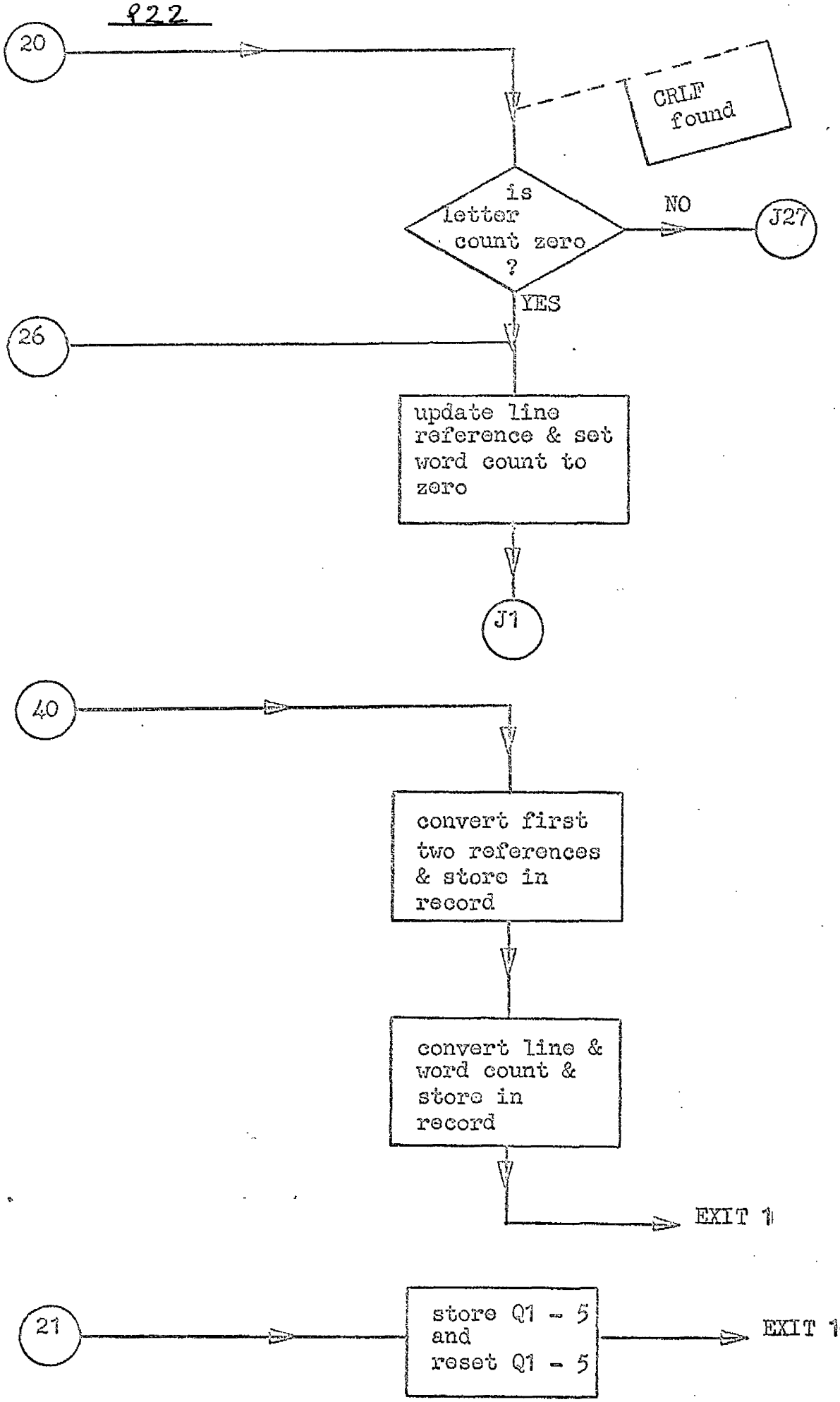

P4
JS30; left justifies word
Set between marker and update word count
JS25; preserve & reset Q1-5
JS40; convert ref. and set in record buffer
EXIT 1
P4
JS30; left justifies word
Print soft report of long word
JSP24; reverses long word
Update word count
JS40; convert and store reference in buffer
EXIT 1
Set case marker to "shift"
J1
Set case marker to "normal"
J1

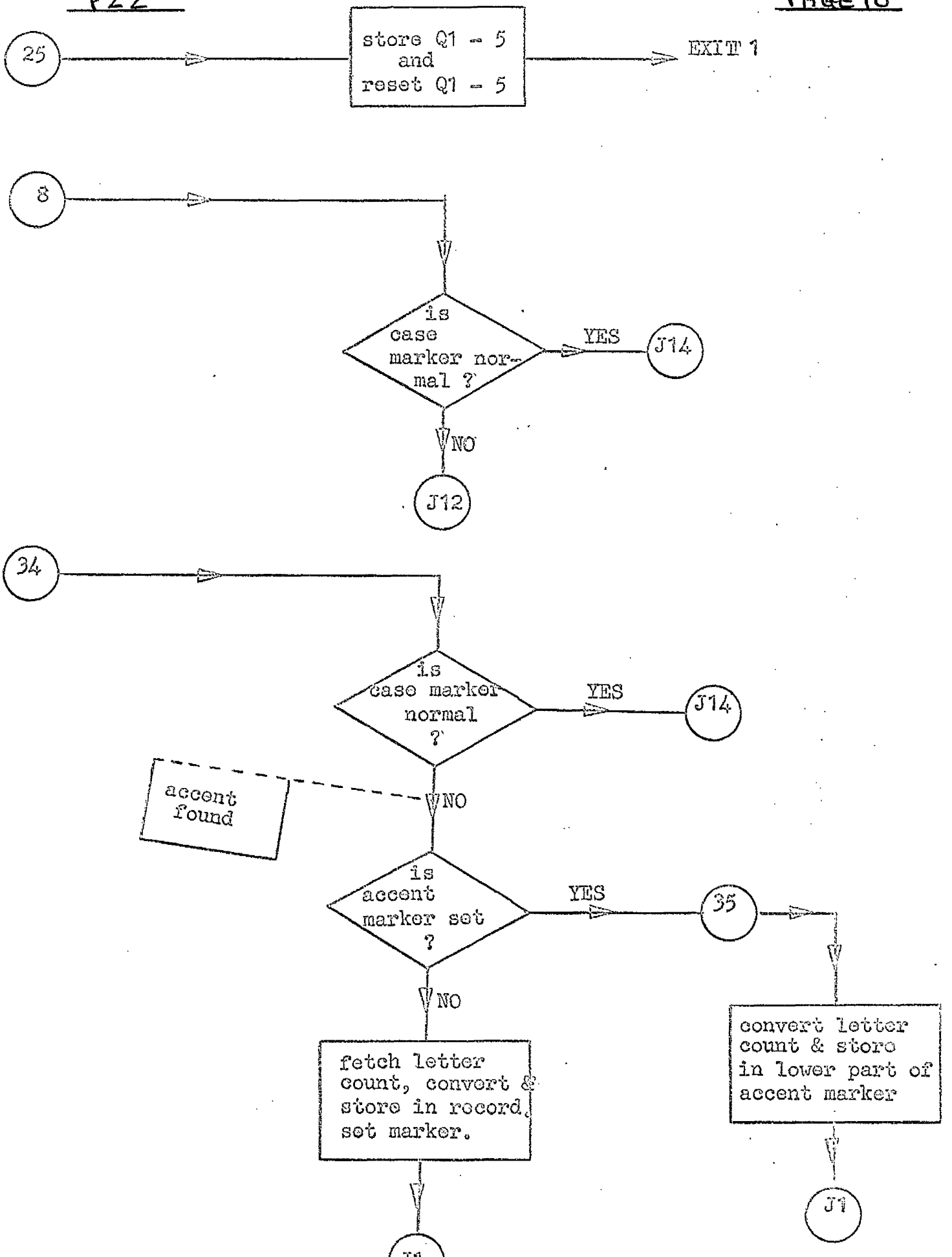

30
Fetch letter count
33
Set C14 = 8- no. of letters in last part of word
Left justify last part of word by this amount
EXIT 1
14
Invalid character found
Print soft report of invalid character
Print word where char. found + reference
Ignore character
J1

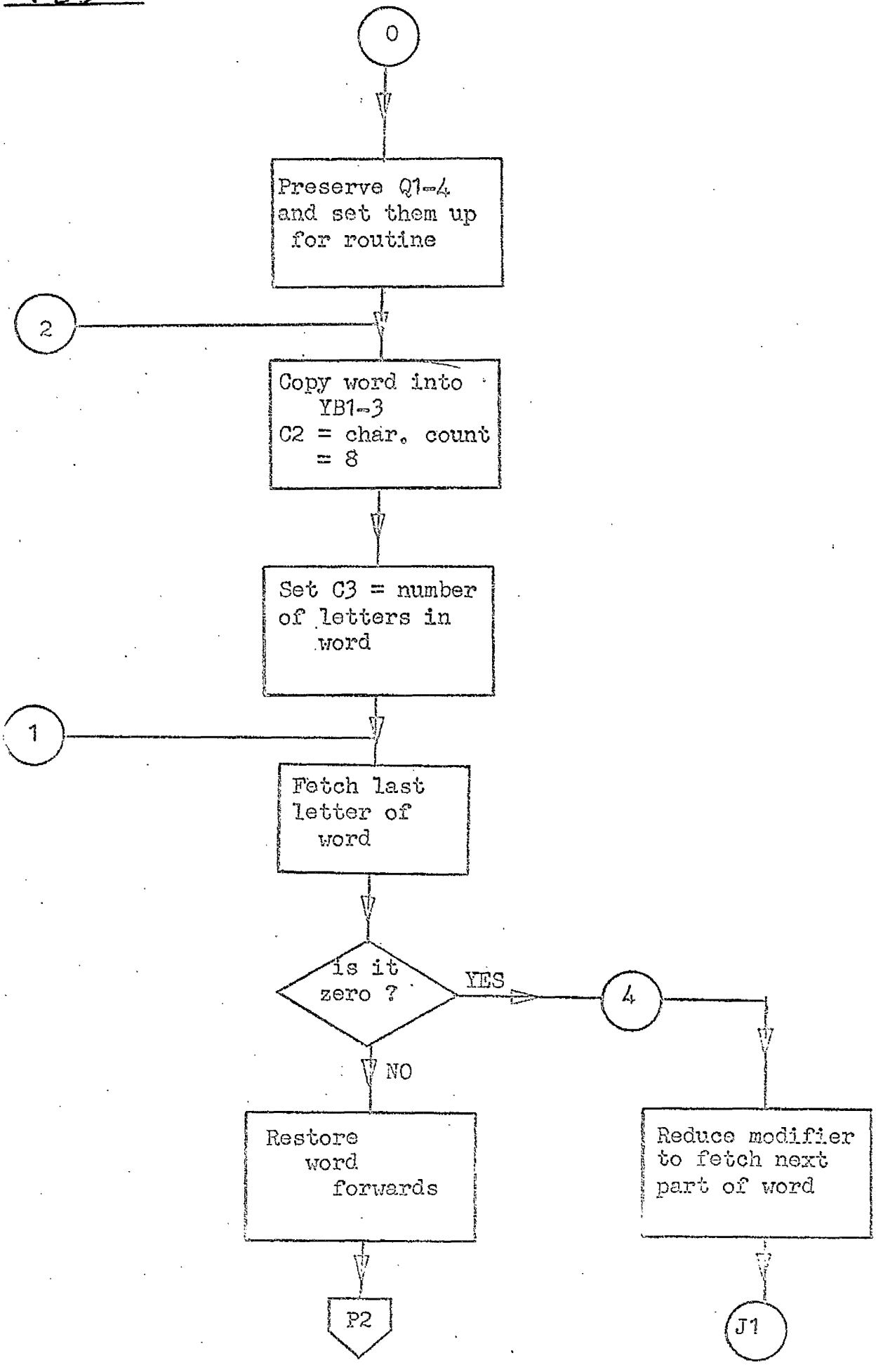

37
P22
possibly end of data found
read next two characters & set in register
does this form end of data word?
YES
17
NO
restore last two characters in input buffer
J32
is letter count = 0?
YES
29
JS25; store & reset Q-stores
EXIT 2
NO
restore end of data character in input buffer
JS27; reverses word & sets up record
EXIT 1

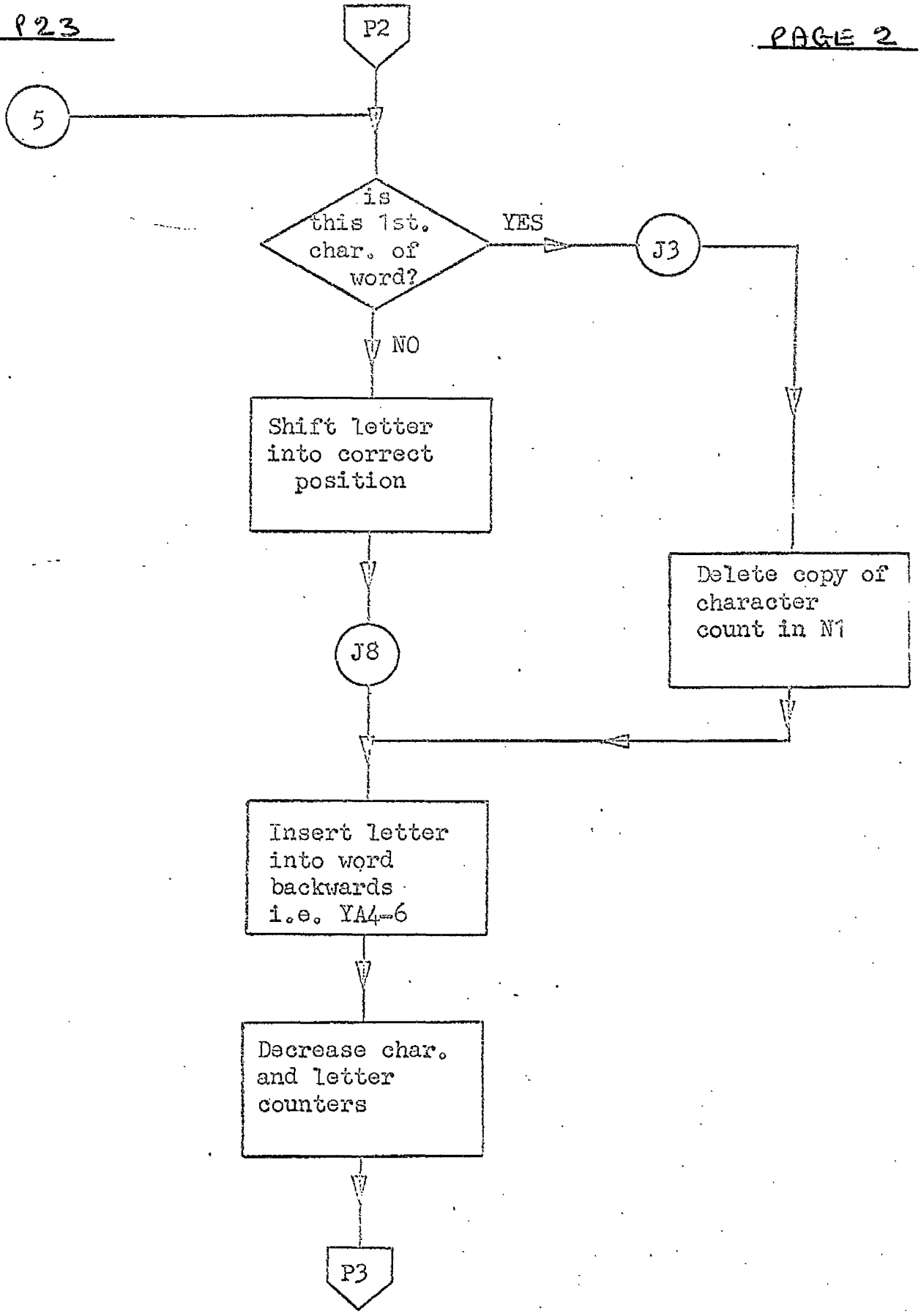

18
is case marker normal?
NO
J14
YES
reference warning character
JSP25; reads & alters references
13
ignore CRLF & spaces
J3

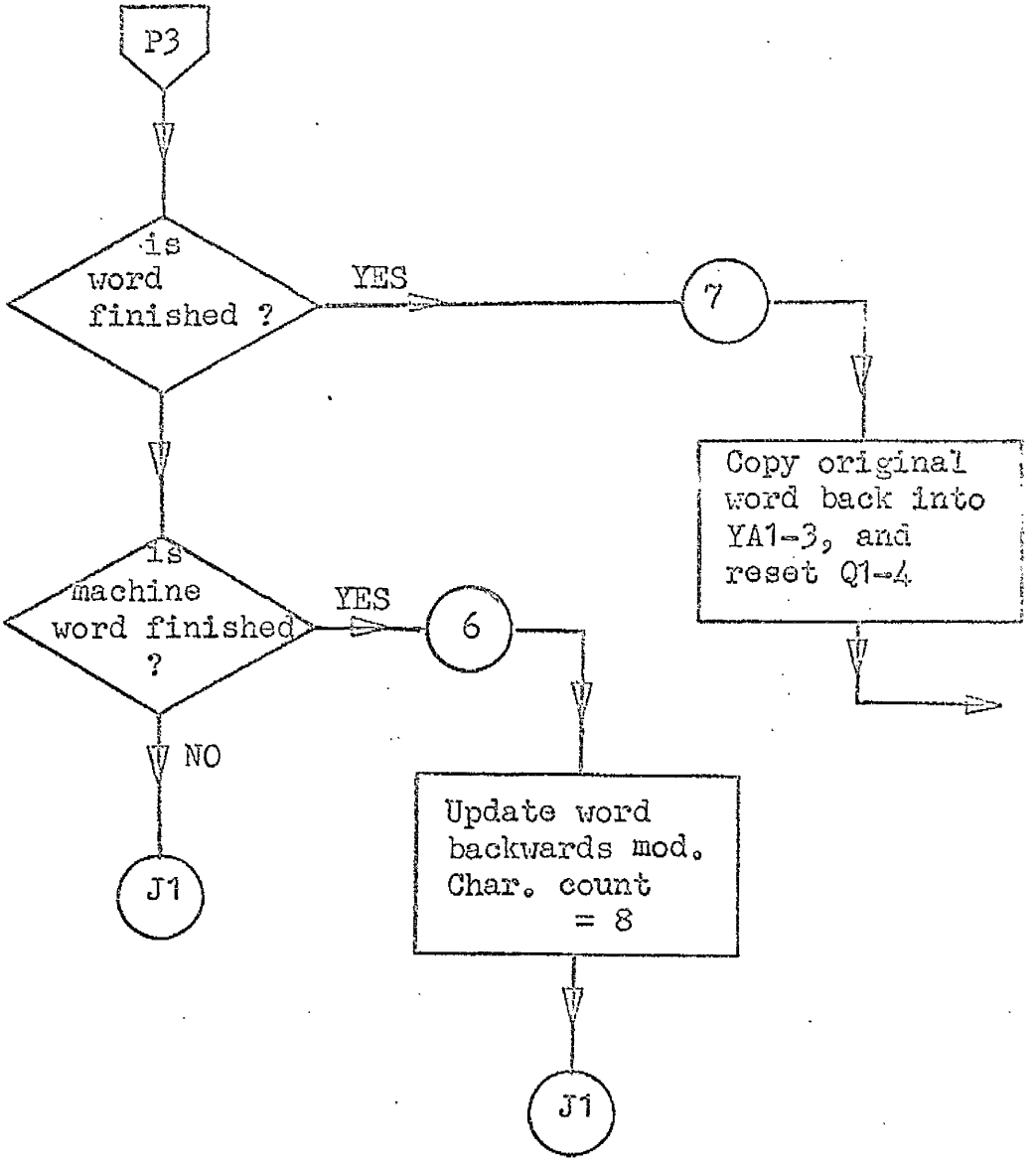

19
is case marker normal ?
NO
J14
22
YES
hyphen found
fetch next character
ignore spaces
is character a CRLF ?
YES
24
NO
fetch letter count, convert to decimal & store in record
fetch letter count, convert, set minus sign & store in record
J3
J26

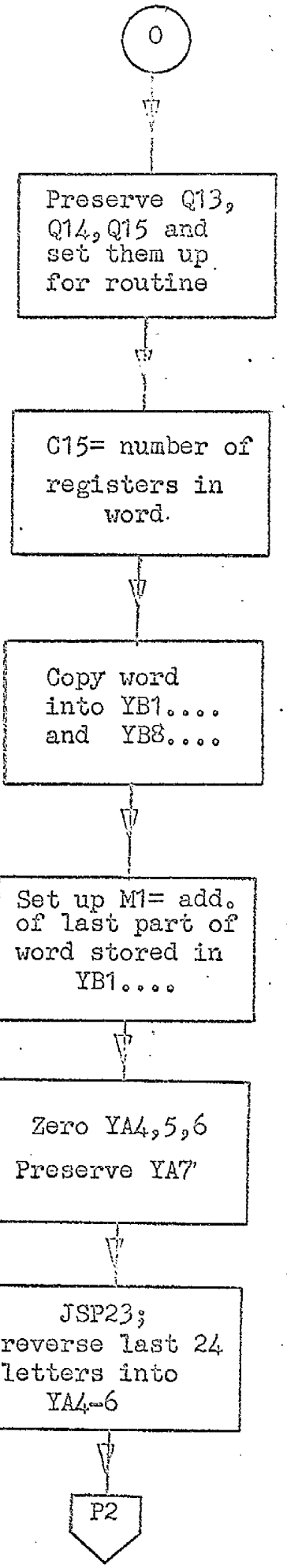

20
CRLF found
is letter count zero ?
NO
J27
YES
26
update line reference & set word count to zero
J1
40
convert first two references & store in record
convert line & word count & store in record
EXIT 1
21
store Q1 - 5 and reset Q1 - 5
EXIT 1

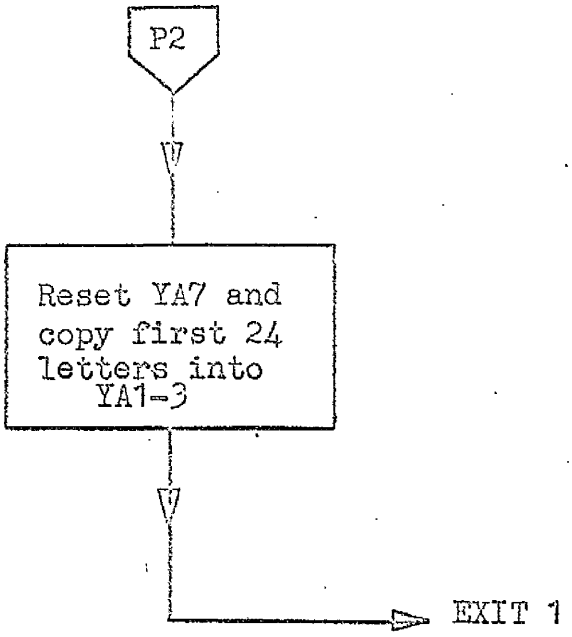

25
store Q1 - 5 and reset Q1 - 5
EXIT 1
8
is case marker normal ?
YES
J14
NO
J12
34
is case marker normal ?
YES
J14
accent found
NO
is accent marker set ?
YES
35
NO
fetch letter count, convert & store in record, set marker.
J1
convert letter count & store in lower part of accent marker
J1

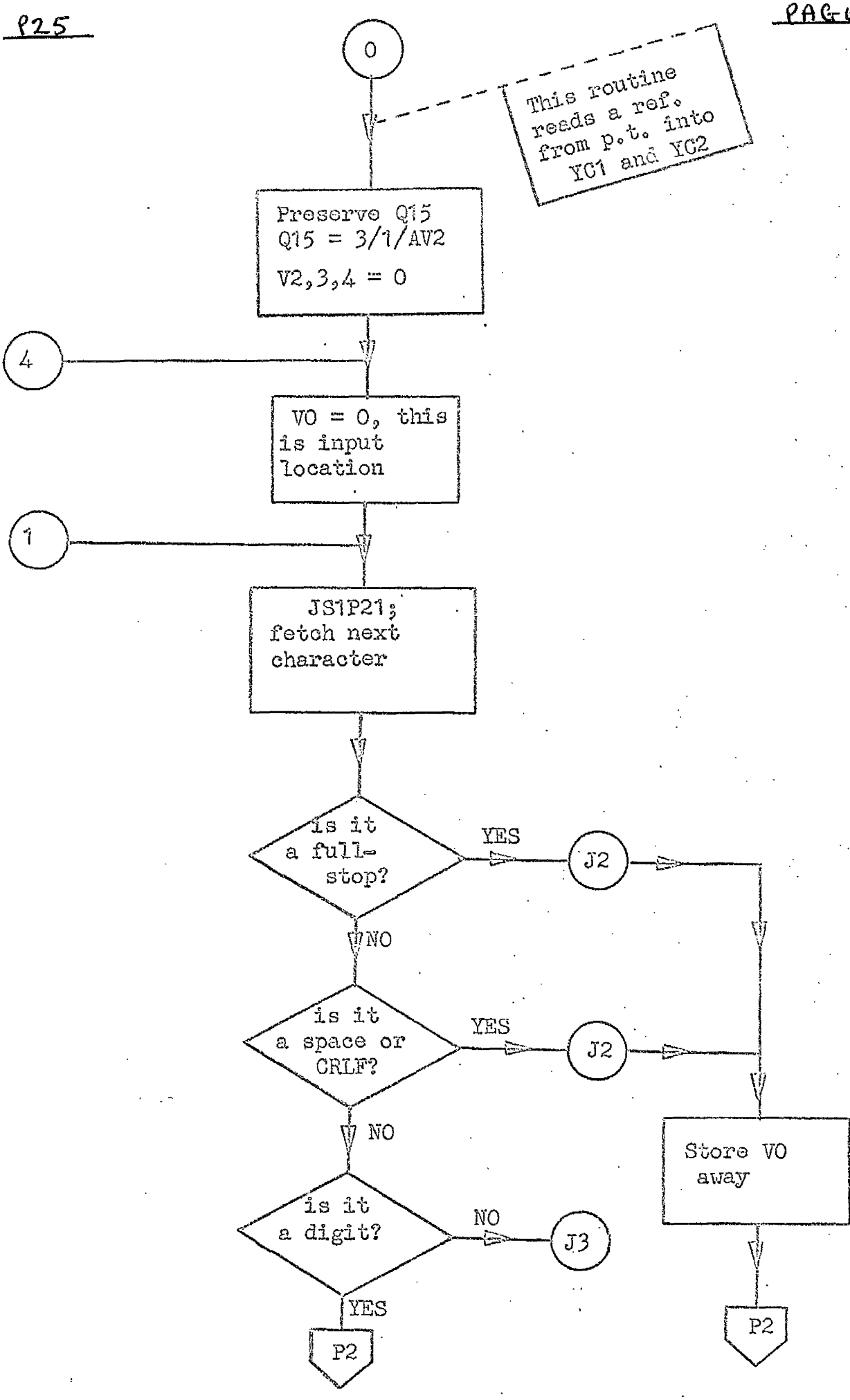

0
Preserve Q1-4 and set them up for routine
2
Copy word into YB1-3
C2 = char. count = 8
Set C3 = number of letters in word
1
Fetch last letter of word
is it zero ?
YES
4
NO
Restore word forwards
Reduce modifier to fetch next part of word
P2
J1

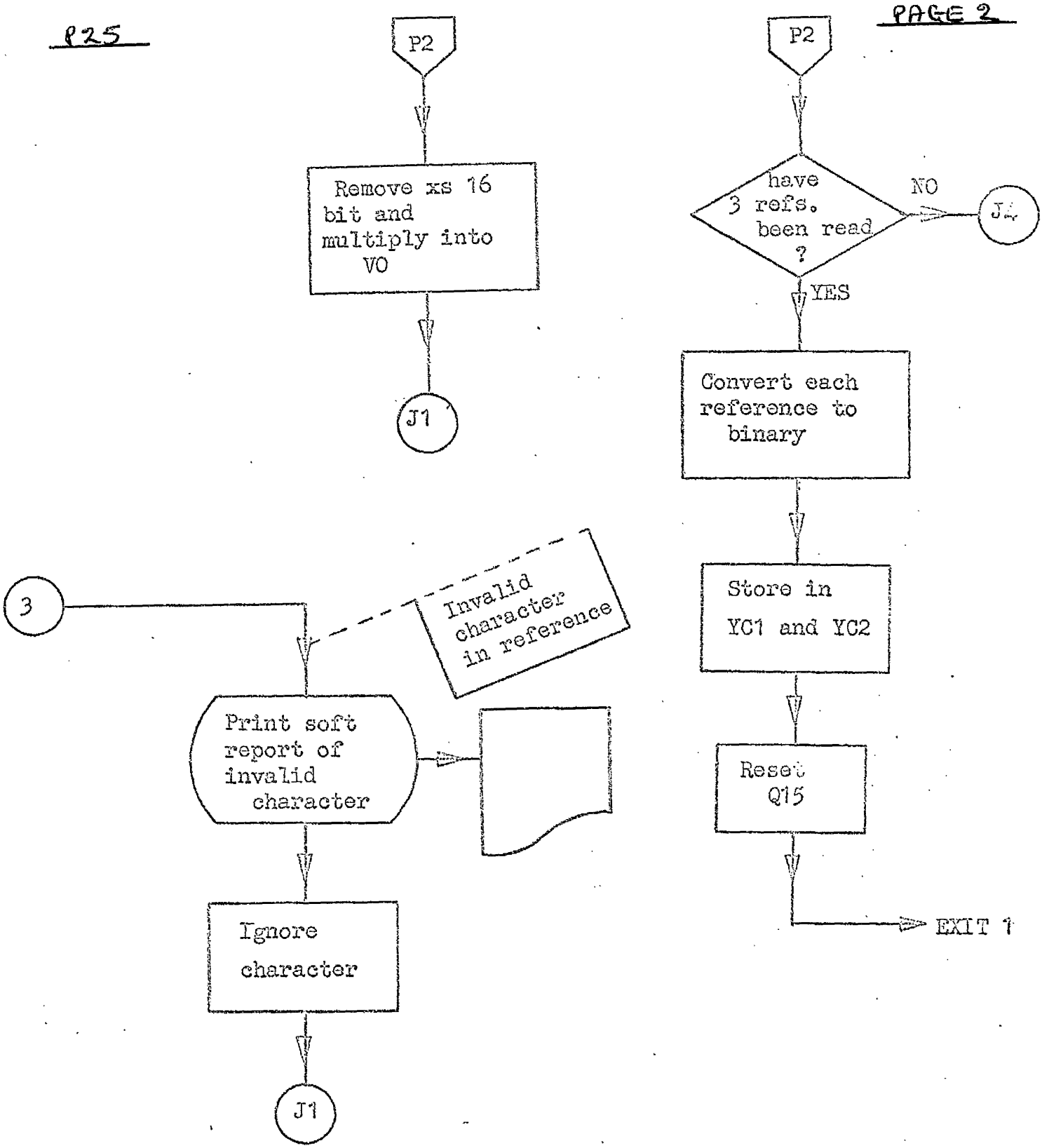

P2
5
is this 1st. char. of word?
YES
J3
NO
Shift letter into correct position
Delete copy of character count in N1
J8
Insert letter into word backwards i.e. YA4-6
Decrease char. and letter counters
P3

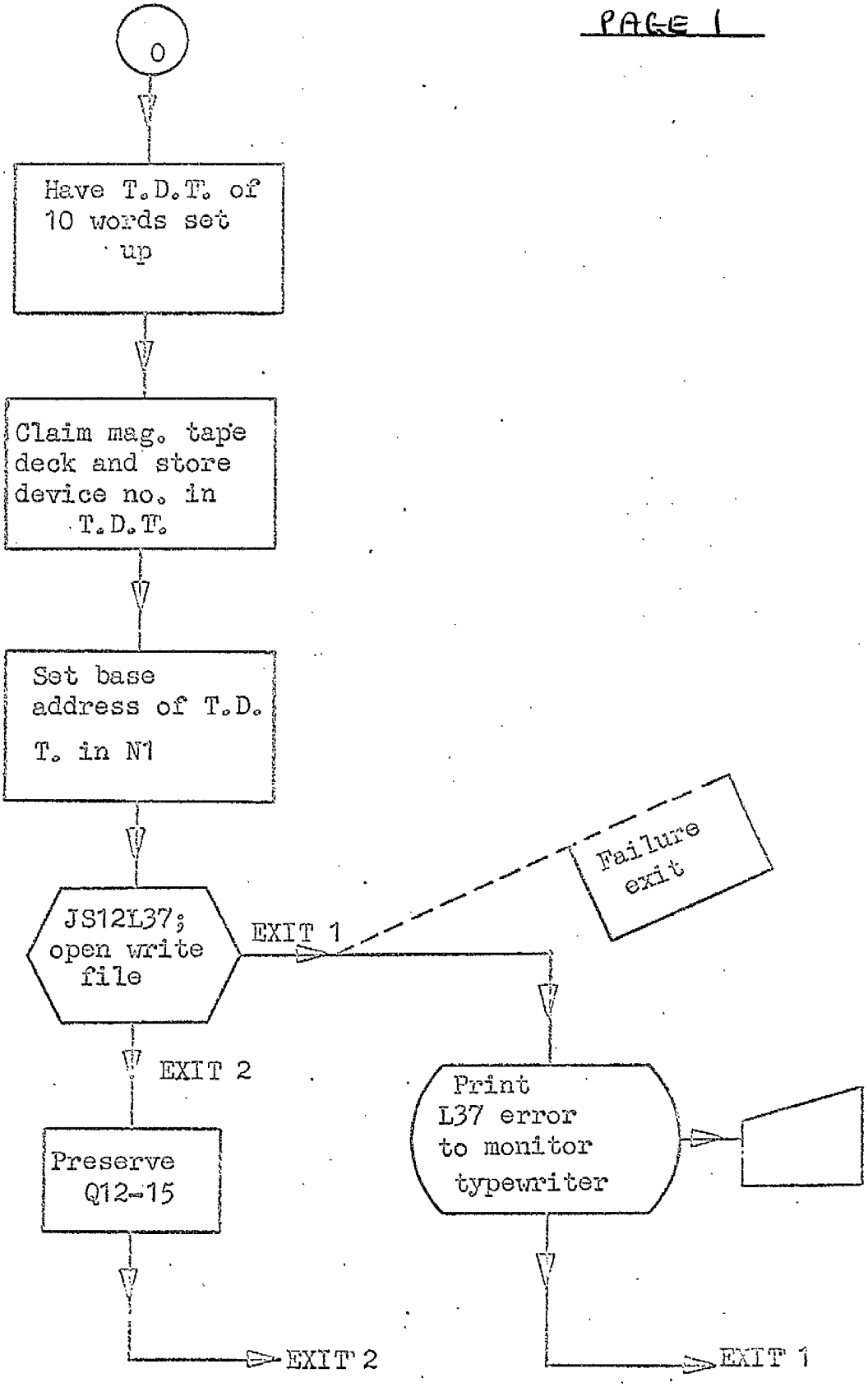

P3
is word finished ?
YES
7
Copy original word back into YA1-3, and reset Q1-4
is machine word finished ?
YES
6
NO
J1
Update word backwards mod. Char. count = 8
J1

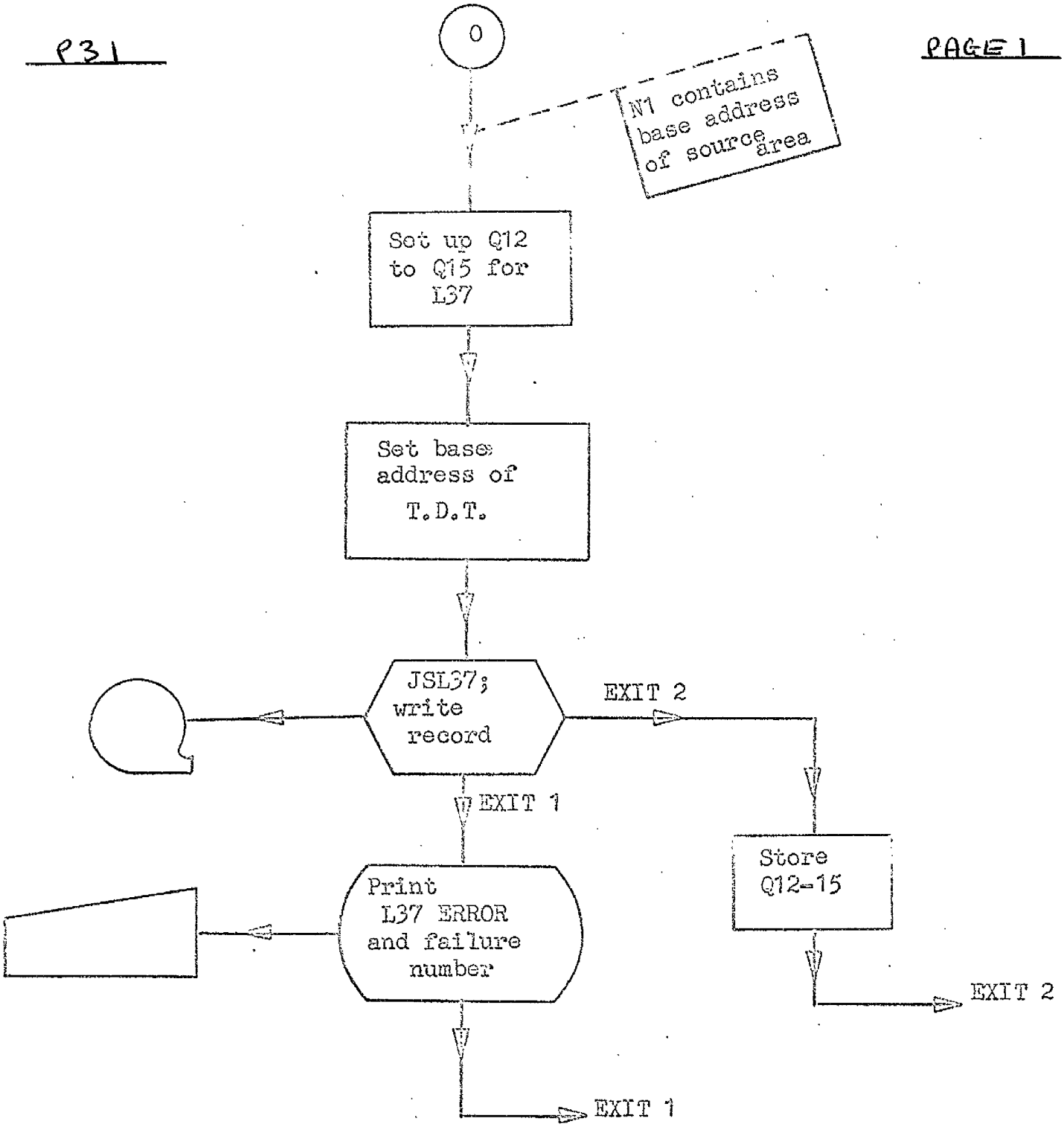

0
Preserve Q13, Q14, Q15 and set them up for routine
C15= number of registers in word.
Copy word into YB1.... and YB8....
Set up M1= add. of last part of word stored in YB1....
Zero YA4,5,6
Preserve YA7'
JSP23; reverse last 24 letters into YA4-6
P2

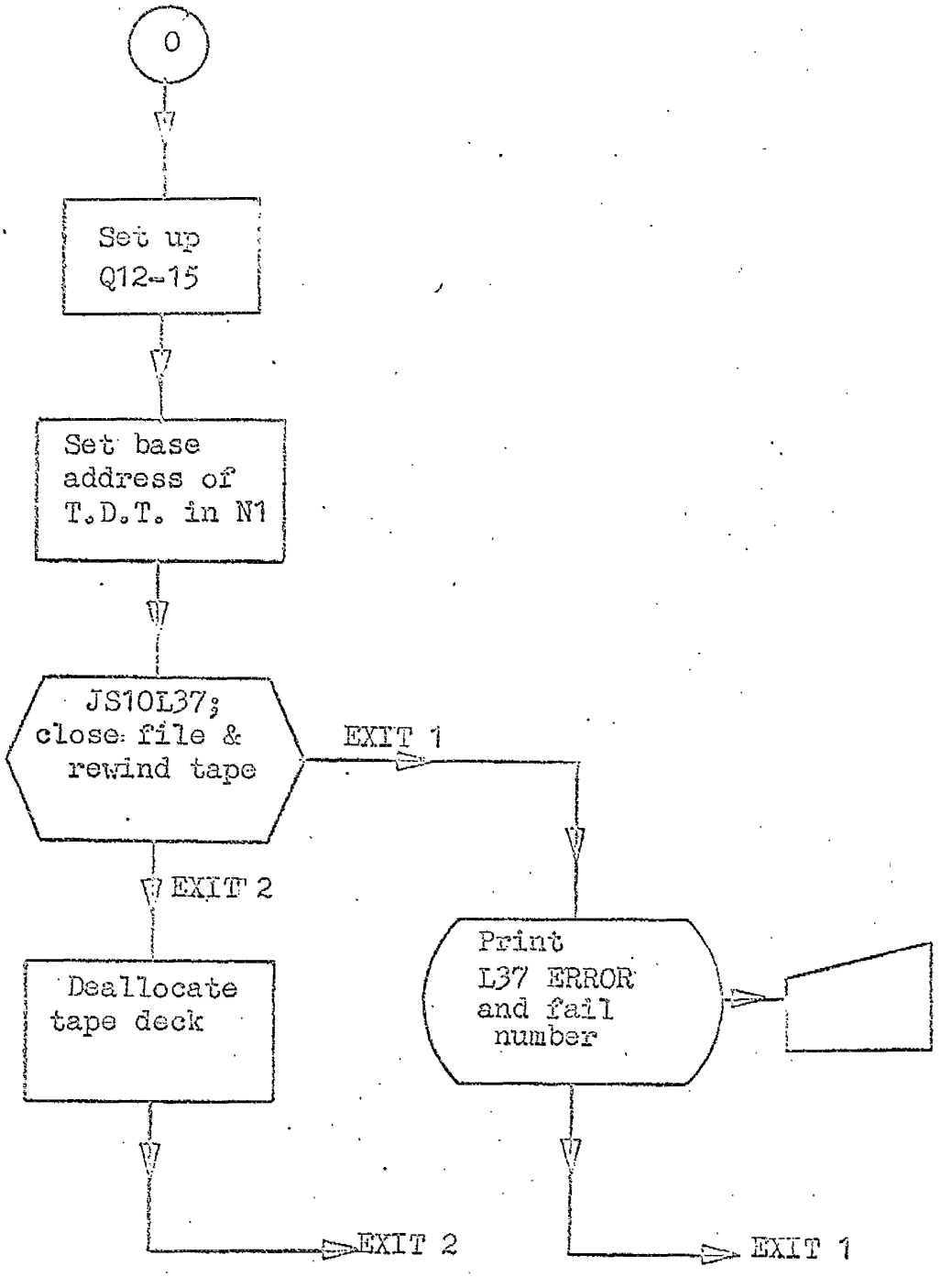

P2
Reset YA7 and copy first 24 letters into YA1-3
EXIT 1

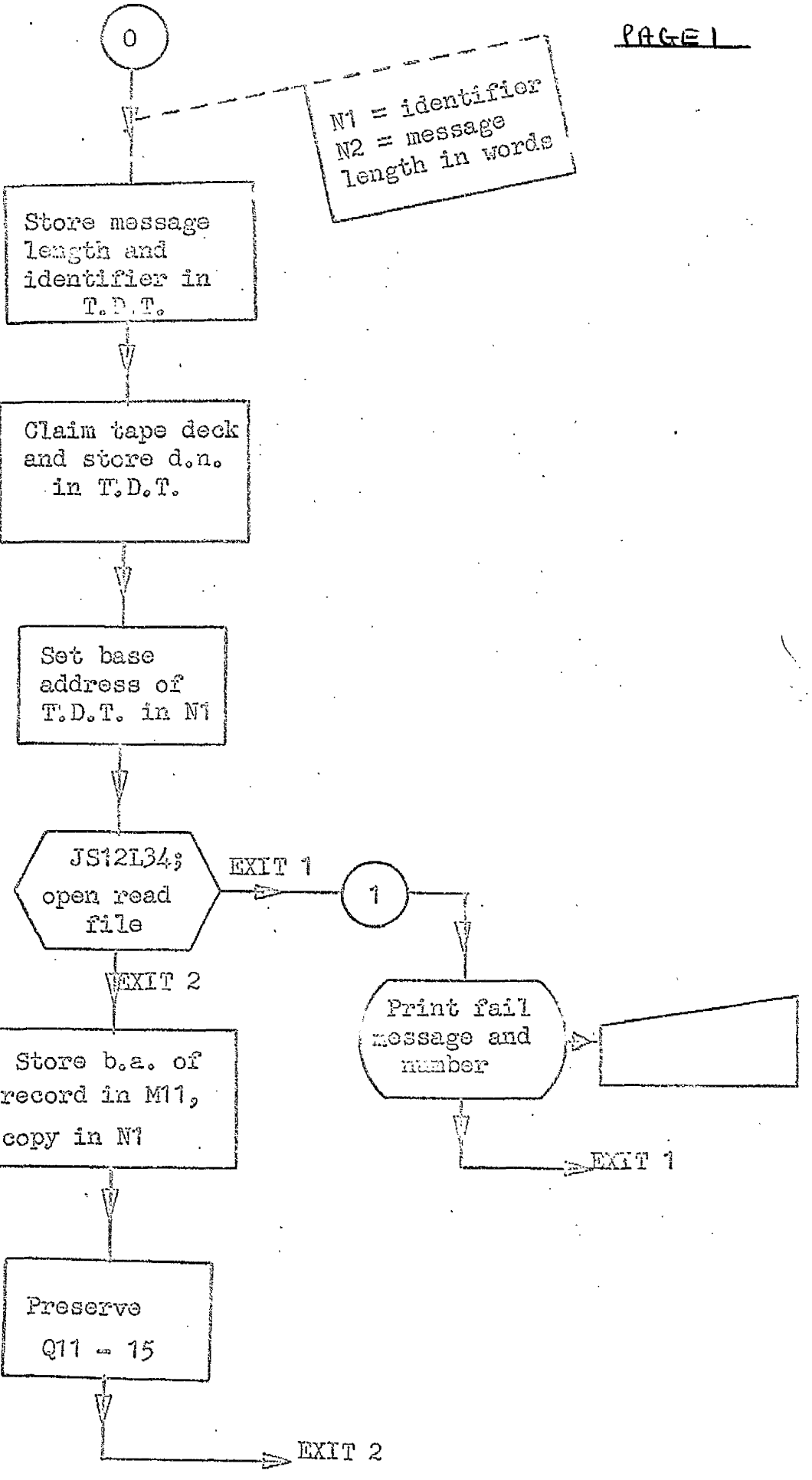

0
This routine reads a ref. from p.t. into YC1 and YC2
Preserve Q15
Q15 = 3/1/AV2
V2,3,4 = 0
4
V0 = 0, this is input location
1
JS1P21; fetch next character
is it a full-stop?
YES
J2
NO
is it a space or CRLF?
YES
J2
NO
is it a digit?
NO
J3
YES
P2
Store V0 away
P2

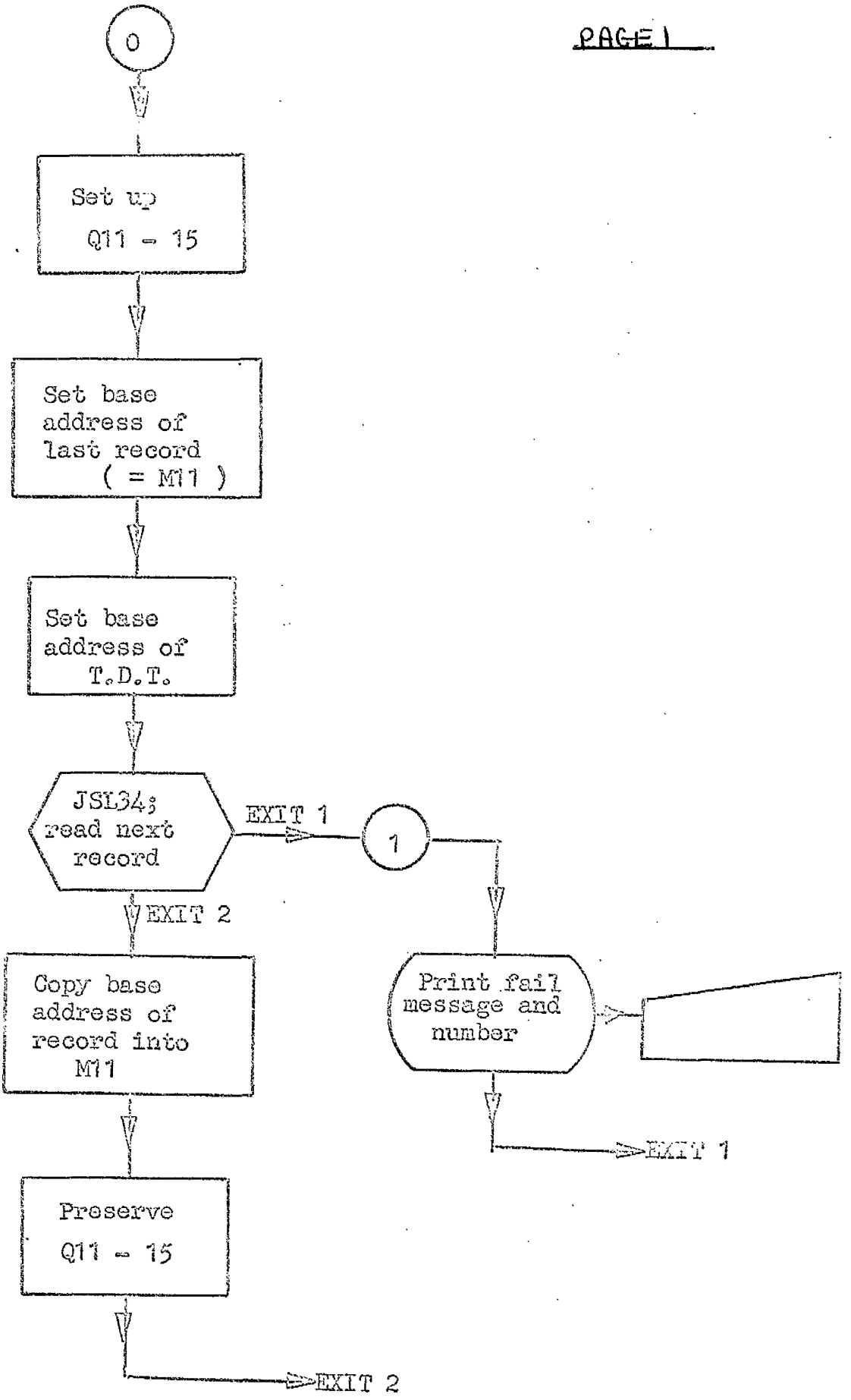

P2
Remove xs 16 bit and multiply into VO
J1
P2
have 3 refs. been read ?
NO
J4
YES
Convert each reference to binary
Store in YC1 and YC2
Reset Q15
EXIT 1
3
Invalid character in reference
Print soft report of invalid character
Ignore character
J1

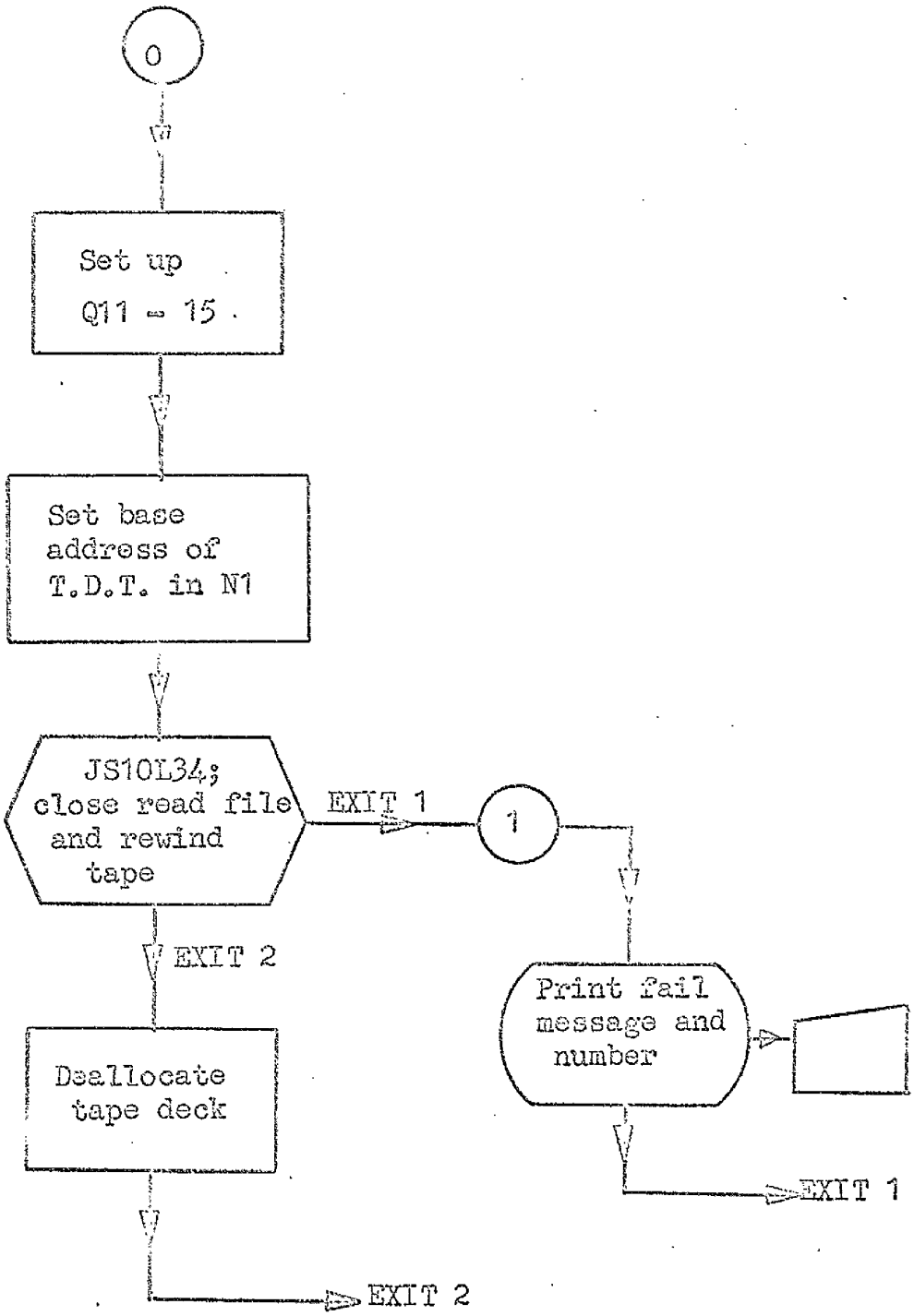

0
Have T.D.T. of 10 words set up
Claim mag. tape deck and store device no. in T.D.T.
Set base address of T.D.T. in N1
JS12L37; open write file
EXIT 1
Failure exit
EXIT 2
Preserve Q12-15
Print L37 error to monitor typewriter
EXIT 2
EXIT 1

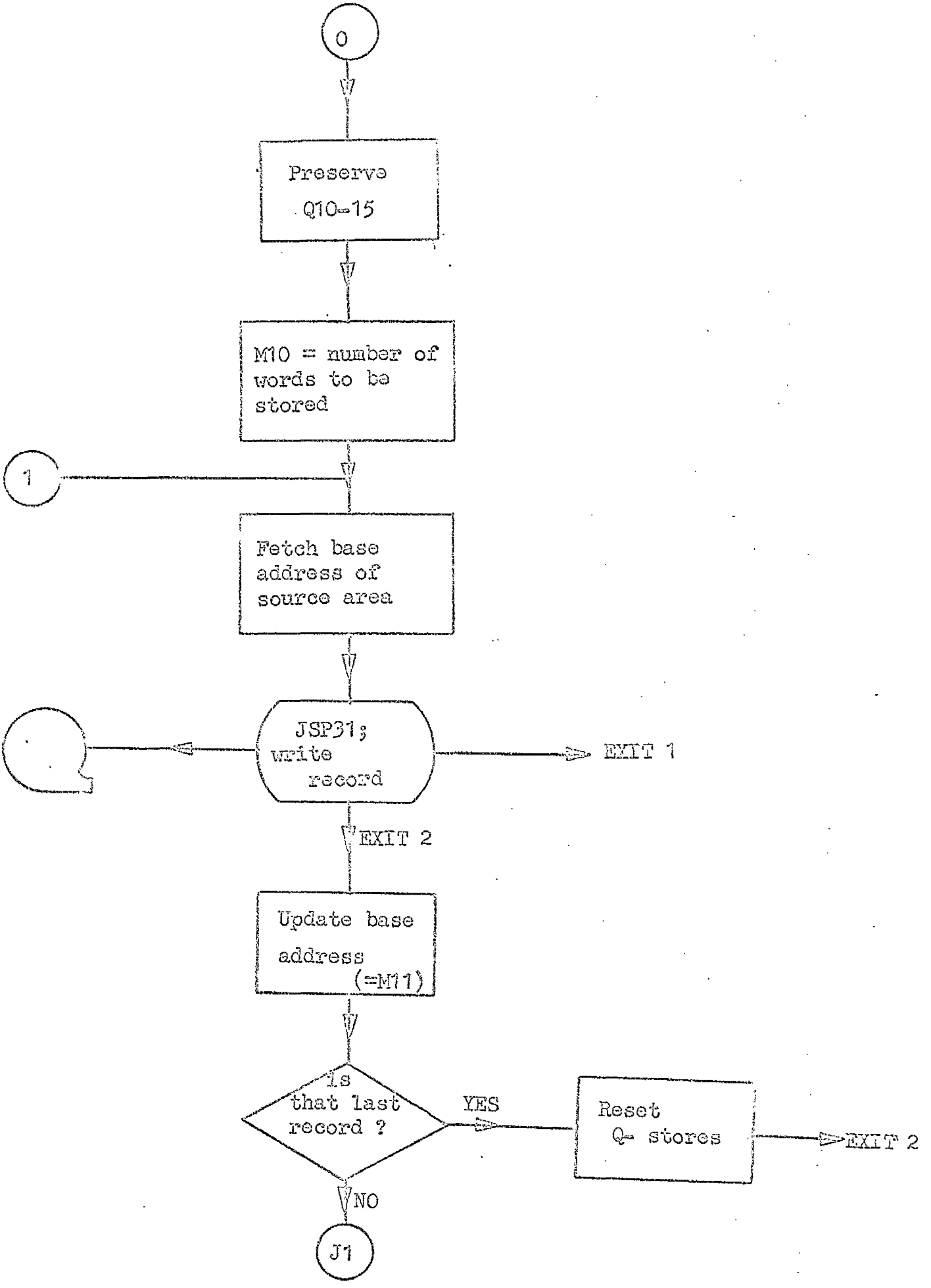

0
N1 contains base address of source area
Set up Q12 to Q15 for L37
Set base address of T.D.T.
JSL37; write record
EXIT 2
EXIT 1
Store Q12-15
Print L37 ERROR and failure number
EXIT 2
EXIT 1

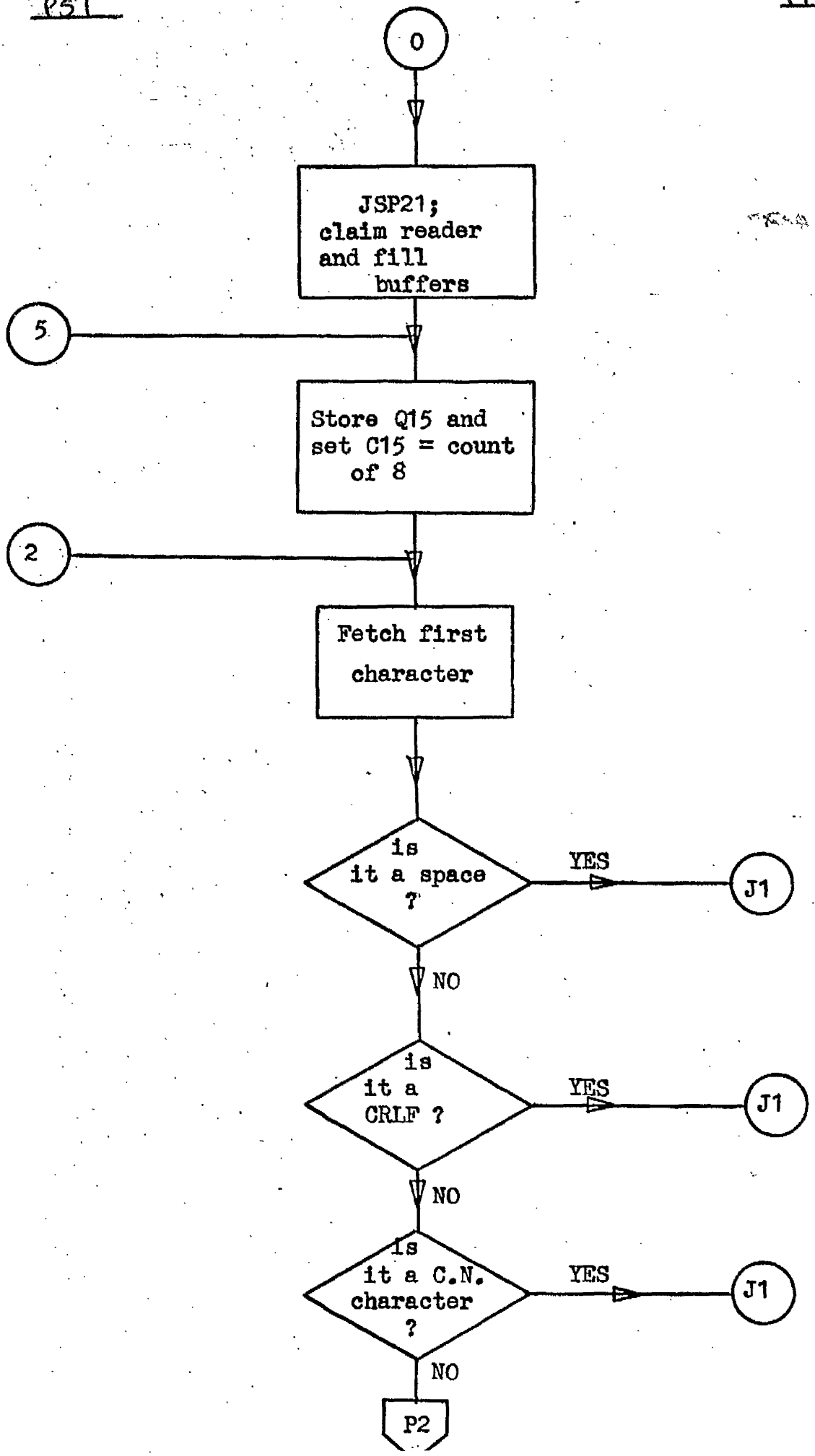

0
Set up Q12-15
Set base address of T.D.T. in N1
JS10L37; close file & rewind tape
EXIT 1
EXIT 2
Deallocate tape deck
Print L37 ERROR and fail number
EXIT 2
EXIT 1

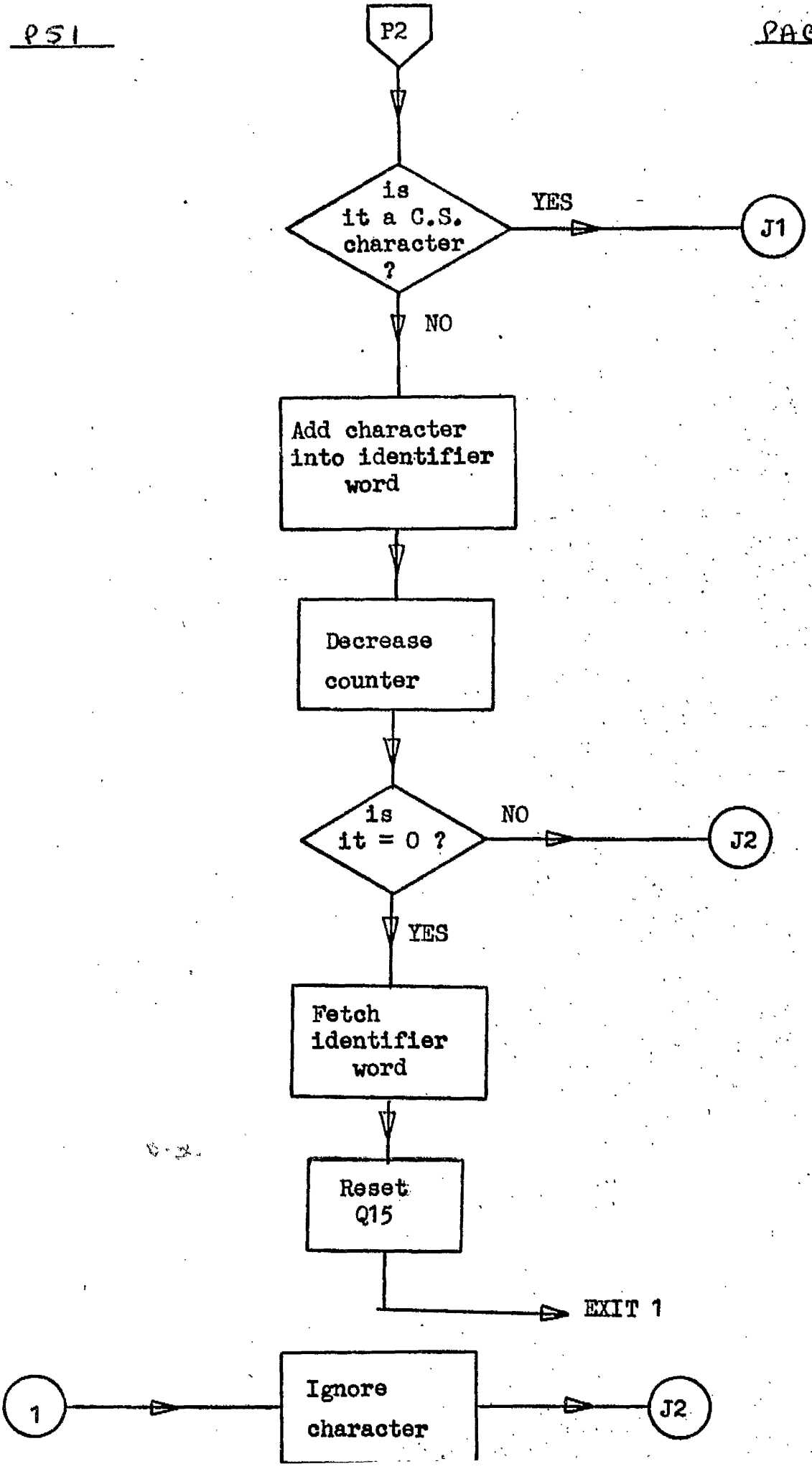

0
N1 = identifier
N2 = message length in words
Store message length and identifier in T.D.T.
Claim tape deck and store d.n. in T.D.T.
Set base address of T.D.T. in N1
JS12L34; open read file
EXIT 1
1
EXIT 2
Print fail message and number
Store b.a. of record in M11, copy in N1
EXIT 1
Preserve Q11 - 15
EXIT 2

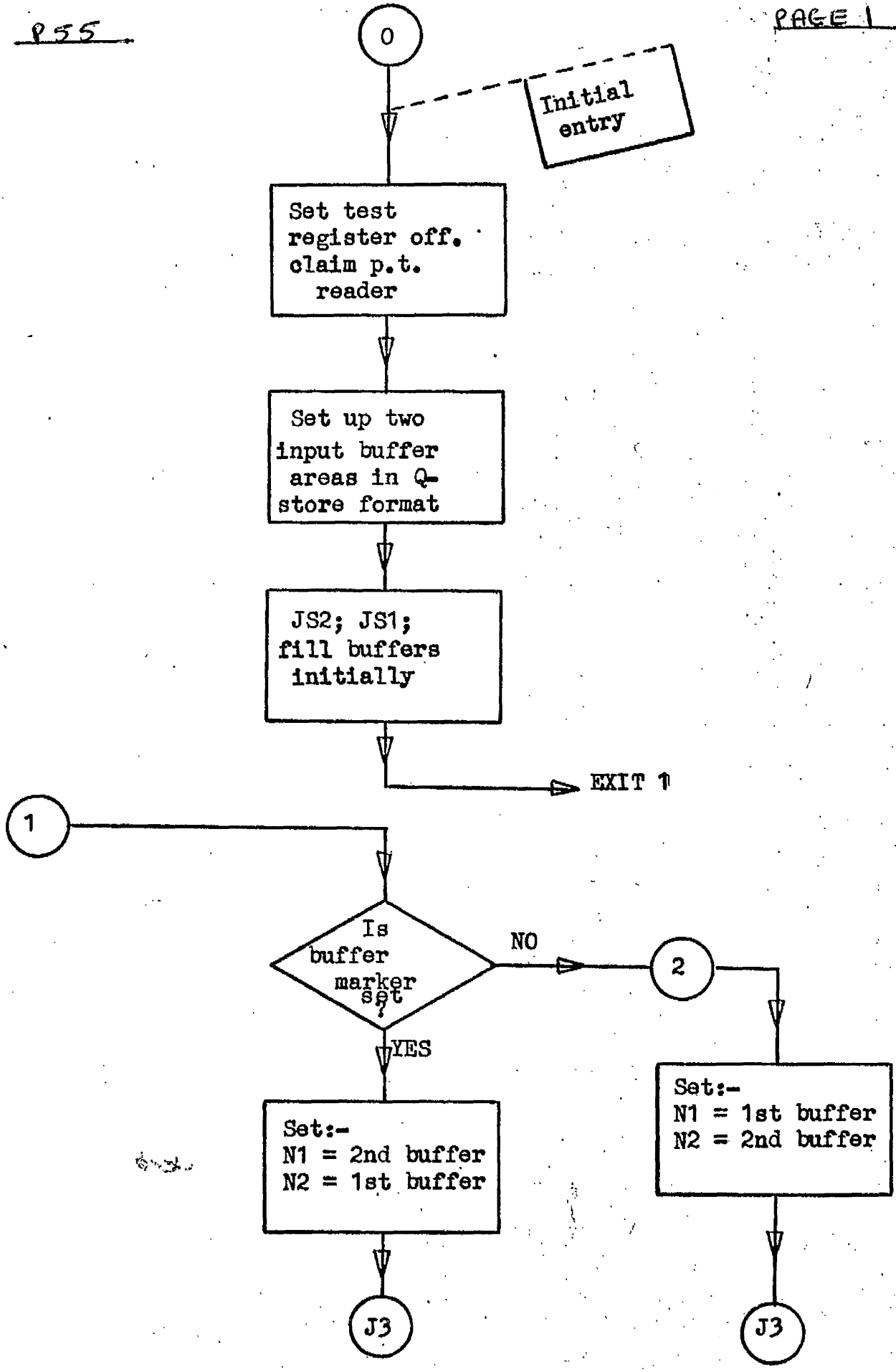

0
Set up
Q11 - 15
Set base address of last record (= M11)
Set base address of T.D.T.
JSL34; read next record
EXIT 1
1
EXIT 2
Copy base address of record into M11
Print fail message and number
EXIT 1
Preserve
Q11 - 15
EXIT 2

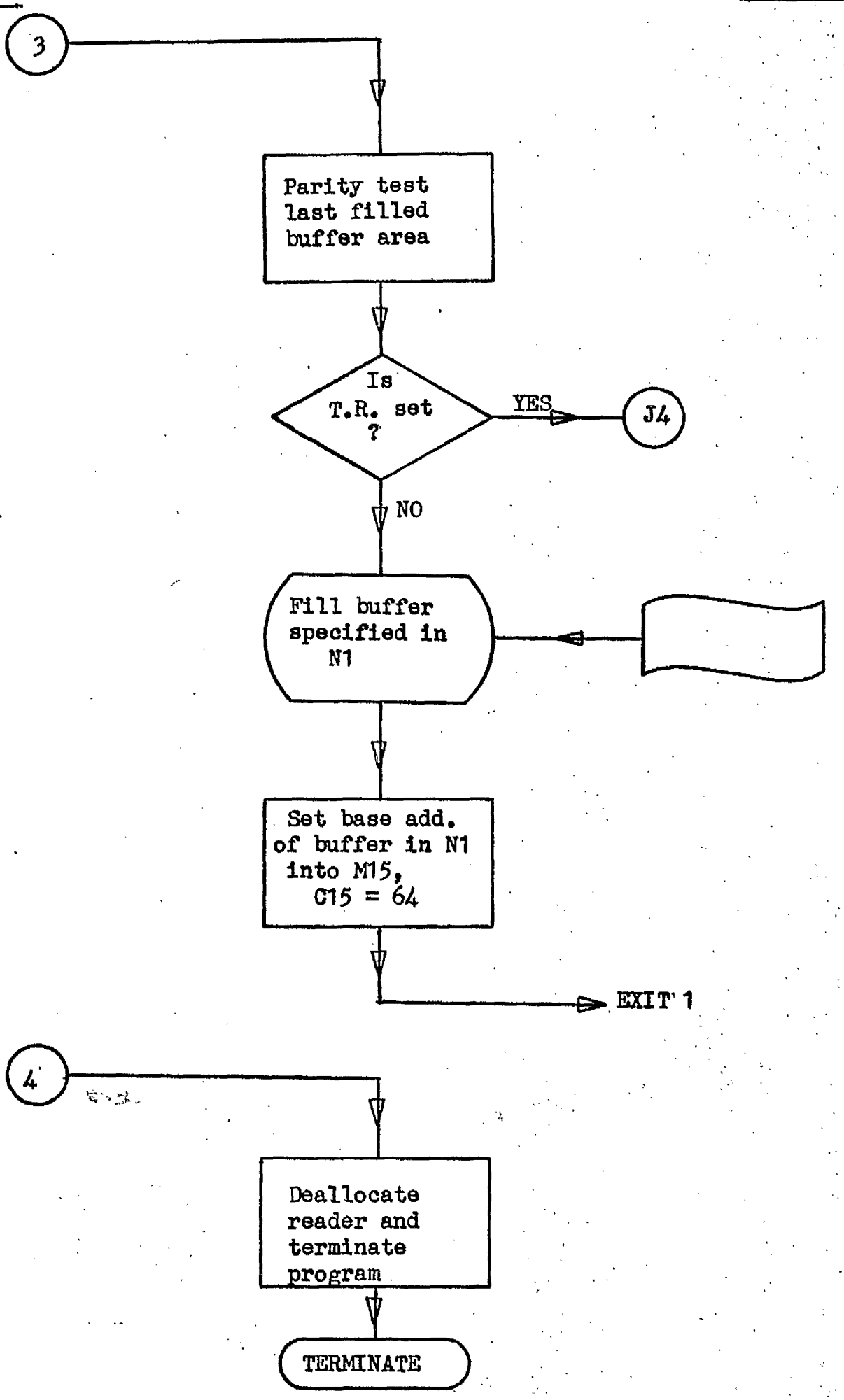

0
Set up
Q11 - 15.
Set base
address of
T.D.T. in N1
JS10L34;
close read file
and rewind
tape
EXIT 1
1
EXIT 2
Deallocate
tape deck
Print fail
message and
number
EXIT 1
EXIT 2


0
Preserve Q10-15
M10 = number of words to be stored
1
Fetch base address of source area
JSP31; write record
EXIT 1
EXIT 2
Update base address (=M11)
Is that last record ?
YES
Reset Q- stores
EXIT 2
NO
J1


0
JSP21; claim reader and fill buffers
5
Store Q15 and set C15 = count of 8
2
Fetch first character
is it a space ?
YES
J1
NO
is it a CRLF ?
YES
J1
NO
is it a C.N. character ?
YES
J1
NO
P2


P2
is it a C.S. character ?
YES
J1
NO
Add character into identifier word
Decrease counter
is it = 0 ?
NO
J2
YES
Fetch identifier word
Reset Q15
EXIT 1
1
Ignore character
J2


0
Initial entry
Set test register off. claim p.t. reader
Set up two input buffer areas in Q-store format
JS2; JS1; fill buffers initially
EXIT 1
1
Is buffer marker set ?
NO
2
YES
Set:-
N1 = 2nd buffer
N2 = 1st buffer
Set:-
N1 = 1st buffer
N2 = 2nd buffer
J3
J3


3
Parity test last filled buffer area
Is T.R. set ?
YES
J4
NO
Fill buffer specified in N1
Set base add. of buffer in N1 into M15, C15 = 64
EXIT 1
4
Deallocate reader and terminate program
TERMINATE

```
P
DDO26AEOOUPU
TEST WORD PROG→
ST 0;
TL 7200;
V38;
W30;
YA12;
YB13;
YC3;
YZ127;
RESTART; J2;
PROG;
      VO=QO/AYA1/AYA12;
      V1=QO/AYC1/AYC3;
      V6/16=P[p]WORD**FORWARDS**********WORD**BACKWARDS**********
            LETTER**HYPHEN**ACCENT**REFERENCE**[c];
      V17/28=P[51s] COUNT***MARKER**MARKER*R1**R2*LINE*WD[7Dc];
      V33/36=P[p]TEXT**ANALYSIS**SOFT**REPORTS[2c];
      V37=QO/AV32/AV36;
      V30=QO/AV6/AV28;
      V38=PDGO10050;

      JSL2;
      SET 12; V38;
      JSP30;  (open write file);
      J2;  (failure exit);
      SET 3; SET 5; OUT; =V5;  (line printer);
      JSP21;
      SET AY1; =RM5;  (Q5=O/1/AY1);
      Q5; =V31;
      ZERO;
      VOP22; =V17P22;  (sets up V17 (=Q1) initially as 6/1/AYA1);
      SET B70; =V32; V37; SET 8; OUT;  (outputs heading for soft
                                        reports on STR. 70);

10;   ERASE; JS1P21;
      DUP; J10=Z; DUP;  =V21P22;  (end of data ch);
      DUP; DUP; SHL6; OR;
      SHL6; OR;
      =V20P22;  (sets up V20P22 as marker for end of data);
      SET 1; SHL24; NOT; NEG;  (set ch. and page at 1);
      =YC1;

3;    JSP22;
      J1;
      J2;

1;    V31; =Q5;
      SET AYA1; =RM4; SET 12; =C4;
```

```
*4;   MOM4Q; =MOM5Q; *J4C4NZS;
      Q5; =V31;
      SET AYA1; JSP31;  (writes record in YA1...YA12);
      J2;  (failure exit);
      C5; NEG; SET 720; J5=;
      ERASE;
      J3;

5;    V30; =Q4; V5; =C4; LPQ4; . (outputs heading);
      SET AY1; =I5; V5; =C5;
      LPQ5;  (outputs buffer of 720 wds);
      SET AY1; =RM5;
      Q5; =V31; JS6;
      ERASE;
      J3;

6;    SET 720; =RC1;
      SET AY1; =M1;

*8;   ZERO; =MOM1Q; *J8C1NZS;
      EXIT 1;

2;    V31; =Q5;
      V30; =Q4; V5; =C4; LPQ4;
      SET AY1; =I5; V5; =C5;
      LPQ5;
      V5; SET 6; OUT;
      JSP32;  (closes write file);
      J4P55;  (failure exit);
      J4P55;
```

```
P
DD026AGOOUPU
CHANGE SORT TAPE→
ST 0;
TL 7200;
V5;
W30;
YA13;
YB13;
RESTART;
PROG;

      V0=POUTPUTMT;   (sorted file);
      V1=PDG010064;   (new file);
      V5=Q0/1/AY14;

      JSL2; SET 12;
      V0; JSP40; J11;  (opens read m.t. file);
      SET 13; V1; JSP30; J11;  (opens write m.t.file);
      =RM1; SET 12; =C1; (Q1=12/1/b.a. of message);
      SET AYA1; =RM2;  (Q2=0/1/AYA1);
      Q2; =V4;
      SET AYB1; =RM3;  (Q3=0/1/AYB1);
      Q3; =V3;
      V5; =Q4;   (Q4=0/1/AY14);

*1;   MOM1Q;  =MOM2Q; *J1C1NZS;  (stores record in YA1-12);
      YA12; =YA13; ZERO; NOT; NEG; =YA12;  (new record size
                  of 13 with 12th word containing frequency
                                                  count);

7;    V3; =Q3; SET 13; =C3;

*14;  ZERO; =MOM3Q; *J14C3NZS;
      JSP41; J10;
      =RM1; V3; =Q3; SET 12; =C1;

*2;   MOM1Q; =MOM3Q; *J2C1NZS;  (stores eles. in YB1-12);
      YB12; =YB13; ZERO; =YB12;
      V4; =Q2; V3; =Q3; SET 9; =C2;

*3;   MOM2Q; MOM3Q; J4≠; ERASE; J3C2NZS;
      YA12; NOT; NEG; =YA12;  (updates counter);
      ZERO; V3; =Q3; SET 9; =C3;

*5;   DUP; =MOM3Q; *J5C3NZS;  (zeroes YB1-9);
      =YB12;
      V3; =Q3; SET 13; =C3;
```

```
*6;   MOM3Q; =MOM4Q; *J6C3NZS;  (stores record away);
      J7;

4;    ERASE; V4; =Q2; SET AY1; =RM5;
      SET 13; =C5;
      YA12; JS2P20; =YA12;  (converts frequency count);

*9;   MOM2Q; =MOM5Q;  (fills Y1-Y13);*J9C5NZS;
      M4; JSP50;  (transfers buffer in Y-stores to m.t.);
      J11; (failure exit);
      V3; =Q3; V4; =Q2; SET 12; =C3;

*8;   MOM3Q; =MOM2Q; *J8C3NZS;
      ZERO; NOT; NEG; =YA12; V5; =Q4; J7;

10;   (failure exits from routines);
      J11=Z;
      V4; =Q2; SET AY1; =RM5;
      YA12; JS2P20; =YA12;
      SET 13; =C5;

*12;  MOM2Q; =MOM5Q; *J12C5NZS;
      M4; JSP50;  (transfers over last buffer);
      J11;  (failure exit);

11;   JSP42; JSP32;  (closes down files);
      J13;

13;   ZERO; OUT;
```

```
P
DD026ALOOUPU+2010052-SSP
NO REFERENCE PROG→
ST 0;
TL 7200;
V2;
W30;
YA13;
RESTART; J4;
PROG;
      V0=PDG010064;   (read file);
      V1=PDG010068;   (write file);
      V2=Q0/1/AYA1;

      JSL2;
      SET 11; V1; JSP30;  (opens write file);
      J3;
      SET 13; V0; JSP40; J3;  (opens read file);
      =RM1; SET 13; =C1;  (Q1=13/1/ba of record);
      V2; =Q2;  (Q2=0/1/AYA1);

*1;   MOM1Q; =MOM2Q; *J1C1NZS;
      YA1; J2=Z;
      YA12; =YA10; YA13; =YA11;
      SET AYA1; JSP31;  (writes record);
      J3;

2;    JSP41; J3;
      =RM1; MOM1; J2=Z;
      SET 13; =C1; V2; =Q2; J1;

3;    ERASE;

4;    JSP32; J5;

5;    JSP42;
      J6;

6;    ZERO; OUT;
```

```
P
DD026AFOOUPU
TEST READ FILE→
ST 0;
TL 7200;
V1;
W30;
YZ128;
RESTART;
PROG;

      V1=Q0/1/AY1;

      JSL2;
      JSP51;  (reads m.t. identifier from paper tape);
      SET 12; JSP40;  (opens read file and sets address of
                                   1st record in N1);
      J6;
      SET 3; SET 5; OUT; =V0;  (claims line printer);
      V1; =Q2; J7;

2;    JSP41;  (reads next message);
      J5;  (failure exit);

7;    =RM1; SET 12; =C1;  (Q1=12/1/b.a. of message);

*1;   MOM1Q; =MOM2Q; *J1C1NZS;
      C2; NEG; SET 1500; J3=; ERASE;
      J2;

3;    ERASE; M-I2; SET AY1; =I2; V0; =C2;  (Q2=dn/ba/ua);
      LPQ2;
      SET 1500; =RC2; SET AY1; =M2;

*4;   ZERO; =MOM2Q; *J4C2NZS;
      V1; =Q2; J2;

5;    J6=Z; J6C2Z;
      SET AY1; =I2; V0; =C2;
      LPQ2;

6;    JSP42; V0; SET 6; OUT;
      J4P55;
```

```
P20V4;  (to carry out character conversion);
     V0=B1212121212121212; (binary equivalent of decimal 10);
     V1=B1717171717171717; (to remove xs 16 bit);
     V2=B2020202020202020; (to insert xs 16 bit);
     V3=B0000000000000020;

1;   (to convert to binary from decimal);
     V0; REV;
     V1; AND; (removes xs 16 bit);
     TOB;
     EXIT 1;


2;   (to convert to decimal from binary);
     V0; REV; FRB;
     V2; OR; (inserts xs 16 bit);
     ZERO; REV; ZERO;

3;   ERASE; REV; NOT; NEG; SET 9; J5=; REV;
     ZERO; SHLD+6; V3; J3=;
     CAB; SET 6; XD; CONT;
     Q15; =V4; NEG; =C15; SHLDC15; ERASE;

6;   V4; =Q15;
     EXIT 1;

5;   ERASE; ERASE; V3; J6;
```

```
P21V3;  (reads next character);

        Q15; =V2; Q14; =V3;  (stores Q14and Q15 away);
        JSP55;  (initializes reader and fills buffer);
        SET 8; =RC14; J5;  (Q14=8/1/0);

1;      JS6; J2C14Z;  (sets up Qstores and jumps if 8 chs. read);
        MOM15;  (fetches input word);

8;      ZERO; SHLD6; DC14;  (isolate next char.);
        REV; =MOM15;  (restore input word);
        J5;

2;      DC15; M+I15;  (updates Q15);
        J4C15Z;  (jumps if buffer empty);

3;      SET 8; =RC14; MOM15; J8;  (resets Q14);

4;      JS1P55; J3;  (refills buffer and jumps back);

5;      Q15; =V0; Q14; =V1;
        V2; =Q15; V3; =Q14;
        EXIT 1;  (stores and resets Q14,15);

6;      Q15; =V2; Q14; =V3;
        V0; =Q15; V1; =Q14;
        EXIT 1;  (stores and resets Q14,15);

7;      JS6; MOM15; REV; SHLD-6;
        ERASE; =MOM15;
        ZERO; NOT; NEG; =+C14;
        J5;  (replaces char. and increases C14);
```

```
P22V65;  (sets up YA1-12 for new word or punct.);

     V20=0;  (end of data word);
     V21=0;  (end of data ch);
     V0=Q6/1/AYA1;
     V7/10= P INVALID*CHARACTER*FOUND*IN*WORD[c];
     V63=Q0/AV6/AV10;
     V12/13=P LONG*WORD*FOUND[c];
     V64=Q0/AV11/AV13;
     V27/28=PAT*REFERENCE[dddc];
     V65=Q0/AV26/AV28;
     V30U=AR6;        V30L=AR14;
     V31U=AR20;       V31L=AR14;
     V32U=AR14;       V32L=AR14;
     V33U=AR9;        V33L=AR10;
     V34U=AR14;       V34L=AR14;
     V35U=AR14;       V35L=AR14;
     V36U=AR14;       V36L=AR14;
     V37U=AR14;       V37L=AR8;
     V38U=AR8;        V38L=AR14;
     V39U=AR14;       V39L=AR34;
     V40U=AR34;       V40L=AR14;
     V41U=AR14;       V41L=AR14;
     V42U=AR8;        V42L=AR8;
     V43U=AR7;        V43L=AR8;
     V44U=AR12;       V44L=AR18;
     V45U=AR19;       V45L=AR12;
     V46U=AR14;       V46L=AR32;
     V47U=AR32;       V47L=AR32;
     V48U=AR32;       V48L=AR32;
     V49U=AR32;       V49L=AR32;
     V50U=AR32;       V50L=AR32;
     V51U=AR32;       V51L=AR32;
     V52U=AR32;       V52L=AR32;
     V53U=AR32;       V53L=AR32;
     V54U=AR32;       V54L=AR32;
     V55U=AR32;       V55L=AR32;
     V56U=AR32;       V56L=AR32;
     V57U=AR32;       V57L=AR32;
     V58U=AR32;       V58L=AR32;
     V59U=AR32;       V59L=AR14;
     V60U=AR14;       V60L=AR14;
     V61U=AR14;       V61L=AR14;

     JS21;
     SET AV30; =RM4;  (Q4=0/1/AV30);
     ZERO; DUP; DUPD; DUPD; =V1; =V3; =V5; =V29; =YA8; =YA9;

1;   JS1P21;  (fetches next character);
```

```
3;    DUP; V21;
      J31=; =M5;
      M4M5H; SHL-24;
      =LINK; EXIT;

32;   (letters);
      V3; NOT; NEG; =V3;  (adds 1 to letter count);
      V4; J4=Z;  (tests between marker);
      V1; SHL6; OR; DUP;  (adds in letter);
      J2<Z;  (jumps if output word full);
      =V1; J1;  (stores word back and jumps back);

31;   (possibly end of data);
      ERASE;
      SHL6; JS1P21; OR; SHL6; JS1P21; OR;
      V20; J17=; ZERO; REV; SHLD-6;
      REV; SHL-42; JS7P21; ZERO;
      REV; SHLD-6; REV; SHL-42; JS7P21; J32;  (with 1st letter
                                                 in N1);

2;    =MOM1Q;  (stores in word buffer);
      ZERO; =V1; J1;  (resets output word and jumps back);

4;    V0; DUP; =Q1;  (Q1=6/1/AYA1);
      REV; SET 12; =C1;  (Q1=12/1/AYA1);
      YA9; =YA0;  (preserves accent marker word);

*5;   ZERO; =MOM1Q; *J5C1NZS;  (sets up clear buffer area);
      ZERO; =V5;  (sets accent marker off);
      SET 1; =V4;  (sets between marker off);
      V1; OR; =V1; =Q1;
      YA0; =YA9; J1;  (stores 1st ch. of new word away);

6;    (space ch. found);
      V4; J7=Z;  (jumps if between marker set);
      J12;  (jumps to treat as terminator);

7;    ERASE; J1;  (ignores space char.);

9;    SET 1; =V2; ERASE; J1;  (sets case marker to shift);

10;   ZERO; =V2; ERASE; J1;  (sets case marker to normal);

12;   V3; J15=Z;  (jumps if word already output);
```

```
15;  DUP; JS2P20; =YA1;
     ZERO; DUP; =YA2; =YA3;
     V2; NEG; SET 7; +; SHL6; OR;  (puts correct case before
                                                punctuation);
     SHL6; SET 7; OR;  (sets case to normal after punctuation);
     =YA4; ZERO; DUP; DUP; =YA5; =YA6; =YA7;
     JS40;
     EXIT 1;

16;  SET AYA0; =RI2; MOM1; =YB8; JS30;
     SET B70; =YA0;
     M1TDQ2; SET 1; =+M2; SET 2; =MOM2;
     (Q2=0/AYA0/AYA...);
     SET B70; =V11; V64; SET 8; OUT;  (prints message);
     Q2; SET 8; OUT;  (prints long word);
     YB8; =MOM1;
     JSP24;
     JS25; ZERO; DUP; =V5; =V4;  (reverses word);
     YC2; NOT; NEG; =YC2;  (updates word count);
     JS40;
     EXIT 1;

18;  (found reference warning ch);
     V2; J14≠Z;
     ERASE; JSP25;  (reads no. and alters references);

13;  JS1P21; SET 2; J11=; DUP; J11=Z; J3;

11;  ERASE; J13;

19;  V2; J14≠Z;  (rejects asterisk-might alter this to be page
                                              warning ch);
     (hyphen found);
     ERASE;  (rejects hyphen sign);

22;  JS1P21;  (fetches next char);
     DUP; J23=Z;  (jumps if space ch);
     SET 2; J24=;  (jumps if a CRLF);
     V3; JS2P20; YA8; SHL24; OR; =YA8;  (allows up to two
                                          hyphens in a word);
     J3;

23;  ERASE; J22;  (ignores space ch);

24;  V3; JS2P20; SET B36; SHL12; OR; YA8; SHL24;
     OR; =YA8;  (stores minus  count);
     J26;
```

```
27;  V1; =MOM1;  (stores away rest of word);
     JS7P21;  (restores punct in input buffer);
     C1; SET 4; -; J16<Z;  (jumps if word has >24 char.);
     JSP23;  (reverses word);
     JS30; YC2; NOT; NEG; =YC2;
     ZERO; =V4;  (sets between marker);
     JS25;
     JS40;
     EXIT 1;

30;  V3;

33;  SET 8; -; DUP; J33≥Z; SET 8; +; SET 8; REV; -; SET 6;
     ×D; CONT; =RC14; MOM1; SHLC14; =MOM1;
     EXIT 1;

14;  (invalid character);
     SET B70; =V6; V63; SET 8; OUT;  (prints soft report);
     V1; =MOM1;  (stores rest of word so far);
     M1TOQ2; SET 1; =+M2; SET 2; =MOM2;
     SET AYA0; =I2;  (Q2=0/AYA0/AYA...);
     SET B70; =YA0; Q2; SET 8; OUT;  (prints word so far found);
     ZERO; =MOM2;  (removes CRLF inserted);
     SET B70; =V26; V65; SET 8; OUT;  (prints AT REFERENCE);
     YC1; JS41; REV;
     ERASE;
     YC2; JS41; =YB2; ERASE; =YB1;
     SET AYB1; DUP; NEG; NOT; =RI2; NOT; NEG;
     NOT; NEG; =M2; SET 2; =MOM2;
     SET B70; =YB0; Q2; SET 8; OUT;  (prints reference);
     ERASE; J1;  (ignores invalid ch. and jumps back);

17;  (end of data);
     ERASE; V3; J29=Z; V21;  (sets end of data ch);
     JS27;
     JS40;
     EXIT 1;  (exit with word still to be stored away);

29;  JS25;
     EXIT 2;  (end of data exit);
```

```
20;  (CRLF found);
     V3; J27≠Z;  (jumps if CRLF is word terminator);

26;  YC2; SHL-24; NOT; NEG;
     SHL24; =YC2;  (1:=1+1,W:=0);
     ERASE; J1;

40;  YC1; JS41; =YA10; ERASE;
     YC2; JS41; =YA11; SET 2; =YA12; ERASE;
     V3; JS2P20; =YA7;
     EXIT 1;

41;  ZERO; SHLD24; JS2P20; REV; SHL-24;
     JS2P20; SHL24; REV; SHLD24; EXIT 1;

21;  Q1; =V14; Q2; =V15; Q3; =V16; Q4; =V22; Q5; =V24;
     V17; =Q1; V18; =Q2; V19; =Q3; V23; =Q4; V25; =Q5;
     EXIT 1;

25;  Q1; =V17; Q2; =V18; Q3; =V19; Q4; =V23; Q5; =V25;
     V14; =Q1; V15; =Q2; V16; =Q3; V22; =Q4; V24; =Q5;
     EXIT 1;

8;   V2; J14=Z; J12;  (jumps to 14 if digit found);

34;  V2; J14=Z;  (ignores digits);
     (accent on next ch);
     ERASE; V29; J35≠Z;
     SET 1; =V29;  (accent found in this word marker);
     V3; NOT; NEG; JS2P20; =YA9; J1;

35;  V3; NOT; NEG; JS2P20; YA9; SHL24;
     OR; =YA9; J1;
```

```
P23V7;

      V4=Q8/1/AYA4;
      V5=Q0/6/6;
      V6=Q3/1/AYA1;
      V7=Q0/1/AYB1;

      Q1; =V0; Q2; =V1; Q3; =V2; Q4; =V3;  (stores away Q stores);
      V6; =Q2; V7; =Q3;

*2;   MOM2Q; =MOM3Q; *J2C2NZS;
      V5; =Q3;  (Q3=0/6/6);
      V4; =Q2;  (Q2=8/1/AYA4);
      V3P22; =C3;  (Q3=no. of letters/6/6);

1;    ZERO; MOM1; SHLD-6;  (isolates next character);
      REV; DUP; J4=Z;
      REV; =MOM1;  (restores input word);

5;    C2; SET 8; J3=;  (reduces counter and jumps if first time);
      ERASE;
      M3; NEG; =RC4; M+I3;  (sets C4 and updates M3);
      SHLC4;  (shifts ch. into correct position);
      J8;

3;    ERASE;

8;    MOM2; OR; =MOM2; DC2;  (stores ch. away and decreases C2);
      DC3;
      J7C3Z;  (checks if word finished);
      J6C2Z; J1;  (checks if output word is empty);

4;    ERASE; ERASE; M-I1; J1;  (alters input word and jumps back);

6;    M+I2; SET 8; =C2;
      SET 6; =M3; J1;

7;    V6; =Q2; V7; =Q3;

*9;   MOM3Q; =MOM2Q; *J9C2NZS;
      V0; =Q1; V1; =Q2; V2; =Q3; V3; =Q4;  (resets Q stores);
      EXIT 1;  (exits normal);
```

```
P24V6;  (deals with reversing of a word with >24 characters);
      V0=Q0/1/AYB1;
      V1=Q0/1/AYA1;
      V2=Q0/1/AYB8;

      Q13; =V3; Q14; =V4; Q15; =V5;
      V0; =Q14; V1; =Q15; V2; =Q13;
      M1;
      SET AYA0; -; =C15;  (Q15=no. of store wds./1/AYA1);

1;  MOM15Q; DUP; =MOM14Q; =MOM13Q; J1C15NZ;  (stores away word
                                              in YB1... and YB8..);
      M-I14; M14TOQ1; ZERO; DUP; DUP; =YA4; =YA5; =YA6;
      YA7; =YB7;  (preserves copy of letter count);
      JSP23;  (reverses word into YA4...);
      YB7; =YA7;  (restores no. count);
      V2; =Q13; SET 3; =C13;
      V1; =Q15;

*2;  MOM13Q; =MOM15Q; *J2C13NZS;
      V3; =Q13; V4; =Q14; V5; =Q15;
      EXIT 1;  (exits normal);
```

```
P25V13;  (subroutine to read reference number);

      V1=Q2/1/AV2;
      V13=Q0/AV6/AV11;
      V7/11=PINVALID*CHARACTER*IN*LINE*REFERENCE[4dc];
      V12=B1212121212121212;


      Q15; =V5;  (stores contents of Q15);
      V1; =Q15; SET 3; =C15;   (Q15=3/1/AV2);
      ZERO; DUP; DUP; =V2; =V3; =V4;

4;    ZERO; =V0;

1;    JS1P21;  (fetches next character);
      SET B37; J2=;  (tests for separator);
      DUP; J2=Z; SET 2; J2=;  (reference no. terminator
                                          = space or CRLF);
      DUP; SET B20; -; J3<Z;
      DUP; SET B31; -; J3>Z;  (jumps if not a digit);
      SET B17; AND; V0;  (removes xs 16 bit);
      SET 10; XD; CONT; +; =V0; J1;  (adds in no.);

2;    ERASE; V0; =MOM15Q; J4C15NZ;  (stores no. away);
      V2; V12; DUP; CAB; TOB; SHL24;
      V3; CAB; DUP; CAB; TOB; CAB;
      OR; =YC1;
      V4; TOB; SHL24; =YC2;
      V5; =Q15;
      EXIT 1;

3;    (invalid character in line reference);
      SET B70; =V6; V13; SET 8; OUT;  (prints soft report of i.c.);
      ERASE; J1;  (ignores i.c.and jumps back);
```

```
P30V600;  (this procedure opens a mag. tape file);
          (enter with N1=m.t. identifier
                      N2=message length in words);
      V0=0;
      V1=Q0/1/0;
      V2=0;
      V3=Q0/0/256;
      V4=0;  (to hold tape deck no.);
      V5=Q0/AV276/AV20;  (b.a. of buffers);
      V6=0;
      V7=Q1/1/0;
      V8=0;
      V9=0;  (to hold message length in words);
      V12=PL37ERROR;

      DUP; =V0; REV; =V9;  (message length);
      SET 4; OUT; SHL16; =V4;  (claims tape d.n.);
      SET AV0; JS12L37; J1;
      Q12; =V597; Q13; =V598; Q14; =V599; Q15; =V600;
                              (preserves Q stores);
      EXIT 2;  (normal exit);

1;    V12; SET 1; JS2L89; JS1L89;  (prints failure message);
      EXIT 1;
```

```
P31V0;  (write record);
        (N1 contains b.a. of source area);

      V597P30; =Q12;
      V598P30; =Q13;
      V599P30; =Q14;
      V600P30; =Q15;
      SET AV0P30; JSL37; J1;
      Q12; =V597P30;
      Q13; =V598P30;
      Q14; =V599P30;
      Q15; =V600P30;
      EXIT 2;  (normal exit);

1;    V12P30; SET 1; JS2L89; JS1L89;
      EXIT 1;  (failure exit);



P32V0;  (close write file);

      V597P30; =Q12;
      V598P30; =Q13;
      V599P30; =Q14;
      V600P30; =Q15;
      SET AV0P30; JS10L37; J1;
      V4P30; SHL16; SHL-32; SET 6; OUT;  (deallocates deck);
      EXIT 2;

1;    V12P30; SET 1; JS2L89; JS1L89;
      EXIT 1;
```

```
P40V550;   (open read file);
           (enter with m.t. identifier in N1
            and with message length in N2);
V0=0;
V1=Q0/1/0;
V2=0;
V3=Q0/4096/256;
V4=0;
V5=Q0/AV276/AV20;   (b.a. of buffers);
V6=0;
V7=PL34 ERROR;

      DUP; =V0; REV; =V6;
      SET 4; OUT; SHL16; =V4;   (claims tape d.n.);
      SET AV0; JS12L34; J1; DUP; =M11;
      Q11; =V12; Q12; =V8; Q13; =V9; Q14; =V10; Q15; =V11;
                                  (preserves Qstores);
      EXIT 2;   (exits with address of 1st record in N1);

1;    V7; SET 1; JS2L89; JS1L89;   (prints failure message);
      EXIT 1;   (failure exit);
```

```
P42V0;  (close read file);
      V12P40; =Q11; V8P40; =Q12; V9P40; =Q13; V10P40; =Q14; V11P40;
      =Q15;
      SET AV0P40; JS10L34;
      J1; V4P40; SHL16; SHL-32; SET 6; OUT;
      EXIT 1;

1;    V740; SET 1; JS2L89; JS1L89; EXIT 1;
```

```
P41V0;  (read message);
      V12P40; =Q11;
      V8P40; =Q12;
      V9P40; =Q13;
      V10P40; =Q14;
      V11P40; =Q15;

      M11; SET AV0P40; JSL34; J1;
      DUP; =M11;
      Q11; =V12P40; Q12; =V8P40; Q13; =V9P40; Q14; =V10P40;
      Q15; =V11P40;
      EXIT 2;  (normal exit with add. of next message in N1);

1;    SET 2; J2=; V7P40; SET 1; JS2L89; JS1L89;
      ZERO;

2;    EXIT 1;  (failure exit);
```

```
P50V9;  (to store on m.t. what is in Y1 onwards
          -upper address being in N1 -multiple of 13);
      V3=Q0/13/AY1;
      Q15; =V0; Q14; =V1; Q12; =V6; Q13; =V7; Q10; =V8; Q11; =V9;
      SET AY1; -; =RM10; SET 13; =I10;
      V3; =Q11;  (0/13/AY1);

1;    M11; JSP31;   (writes 13 word record);
      J2;
      M+I11;  (Q11=0/13/AY(1+x×13));
      M-I10; M10;
      J1≠Z;
      V0; =Q15; V1; =Q14; V6; =Q12; V7; =Q13; V8; =Q10; V9; =Q11;
      EXIT 2;

2;    EXIT 1;  (failure exit);
```

```
P51V1;  (reads m.t. identifier as input on p.t.);

        JSP21;  (claims reader and fills buffer);
        Q15; =V1;
        SET 8; =RC15;

2;      JS1P21; DUP; J1=Z; SET 2; J1=; SET 7; J1=;
        SET 6; J1=;
        V0; SHL6; OR; =V0; DC15; J2C15NZ; V0;
        V1; =Q15;
        EXIT1;  (with identifier in N1);

1;      ERASE; J2;
```

```
P55V8;  (initializes reader and fills buffer);

        V0=Q0/AYZ0/AYZ63;
        V1=Q0/AYZ64/AYZ127;
        V2=0;
        V3=0;
        V4/6=PPARITY*FAIL*ON*INPUT[3dc];
        V7=Q0/AV3/AV6;

        ZERO; =TR;
        SET 2; SET 5; OUT;  (reader);
        SHL+32; DUP; V0; OR; =V0;
        V1; OR; =V1;  (stores d.n. in V stores);
        JS2; JS1;  (initial input);
        EXIT 1;

1;      V2; DUP; NOT; =V2; J2≠Z;  (tests marker);
        V1; V0; J3;

2;      V0; V1;

3;      =Q15; PARQ15; J5TR;  (tests for parit failure);
        PIBQ15;  (fills next input buffer);
        SHL-16; =RM15;
        SET 64; =C15;  (Q15=64/1/buffer b.a. for ch. read);
        EXIT 1;  (load input buffers);

4;      V0; SHL-32; SET 6; OUT;
        ZERO; OUT;  (deallocates device and terminates prog);

5;      V7; SET 8; OUT; J4;
```

APPENDIX 5

Sample Input and Output

THE DAFFODILS

I wandered lonely as a cloud
That floats on high o'er vales and hills,
When all at once I saw a crowd,
A host of golden daffodils,
Beside the lake, beneath the trees,
Fluttering and dancing in the breeze.

Continuous as the stars that shine
And twinkle on the milky way,
They stretched in never-ending line
Along the margin of a bay:
Ten thousand saw I at a glance
Tossing their heads in sprightly dance.

The waves beside them danced, but they
Out-did the sparkling waves in glee:
A poet could not but be gay
In such a jocund company!
I gazed, and gazed, but little thought
What wealth the show to me had brought.

For oft, when on my couch I lie
In vacant or in pensive mood,
They flash upon that inward eye
Which is the bliss of solitude;
And then my heart with pleasure fills,
And dances with the daffodils.

WILLIAM WORDSWORTH (1770 - 1850)

I wandered lonely as a cloud
That floats on high oer vales and hills,
When all at once I saw a crowd,
A host of golden daffodils,
Beside the lake, beneath the trees,
Fluttering and dancing in the breeze.

+1.2.1
Continuous as the stars that shine
And twinkle on the milky way,
They stretched in never-ending line
Along the margin of a bay:
Ten thousand saw I at a glance
Tossing their heads in sprightly dance.

+1.3.1
The waves beside them danced, but they
Out-did the sparkling waves in glee:
A poet could not but be gay
In such a jocund company!
I gazed, and gazed, but little thought
What wealth the show to me had brought.

+1.4.1
For oft, when on my couch I lie
In vacant or in pensive mood,
They flash upon that inward eye
Which is the bliss of solitude;
And then my heart with pleasure fills,
And dances with the daffodils.

LIST 1

WORD FORWARDS	WORD BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	REFERENCE R2	REFERENCE LINE	REFERENCE WD
I	I	1			1	1	1	1
WANDERED	DEREDNAW	8			1	1	1	2
LONELY	YLENOL	6			1	1	1	3
AS	SA	2			1	1	1	4
A	A	1			1	1	1	5
CLOUD	DUOLC	5			1	1	1	6
THAT	TAHT	4			1	1	2	1
FLOATS	STAOLF	6			1	1	2	2
ON	NO	2			1	1	2	3
HIGH	HGIH	4			1	1	2	4
OER	REO	3			1	1	2	5
VALES	SELAV	5			1	1	2	6
AND	DNA	3			1	1	2	7
HILLS	SLLIH	5			1	1	2	8
31	9	0			1	1	2	8
WHEN	NEHW	4			1	1	3	1
ALL	LLA	3			1	1	3	2
AT	TA	2			1	1	3	3
ONCE	ECNO	4			1	1	3	4
I	I	1			1	1	3	5
SAW	WAS	3			1	1	3	6
A	A	1			1	1	3	7
CROWD	DWORC	5			1	1	3	8
31	9	0			1	1	3	8
A	A	1			1	1	4	1
HOST	TSOH	4			1	1	4	2
OF	FO	2			1	1	4	3
GOLDEN	NEDLOG	6			1	1	4	4
DAFFODILS	SLLIDOFFAD	9			1	1	4	5
31	9	0			1	1	4	5
BESIDE	EDISEB	6			1	1	5	1
THE	EHT	3			1	1	5	2
LAKE	EKAL	4			1	1	5	3
31	9	0			1	1	5	3

WORD FORWARDS	WORD BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	R2	LE	WD
BENEATH	HTAENEB	7			1	1	5	4
THE	EHT	3			1	1	5	5
TREES	SEERT	5			1	1	5	6
31		0			1	1	5	6
FLUTTERING	GNIRETTULF	10			1	1	6	1
AND	DNA	3			1	1	6	2
DANCING	GNICNAD	7			1	1	6	3
IN	NI	2			1	1	6	4
THE	EHT	3			1	1	6	5
BREEZE	EZEERB	6			1	1	6	6
31		0			1	1	6	6
CONTINUOUS	SUOUNITNOC	10			1	2	1	1
AS	SA	2			1	2	1	2
THE	EHT	3			1	2	1	3
STARS	SRATS	5			1	2	1	4
THAT	TAHT	4			1	2	1	5
SHINE	ENIHS	5			1	2	1	6
AND	DNA	3			1	2	2	1
TWINKLE	ELKNIWT	7			1	2	2	2
ON	NO	2			1	2	2	3
THE	EHT	3			1	2	2	4
MILKY	YKLIM	5			1	2	2	5
WAY	YAW	3			1	2	2	6
31		0			1	2	2	6
THEY	YEHT	4			1	2	3	1
STRETCHED	DEHCTERTS	9			1	2	3	2
IN	NI	2			1	2	3	3
NEVERENDING	GNI[illegible]GREVEN	11	5		1	2	3	4
LINE	ENIL	4			1	2	3	5
ALONG	GNOLA	5			1	2	4	1
THE	EHT	3			1	2	4	2
MARGIN	NIGRAM	6			1	2	4	3
OF	FO	2			1	2	4	4

WORD	WORD BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	R2	LINE	WD
A	A	1			1	2	4	5
BAY	YAB	3			1	2	4	6
15	:	0			1	2	4	6
TEN	NET	3			1	2	5	1
THOUSAND	DNASUOHT	8			1	2	5	2
SAW	WAS	3			1	2	5	3
I	I	1			1	2	5	4
AT	TA	2			1	2	5	5
A	A	1			1	2	5	6
GLANCE	ECNALG	6			1	2	5	7
TOSSING	GNISSOT	7			1	2	6	1
THEIR	RIEHT	5			1	2	6	2
HEADS	SDAEH	5			1	2	6	3
IN	NI	2			1	2	6	4
SPRIGHTLY	YLTHGIRPS	9			1	2	6	5
DANCE	ECNAD	5			1	2	6	6
31	.	0			1	2	6	6
THE	EHT	3			1	3	1	1
WAVES	SEVAW	5			1	3	1	2
BESIDE	EDISEB	6			1	3	1	3
THEM	MEHT	4			1	3	1	4
DANCED	DECNAD	6			1	3	1	5
31	,	0			1	3	1	5
BUT	TUB	3			1	3	1	6
THEY	YEHT	4			1	3	1	7
OUTDID	DIDTUO	6	3		1	3	2	1
THE	EHT	3			1	3	2	2
SPARKLING	GNILKRAPS	9			1	3	2	3
WAVES	SEVAW	5			1	3	2	4
IN	NI	2			1	3	2	5
GLEE	EELG	4			1	3	2	6
15	:	0			1	3	2	6

WORD FORWARDS	WORD BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	R2	LINE	WD
A	A	1			1	3	3	1
POET	TEOP	4			1	3	3	2
COULD	DLUOC	5			1	3	3	3
NOT	TON	3			1	3	3	4
BUT	TUB	3			1	3	3	5
BE	EB	2			1	3	3	6
GAY	YAG	3			1	3	3	7
IN	NI	2			1	3		1
SUCH	HCUS	4			1	3	4	2
A	A	1			1	3	4	3
JOCUND	DNUCOJ	6			1	3	4	4
COMPANY	YNAPMOC	7			1	3	4	5
16	↑	0			1	3	4	5
I	I	1			1	3	5	1
GAZED	DEZAG	5			1	3	5	2
31	,	0			1	3	5	2
AND	DNA	3			1	3	5	3
GAZED	DEZAG	5			1	3	5	4
31	,	0			1	3	5	4
BUT	TUB	3			1	3	5	5
LITTLE	ELTTIL	6			1	3	5	6
THOUGHT	THGUOHT	7			1	3	5	7
WHAT	TAHW	4			1	3	6	1
WEALTH	HTLAEW	6			1	3	6	2
THE	EHT	3			1	3	6	3
SHOW	WOHS	4			1	3	6	4
TO	OT	2			1	3	6	5
ME	EM	2			1	3	6	6
HAD	DAH	3			1	3	6	7
BROUGHT	THGUORB	7			1	3	6	8
31	.	0			1	3	6	8

WORD FORWARDS	WORD BACKWARDS	LETTER COUNT	HYPHEN MARKER	ACCENT MARKER	REFERENCE R1	R2	LINE	WD
AND	DNA	3			1	4	5	1
THEN	NGHT	4			1	4	5	2
MY	YM	2			1	4	5	3
HEART	TRAEH	5			1	4	5	4
WITH	HTIW	4			1	4	5	5
PLEASURE	ERUSAELP	8			1	4	5	6
FILLS	SLLIF	5			1	4	5	7
31		0			1	4	5	7
AND	DNA	3			1	4	6	1
DANCES	SECNAD	6			1	4	6	2
WITH	HTIW	4			1	4	6	3
THE	EHT	3			1	4	6	4
DAFFODILS	SLIDOFFAD	9			1	4	6	5
31		0			1	4	6	5

LIST 2

15	:	0	1	2	4	6
15	:	0	1	3	2	6
16	↑	0	1	3	4	5
28	;	0	1	4	4	6
31	,	0	1	1	2	8
31	,	0	1	1	3	8
31	,	0	1	1	4	5
31	,	0	1	1	5	3
31	,	0	1	1	5	6
31	,	0	1	2	2	6
31	,	0	1	3	1	5
31	,	0	1	3	5	2
31	,	0	1	3	5	4
31	,	0	1	4	1	2
31	,	0	1	4	2	6
31	,	0	1	4	5	7
31	.	0	1	1	6	6
31	.	0	1	2	6	6
31	.	0	1	3	6	8
31	.	0	1	4	6	5
A	A	1	1	1	1	5
A	A	1	1	1	3	7
A	A	1	1	1	4	1
A	A	1	1	2	4	5
A	A	1	1	2	5	6
A	A	1	1	3	3	1
A	A	1	1	3	4	3
ALL	LLA	3	1	1	3	2
ALONG	GNOLA	5	1	2	4	1
AND	DNA	3	1	1	2	7
AND	DNA	3	1	1	6	2
AND	DNA	3	1	2	2	1
AND	DNA	3	1	3	5	3

AND	DNA	3	1	4	5	1
AND	DNA	3	1	4	6	1
AS	SA	2	1	1	1	4
AS	SA	2	1	2	1	2
AT	TA	2	1	1	3	3
AT	TA	2	1	2	5	5
BAY	YAB	3	1	2	4	6
BE	EB	2	1	3	3	6
BENEATH	HTAENEB	7	1	1	5	4
BESIDE	EDISEB	6	1	1	5	1
BESIDE	EDISEB	6	1	3	1	3
BLISS	SSILB	5	1	4	4	4
BREEZE	EZEERB	6	1	1	6	6
BROUGHT	THGUORB	7	1	3	6	8
BUT	TUB	3	1	3	1	6
BUT	TUB	3	1	3	3	5
BUT	TUB	3	1	3	5	5
CLOUD	DUOLC	5	1	1	1	6
COMPANY	YNAPMOC	7	1	3	4	5
CONTINUOUS	SUOUNITNOC	10	1	2	1	1
COUCH	HCUOC	5	1	4	1	6
COULD	DLUOC	5	1	3	3	3
CROWD	DWORC	5	1	1	3	8
DAFFODILS	SLIDOFFAD	9	1	1	4	5
DAFFODILS	SLIDOFFAD	9	1	4	6	5
DANCE	ECNAD	5	1	2	6	6
DANCED	DECNAD	6	1	3	1	5
DANCES	SECNAD	6	1	4	6	2
DANCING	GNICNAD	7	1	1	6	3
EYE	EYE	3	1	4	3	6
FILLS	SLLIF	5	1	4	5	7
FLASH	HSALF	5	1	4	3	2
FLOATS	STAOLF	6	1	1	2	2

FLUTTERING	GNIRETTULF	10	1	1	6	1
FOR	ROF	3	1	4	1	1
GAY	YAG	3	1	3	3	7
GAZED	DEZAG	5	1	3	5	2
GAZED	DEZAG	5	1	3	5	4
GLANCE	ECNALG	6	1	2	5	7
GLEE	EELG	4	1	3	2	6
GOLDEN	NEDLOG	6	1	1	4	4
HAD	DAH	3	1	3	6	7
HEADS	SDAEH	5	1	2	6	3
HEART	TRAEH	5	1	4	5	4
HIGH	HGIH	4	1	1	2	4
HILLS	SLLIH	5	1	1	2	8
HOST	TSOH	4	1	1	4	2
I	I	1	1	1	1	1
I	I	1	1	1	3	5
I	I	1	1	2	5	4
I	I	1	1	3	5	1
I	I	1	1	4	1	7
IN	NI	2	1	1	6	4
IN	NI	2	1	2	3	3
IN	NI	2	1	2	6	4
IN	NI	2	1	3	2	5
IN	NI	2	1	3	4	1
IN	NI	2	1	4	2	1
IN	NI	2	1	4	2	4
INWARD	DRAWNI	6	1	4	3	5
IS	SI	2	1	4	4	2
JOCUND	DNUCOJ	6	1	3	4	4
LAKE	EKAL	4	1	1	5	3
LIE	EIL	3	1	4	1	8
LINE	ENIL	4	1	2	3	5
LITTLE	ELTTIL	6	1	3	5	6

LONELY	YLJENOL	6		1	1	1	3
MARGIN	NIGRAM	6		1	2	4	3
ME	EM	2		1	3	6	6
MILKY	YKLIM	5		1	2	2	5
MOOD	DOOM	4		1	4	2	6
MY	YM	2		1	4	1	5
MY	YM	2		1	4	5	3
NEVERENDING	GNIDNEREVEN	11	5	1	2	3	4
NOT	TON	3		1	3	3	4
OER	REO	3		1	1	2	5
OF	FO	2		1	1	4	3
OF	FO	2		1	2	4	4
OF	FO	2		1	4	4	5
OFT	TFO	3		1	4	1	2
ON	NO	2		1	1	2	3
ON	NO	2		1	2	2	3
ON	NO	2		1	4	1	4
ONCE	ECNO	4		1	1	3	4
OR	RO	2		1	4	2	3
OUTDID	DIDTUO	6	3	1	3	2	1
PENSIVE	EVISNEP	7		1	4	2	5
PLEASURE	ERUSAELP	8		1	4	5	6
POET	TEOP	4		1	3	3	2
SAW	WAS	3		1	1	3	6
SAW	WAS	3		1	2	5	3
SHINE	ENIHS	5		1	2	1	6
SHOW	WOHS	4		1	3	6	4
SOLITUDE	EDUTILOS	8		1	4	4	6
SPARKLING	GNILKRAPS	9		1	3	2	3
SPRIGHTLY	YLTHGIRPS	9		1	2	6	5
STARS	SRATS	5		1	2	1	4
STRETCHED	DEHCTERTS	9		1	2	3	2
SUCH	HCUS	4		1	3	4	2

TEN	NET	3	1	2	5	1
THAT	TAHT	4	1	1	2	1
THAT	TAHT	4	1	2	1	5
THAT	TAHT	4	1	4	3	4
THE	EHT	3	1	1	5	2
THE	EHT	3	1	1	5	5
THE	EHT	3	1	1	6	5
THE	EHT	3	1	2	1	3
THE	EHT	3	1	2	2	4
THE	EHT	3	1	2	4	2
THE	EHT	3	1	3	1	1
THE	EHT	3	1	3	2	2
THE	EHT	3	1	3	6	3
THE	EHT	3	1	4	4	3
THE	EHT	3	1	4	6	4
THEIR	RIEHT	5	1	2	6	2
THEM	MEHT	4	1	3	1	4
THEN	NEHT	4	1	4	5	2
THEY	YEHT	4	1	2	3	1
THEY	YEHT	4	1	3	1	7
THEY	YEHT	4	1	4	3	1
THOUGHT	THGUOHT	7	1	3	5	7
THOUSAND	DNASUOHT	8	1	2	5	2
TO	OT	2	1	3	6	5
TOSSING	GNISSOT	7	1	2	6	1
TREES	SEERT	5	1	1	5	6
TWINKLE	ELKNIWT	7	1	2	2	2
UPON	NOPU	4	1	4	3	3
VACANT	TNACAV	6	1	4	2	2
VALES	SELAV	5	1	1	2	6
WANDERED	DEREDNAW	8	1	1	1	2
WAVES	SEVAW	5	1	3	1	2
WAVES	SEVAW	5	1	3	2	4

WAY	YAW	3	1	2	2	6
WEALTH	HTLAEW	6	1	3	6	2
WHAT	TAHW	4	1	3	6	1
WHEN	NEHW	4	1	1	3	1
WHEN	NEHW	4	1	4	1	3
WHICH	HCIHW	5	1	4	4	1
WITH	HTIW	4	1	4	5	5
WITH	HTIW	4	1	4	6	3

LIST 3

15	:	0	1	2	4	6	2
			1	3	2	6	
16	↑	0	1	3	4	5	1
.28	;	0	1	4	4	6	1
31	,	0	1	1	2	8	12
			1	1	3	8	
			1	1	4	5	
			1	1	5	3	
			1	1	5	6	
			1	2	2	6	
			1	3	1	5	
			1	3	5	2	
			1	3	5	4	
			1	4	1	2	
			1	4	2	6	
			1	4	5	7	
31	.	0	1	1	6	6	4
			1	2	6	6	
			1	3	6	8	
			1	4	6	5	
A	A	1	1	1	1	5	7
			1	1	3	7	
			1	1	4	1	
			1	2	4	5	
			1	2	5	6	
			1	3	3	1	
			1	3	4	3	
ALL	LLA	3	1	1	3	2	1
AL[illegible]	GNOLA	5	1	2	4	1	1
AND	DNA	3	1	1	2	7	6
			1	1	6	2	
			1	2	2	1	
			1	3	5	3	
			1	4	5	1	
			1	4	6	1	

AS	SA	2	1	1	1	4	2
			1	2	1	2	
AT	TA	2	1	1	3	3	2
			1	2	5	5	
BAY	YAB	3	1	2	4	6	1
BE	EB	2	1	3	3	6	1
BENEATH	HTAENEB	7	1	1	5	4	1
BESIDE	EDISEB	6	1	1	5	1	2
			1	3	1	3	
BLISS	SSILB	5	1	4	4	4	1
BREEZE	EZEERB	6	1	1	6	6	1
BROUGHT	THGUORB	7	1	3	6	8	1
BUT	TUB	3	1	3	1	6	3
			1	3	3	5	
			1	3	5	5	
CLOUD	DUOLC	5	1	1	1	6	1
COMPANY	YNAPMOC	7	1	3	4	5	1
CONTINUOUS	SUOUNITNOC	10	1	2	1	1	1
COUCH	HCUOC	5	1	4	1	6	1
COULD	DLUOC	5	1	3	3	3	1
CROWD	DWORC	5	1	1	3	8	1
DAFFODILS	SLIDOFFAD	9	1	1	4	5	2
			1	4	6	5	
DANCE	ECNAD	5	1	2	6	6	1
DANCED	DECNAD	6	1	3	1	5	1
DANCES	SECNAD	6	1	4	6	2	1
DANCING	GNICNAD	7	1	1	6	3	1
EYE	EYE	3	1	4	3	6	1
FILLS	SLLIF	5	1	4	5	7	1
FLASH	HSALF	5	1	4	3	2	1
FLOATS	STAOLF	6	1	1	2	2	1
FLUTTERING	GNIRETTUOLF	10	1	1	6	1	1
FOR	ROF	3	1	4	1	1	1

GAY	YAG	3	1	3	3	7	1
GAZED	DEZAG	5	1	3	5	2	2
			1	3	5	4	
GLANCE	ECNALG	6	1	2	5	7	1
GLEE	EELG	4	1	3	2	6	1
GOLDEN	NEDLOG	6	1	1	4	4	1
HAD	DAH	3	1	3	6	7	1
HEADS	SDAEH	5	1	2	6	3	1
HEART	TRAEH	5	1	4	5	4	1
HIGH	HGIH	4	1	1	2	4	1
HILLS	SLLIH	5	1	1	2	8	1
HOST	TSOH	4	1	1	4	2	1
I	I	1	1	1	1	1	5
			1	1	3	5	
			1	2	5	4	
			1	3	5	1	
			1	4	1	7	
IN	NI	2	1	1	6	4	7
			1	2	3	3	
			1	2	6	4	
			1	3	2	5	
			1	3	4	1	
			1	4	2	1	
			1	4	2	4	
INWARD	DRAWNI	6	1	4	3	5	1
IS	SI	2	1	4	4	2	1
JOCUND	DNUCOJ	6	1	3	4	4	1
LAKE	EKAL	4	1	1	5	3	1
LIE	EIL	3	1	4	1	8	1
LONE	ENOL	4	1	2	3	5	1
LOVELY	YLEVOL	6	1	3	5	6	1
LOVELY	YLEVOL	6	1	1	1	3	1
MARGIN	NIGRAM	6	1	2	4	3	1

ME	EM	2		1	3	6	6	1
MILKY	YKLIM	5		1	2	2	5	1
MOOD	DOOM	4		1	4	2	6	1
MY	YM	2		1	4	1	5	2
				1	4	5	3	
NEVERENDING	GNIDNEREVEN	11	5	1	2	3	4	1
NOT	TON	3		1	3	3	4	1
OER	REO	3		1	1	2	5	1
OF	FO	2		1	1	4	3	3
				1	2	4	4	
				1	4	4	5	
OFT	TFO	3		1	4	1	2	1
ON	NO	2		1	1	2	3	3
				1	2	2	3	
				1	4	1	4	
ONCE	ECNO	4		1	1	3	4	1
OR	RO	2		1	4	2	3	1
OUTDID	DIDTUO	6	3	1	3	2	1	1
PENSIVE	EVISNEP	7		1	4	2	5	1
PLEASURE	ERUSAELP	8		1	4	5	6	1
POOP	TEOP	4		1	3	3	2	1
RAW	WAS	3		1	1	3	6	2
				1	2	5	3	
SHINE	ENIHS	5		1	2	1	6	1
SHOW	WOHS	4		1	3	6	4	1
SOLITUDE	EDUTILOS	8		1	4	4	6	1
SPARKLING	GNILKRAPS	9		1	3	2	3	1
SPECTRUM	MURTCEPS	9		1	2	6	5	1
SPARS	SRAPS	5		1	2	1	4	1
SURROUND	DNUORRUS	9		1	2	3	2	1
SUCH	HCUS	4		1	3	4	2	1
TEN	NET	3		1	2	5	1	1

THAT	TAHT	4	1	1	2	1	3
			1	2	1	5	
			1	4	3	4	
THE	EHT	3	1	1	5	2	11
			1	1	5	5	
			1	1	6	5	
			1	2	1	3	
			1	2	2	4	
			1	2	4	2	
			1	3	1	1	
			1	3	2	2	
			1	3	6	3	
			1	4	4	3	
			1	4	6	4	
THEIR	RIEHT	5	1	2	6	2	1
THEM	MEHT	4	1	3	1	4	1
THEN	NEHT	4	1	4	5	2	1
THEY	YEHT	4	1	2	3	1	3
			1	3	1	7	
			1	4	3	1	
THOUGHT	THGUOHT	7	1	3	5	7	1
THOUSAND	DNASUOHT	8	1	2	5	2	1
TO	OT	2	1	3	6	5	1
TOWARDS	SDRAWOT	7	1	2	6	1	1
TREES	SEERT	5	1	1	5	6	1
TWELVE	EVLEWT	7	1	2	2	2	1
UPON	NOPU	4	1	4	3	3	1
VACANT	TNACAV	6	1	4	2	2	1
VASES	SESAV	5	1	1	2	6	1
VAULTED	DETLUAV	8	1	1	1	2	1
VAVES	SEVAV	5	1	3	1	2	2
			1	3	2	4	

WAY	YAW	3	1	2	2	6	1
WEALTH	HTLAEW	6	1	3	6	2	1
WHAT	TAHW	4	1	3	6	1	1
WHEN	NEHW	4	1	1	3	1	2
			1	4	1	3	
WHICH	HCIHW	5	1	4	4	1	1
WITH	HTIW	4	1	4	5	5	2
			1	4	6	3	

LIST 4

15	:	0	2
16	↑	0	1
28	;	0	1
31	'	0	12
31	.	0	4
A	A	1	7
ALL	LLA	3	1
ALONG	GNOLA	5	1
AND	DNA	3	6
AS	SA	2	2
AT	TA	2	2
BAY	YAB	3	1
BE	EB	2	1
BENEATH	HTAENEB	7	1
BESIDE	EDISEB	6	2
BLISS	SSILB	5	1
BREEZE	EZEERB	6	1
BROUGHT	THGUORB	7	1
BUT	TUB	3	3
CLOUD	DUOLC	5	1
COMPANY	YNAPMOC	7	1
CONTINUOUS	SUOUNITNOC	10	1
COUCH	HCUOC	5	1
COULD	DLUOC	5	1
CROWD	DWORC	5	1
DAFFODILS	SLIDOFFAD	9	2
DANCE	ECNAD	5	1
DANCED	DECNAD	6	1

DANCES	SECNAD	6	1
DANCING	GNICNAD	7	1
EYE	EYE	3	1
FILLS	SLLIF	5	1
FLASH	HSALF	5	1
FLOATS	STAOLF	6	1
FLUTTERING	GNIRETTULF	10	1
FOR	ROF	3	1
GAY	YAG	3	1
GAZED	DEZAG	5	2
GLANCE	ECNALG	6	1
GLEE	EELG	4	1
GOLDEN	NEDLOG	6	1
HAD	DAH	3	1
HEADS	SDAEH	5	1
HEART	TRAEH	5	1
HIGH	HGIH	4	1
HILLS	SLLIH	5	1
HOST	TSOH	4	1
I	I	1	5
IN	NI	2	7
INWARD	DRAWNI	6	1
IS	SI	2	1
JOCUND	DNUCOJ	6	1
LAKE	EKAL	4	1
LIE	EIL	3	1
LINE	ENIL	4	1
LITTLE	ELTTIL	6	1
LONELY	YLENOL	6	1
MARGIN	NIGRAM	6	1
ME	EM	2	1
MILKY	YKLIM	5	1

MOOD	DOOM	4		1
MY	YM	2		2
NEVERENDING	GNIDNEREVEN	11	5	1
NOT	TON	3		1
OER	REO	3		1
OF	FO	2		3
OFT	TFO	3		1
ON	NO	2		3
ONCE	ECNO	4		1
OR	RO	2		1
OUTDID	DIDTUO	6	3	1
PENSIVE	EVISNEP	7		1
PLEASURE	ERUSAELP	8		1
POET	TEOP	4		1
SAW	WAS	3		2
SHINE	ENIHS	5		1
SHOW	WOHS	4		1
SOLITUDE	EDUTILOS	8		1
SPARKLING	GNILKRAPS	9		1
SPRIGHTLY	YLTHGIRPS	9		1
STARS	SRATS	5		1
STRETCHED	DEHCTERTS	9		1
SUCH	HCUS	4		1
TEN	NET	3		1
THAT	TAHT	4		3
THE	EHT	3		11
THEIR	RIEHT	5		1
THEM	MEHT	4		1
THEN	NEHT	4		1
THEY	YEHT	4		3
THOUGHT	THGUOHT	7		1

THOUSAND	DNASUOHT	8	1
TO	OT	2	1
TOSSING	GNISSOT	7	1
TREES	SEERT	5	1
TWINKLE	ELKNIWT	7	1
UPON	NOPU	4	1
VACANT	TNACAV	6	1
VALES	SELAV	5	1
WANDERED	DEREDNAW	8	1
WAVES	SEVAW	5	2
WAY	YAW	3	1
WEALTH	HTLAEW	6	1
WHAT	TAHW	4	1
WHEN	NEHW	4	2
WHICH	HCIHW	5	1
WITH	HTIW	4	2

LIST 5

16	↑	0	1
28	;	0	1
ALL	LLA	3	1
ALONG	GNOLA	5	1
BAY	YAB	3	1
BE	EB	2	1
BENEATH	HTAENEB	7	1
BLISS	SSILB	5	1
BREEZE	EZEERB	6	1
BROUGHT	THGUORB	7	1
CLOUD	DUOLC	5	1
COMPANY	YNAPMOC	7	1
CONTINUOUS	SUOUNITNOC	10	1
COUCH	HCUOC	5	1
COULD	DLUOC	5	1
CROWD	DWORC	5	1
DANCE	ECNAD	5	1
DANCED	DECNAD	6	1
DANCES	SECNAD	6	1
DANCING	GNICNAD	7	1
EYE	EYE	3	1
FILLS	SLLIF	5	1
FLASH	HSALF	5	1
FLOATS	STAOLF	6	1
FLUTTERING	GNIRETTULF	10	1
FOR	ROF	3	1
GAY	YAG	3	1
GLANCE	ECNALG	6	1
GLEE	EELG	4	1
GOLDEN	NEDLOG	6	1
HAD	DAH	3	1
HEADS	SDAEH	5	1

HEART	TRAEH	5		1
HIGH	HGIH	4		1
HILLS	SLLIH	5		1
HOST	TSOH	4		1
INWARD	DRAWNI	6		1
IS	SI	2		1
JOCUND	DNUCOJ	6		1
LAKE	EKAL	4		1
LIE	EIL	3		1
LINE	ENIL	4		1
LITTLE	ELTTIL	6		1
LONELY	YLENOL	6		1
MARGIN	NIGRAM	6		1
ME	EM	2		1
MILKY	YKLIM	5		1
MOOD	DOOM	4		1
NEVERENDING	GNIDNEREVEN	11	5	1
NOT	TON	3		1
OER	REO	3		1
OFT	TFO	3		1
ONCE	ECNO	4		1
OR	RO	2		1
OUTDID	DIDTUO	6	3	1
PENSIVE	EVISNEP	7		1
PLEASURE	ERUSAELP	8		1
POET	TEOP	4		1
SHINE	ENIHS	5		1
SHOW	WOHS	4		1
SOLITUDE	EDUTILOS	8		1
SPARKLING	GNILKRAPS	9		1
SPRIGHTLY	YLTHGIRPS	9		1

STARS	SRATS	5	1
STRETCHED	DEHCTERTS	9	1
SUCH	HCUS	4	1
TEN	NET	3	1
THEIR	RIEHT	5	1
THEM	MEHT	4	1
THEN	NEHT	4	1
THOUGHT	THGUOHT	7	1
THOUSAND	DNASUOHT	8	1
TO	OT	2	1
TOSSING	GNISSOT	7	1
TREES	SEERT	5	1
TWINKLE	ELKNIWT	7	1
UPON	NOPU	4	1
VACANT	TNACAV	6	1
VALES	SELAV	5	1
WANDERED	DEREDNAW	8	1
WAY	YAW	3	1
WEALTH	HTLAEW	6	1
WHAT	TAHW	4	1
WHICH	HCIHW	5	1
15	:	0	2
AS	SA	2	2
AT	TA	2	2
BESIDE	EDISEB	6	2
DAFFODILS	SLIDOFFAD	9	2
GAZED	DEZAG	5	2
MY	YM	2	2
SAW	WAS	3	2
WAVES	SEVAW	5	2

WHEN	NEHW	4	2
WITH	HTIW	4	2
BUT	TUB	3	3
OF	FO	2	3
ON	NO	2	3
THAT	TAHT	4	3
THEY	YEHT	4	3
	31	0	4
I	I	1	5
AND	DNA	3	6
A	A	1	7
IN	NI	2	7
THE	EHT	3	11
	31	0	12

LIST 6

15	:	0	1	2	4	6
15	:	0	1	3	2	6
16	↑	0	1	3	4	5
28	;	0	1	4	4	6
31	,	0	1	1	2	8
31	,	0	1	1	3	8
31	,	0	1	1	4	5
31	,	0	1	1	5	3
31	,	0	1	1	5	6
31	,	0	1	2	2	6
31	,	0	1	3	1	5
31	,	0	1	3	5	2
31	,	0	1	3	5	4
31	,	0	1	4	1	2
31	,	0	1	4	2	6
31	,	0	1	4	5	7
31	.	0	1	1	6	6
31	.	0	1	2	6	6
31	.	0	1	3	6	8
31	.	0	1	4	6	5
A	A	1	1	1	1	5
A	A	1	1	1	3	7
A	A	1	1	1	4	1
A	A	1	1	2	4	5
A	A	1	1	2	5	6
A	A	1	1	3	3	1
A	A	1	1	3	4	3
HAD	DAH	3	1	3	6	7
DANCED	DECNAD	6	1	3	1	5
STRETCHED	DEHCTERTS	9	1	2	3	2

WANDERED	DEREDNAW	8		1	1	1	2
GAZED	DEZAG	5		1	3	5	2
GAZED	DEZAG	5		1	3	5	4
OUTDID	DIDTUO	6	3	1	3	2	1
COULD	DLUOC	5		1	3	3	3
AND	DNA	3		1	1	2	7
AND	DNA	3		1	1	6	2
AND	DNA	3		1	2	2	1
AND	DNA	3		1	3	5	3
AND	DNA	3		1	4	5	1
AND	DNA	3		1	4	6	1
THOUSAND	DNASUOHT	8		1	2	5	2
JOCUND	DNUCOJ	6		1	3	4	4
MOOD	DOOM	4		1	4	2	6
INWARD	DRAWNI	6		1	4	3	5
CLOUD	DUOLC	5		1	1	1	6
CROWD	DWORC	5		1	1	3	8
BE	EB	2		1	3	3	6
DANCE	ECNAD	5		1	2	6	6
GLANCE	ECNALG	6		1	2	5	7
ONCE	ECNO	4		1	1	3	4
BESIDE	EDISEB	6		1	1	5	1
BESIDE	EDISEB	6		1	3	1	3
SOLITUDE	EDUTILOS	8		1	4	4	6
GLEE	EELG	4		1	3	2	6
THE	EHT	3		1	1	5	2
THE	EHT	3		1	1	5	5
THE	EHT	3		1	1	6	5
THE	EHT	3		1	2	1	3
THE	EHT	3		1	2	2	4

THE	EHT	3		1	2	4	2
THE	EHT	3		1	3	1	1
THE	EHT	3		1	3	2	2
THE	EHT	3		1	3	6	3
THE	EHT	3		1	4	4	3
THE	EHT	3		1	4	6	4
LIE	EIL	3		1	4	1	8
LAKE	EKAL	4		1	1	5	3
TWINKLE	ELKNIWT	7		1	2	2	2
LITTLE	ELTTIL	6		1	3	5	6
ME	EM	2		1	3	6	6
SHINE	ENIHS	5		1	2	1	6
LIVE	ENIL	4		1	2	3	5
PLEASURE	ERUSAELP	8		1	4	5	6
PENSIVE	EVISNEP	7		1	4	2	5
EYE	EYE	3		1	4	3	6
BREEZE	EZEERB	6		1	1	6	6
OF	FO	2		1	1	4	3
OF	FO	2		1	2	4	4
OF	FO	2		1	4	4	5
DANCING	GNICNAD	7		1	1	6	3
NEVERENDING	GNIDNEREVEN	11	5	1	2	3	4
SPARKLING	GNILKRAPS	9		1	3	2	3
FLUTTERING	GNIRETTULF	10		1	1	6	1
TOSSING	GNISSOT	7		1	2	6	1
ALONG	GNOLA	5		1	2	4	1
WHICH	HCIHW	5		1	4	4	1
COUCH	HCUOC	5		1	4	1	6
SUCH	HCUS	4		1	3	4	2
HIGH	HGIH	4		1	1	2	4

FLASH	HSALF	5	1	4	3	2
BENEATH	HTAENEB	7	1	1	5	4
WITH	HTIW	4	1	4	5	5
WITH	HTIW	4	1	4	6	3
WEALTH	HTLAEW	6	1	3	6	2
I	I	1	1	1	1	1
I	I	1	1	1	3	5
I	I	1	1	2	5	4
I	I	1	1	3	5	1
I	I	1	1	4	1	7
ALL	LLA	3	1	1	3	2
THEM	MEHT	4	1	3	1	4
GOLDEN	NEDLOG	6	1	1	4	4
THEN	NEHT	4	1	4	5	2
WHEN	NEHW	4	1	1	3	1
WHEN	NEHW	4	1	4	1	3
TEN	NET	3	1	2	5	1
IN	NI	2	1	1	6	4
IN	NI	2	1	2	3	3
IN	NI	2	1	2	6	4
IN	NI	2	1	3	2	5
IN	NI	2	1	3	4	1
IN	NI	2	1	4	2	1
IN	NI	2	1	4	2	4
MARGIN	NIGRAM	6	1	2	4	3
ON	NO	2	1	1	2	3
ON	NO	2	1	2	2	3
ON	NO	2	1	4	1	4
UPON	NOPU	4	1	4	3	3
TO	OT	2	1	3	6	5

OER	REO	3	1	1	2	5
THEIR	RIEHT	5	1	2	6	2
OR	RO	2	1	4	2	3
FOR	ROF	3	1	4	1	1
AS	SA	2	1	1	1	4
AS	SA	2	1	2	1	2
HEADS	SDAEH	5	1	2	6	3
DANCES	SECNAD	6	1	4	6	2
TREES	SEERT	5	1	1	5	6
VALES	SELAV	5	1	1	2	6
WAVES	SEVAW	5	1	3	1	2
WAVES	SEVAW	5	1	3	2	4
IS	SI	2	1	4	4	2
DAFFODILS	SLIDOFFAD	9	1	1	4	5
DAFFODILS	SLIDOFFAD	9	1	4	6	5
FILLS	SLLIF	5	1	4	5	7
HILLS	SLLIH	5	1	1	2	8
STARS	SRATS	5	1	2	1	4
BLISS	SSILB	5	1	4	4	4
FLOATS	STAOLF	6	1	1	2	2
CONTINUOUS	SUOUNITNOC	10	1	2	1	1
AT	TA	2	1	1	3	3
AT	TA	2	1	2	5	5
THAT	TAHT	4	1	1	2	1
THAT	TAHT	4	1	2	1	5
THAT	TAHT	4	1	4	3	4
WHAT	TAHW	4	1	3	6	1
POET	TEOP	4	1	3	3	2
OFT	TFO	3	1	4	1	2
THOUGHT	THGUOHT	7	1	3	5	7

BROUGHT	THGUORB	7	1	3	6	8
VACANT	TNACAV	6	1	4	2	2
NOT	TON	3	1	3	3	4
HEART	TRAEH	5	1	4	5	4
HOST	TSOH	4	1	1	4	2
BUT	TUB	3	1	3	1	6
BUT	TUB	3	1	3	3	5
BUT	TUB	3	1	3	5	5
SAW	WAS	3	1	1	3	6
SAW	WAS	3	1	2	5	3
SHOW	WOHS	4	1	3	6	4
BAY	YAB	3	1	2	4	6
GAY	YAG	3	1	3	3	7
WAY	YAW	3	1	2	2	6
THEY	YEHT	4	1	2	3	1
THEY	YEHT	4	1	3	1	7
THEY	YEHT	4	1	4	3	1
MILKY	YKLIM	5	1	2	2	5
LONELY	YLENOL	6	1	1	1	3
SPRIGHTLY	YLTHGIRPS	9	1	2	6	5
MY	YM	2	1	4	1	5
MY	YM	2	1	4	5	3
COMPANY	YNAPMOC	7	1	3	4	5

LIST 7

```
 2 3 3 4 - - - - - 2 3 3 3 4 4 - 2 3 4 - - - 2 2 3 3 - - 2
 4 2 4 4 2 3 4 5 5 2 - 5 5 - 2 5 6 6 6 6 - 3 4 4 5 3 4 - 3 5
 9 6 5 6 8 8 5 3 6 6 5 2 4 2 6 7 6 6 8 5 5 7 - 5 6 - 3 - 5 4
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 - - - - - - - - -
```

```
" " - 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 0 0 0 0
```

```
A A A A A A A A I I I I
```

```
5 5 6 8 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
A A A A A A A A I I I I
```

I	I	1	1	3	5	1
I	I	1	1	4	1	7
AS	SA	2	1	1	1	4
AS	SA	2	1	2	1	2
AT	TA	2	1	1	3	3
AT	TA	2	1	2	5	5
BE	EB	2	1	3	3	6
IN	NI	2	1	1	6	4
IN	NI	2	1	2	3	3
IN	NI	2	1	2	6	4
IN	NI	2	1	3	2	5
IN	NI	2	1	3	4	1
IN	NI	2	1	4	2	1
IN	NI	2	1	4	2	4
IS	SI	2	1	4	4	2
ME	EM	2	1	3	6	6
MY	YM	2	1	4	1	5
MY	YM	2	1	4	5	3
OF	FO	2	1	1	4	3
OF	FO	2	1	2	4	4
OF	FO	2	1	4	4	5
ON	NO	2	1	1	2	3
ON	NO	2	1	2	2	3
ON	NO	2	1	4	1	4
OR	RO	2	1	4	2	3
TO	OT	2	1	3	6	5
ALL	LLA	3	1	1	3	2
AND	DNA	3	1	1	2	7
AND	DNA	3	1	1	6	2
AND	DNA	3	1	2	2	1

AND	DNA	3	3	3	1	3
AND	DNA	3	5	4	1	1
AND	DNA	3	5	6	1	1
BAY	YAB	3	4	4	1	6
BUT	TUB	3	4	3	1	6
BUT	TUB	3	3	3	1	5
BUT	TUB	3	5	3	1	5
EYE	EYE	3	4	4	1	6
FOR	ROF	3	4	4	1	7
GAY	YAG	3	3	3	1	7
HAD	DAH	3	3	4	1	8
LIE	ETL	3	3	4	1	4
NOT	TON	3	3	3	1	5
OER	REO	3	3	2	1	2
OFT	TFO	3	3	4	1	6
SAW	WAS	3	3	5	1	3
SAW	WAS	3	3	5	1	1
TEN	NET	3	3	5	1	2
THE	EHT	3	3	5	1	5
THE	EHT	3	3	5	1	5
THE	EHT	3	3	6	1	3
THE	EHT	3	3	6	1	4
THE	EHT	3	3	2	1	2
THE	EHT	3	3	4	1	1
THE	EHT	3	3	2	1	2
THE	EHT	3	3	6	1	3
THE	EHT	3	3	4	1	3
THE	EHT	3	3	6	1	4
WAY	YAW	3	3	2	1	6

GLEE	EELG	4	1	3	2	6
HIGH	HGIH	4	1	1	2	4
HOST	TSOH	4	1	1	4	2
LAKE	EKAL	4	1	1	5	3
LINE	ENIL	4	1	2	3	5
MOOD	DOOM	4	1	4	2	6
ONCE	ECNO	4	1	1	3	4
POET	TEOP	4	1	3	3	2
SHOW	WOHS	4	1	3	6	4
SUCH	HCUS	4	1	3	4	2
THAT	TAHT	4	1	1	2	1
THAT	TAHT	4	1	2	1	5
THAT	TAHT	4	1	4	3	4
THEM	MEHT	4	1	3	1	4
THEN	NEHT	4	1	4	5	2
THEY	YEHT	4	1	2	3	1
THEY	YEHT	4	1	3	1	7
THEY	YEHT	4	1	4	3	1
UPON	NOPU	4	1	4	3	3
WHAT	TAHW	4	1	3	6	1
WHEN	NEHW	4	1	1	3	1
WHEN	NEHW	4	1	4	1	3
WITH	HTIW	4	1	4	5	5
WITH	HTIW	4	1	4	6	3
ALONG	GNOLA	5	1	2	4	1
BLISS	SSILB	5	1	4	4	4
CLOUD	DUOLC	5	1	1	1	6
COUCH	HCUOC	5	1	4	1	6
COULD	DLUOC	5	1	3	3	3
CROWD	DWORC	5	1	1	3	8

DANCE	ECNAD	5	1	2	6	6	7
FILLS	SLLIF	5	1	4	5	5	2
FLASH	HSALF	5	1	4	3	5	2
GAZED	DEZAG	5	1	3	5	5	4
GAZED	DEZAG	5	1	3	5	5	4
HEADS	SDAEH	5	1	2	6	5	3
HEART	TRAEH	5	1	4	5	5	4
HILLS	SLLIH	5	1	1	2	5	8
MILKY	YKLIM	5	1	2	2	5	5
SHINE	ENIHS	5	1	2	1	5	6
STARS	SRATS	5	1	2	2	5	4
THEIR	RIEHT	5	1	2	2	5	2
TREES	SEERT	5	1	1	1	5	6
VALES	SELAV	5	1	1	1	5	6
WAVES	SEVAW	5	1	3	1	5	2
WAVES	SEVAW	5	1	3	2	5	4
WHICH	HCIHW	5	1	4	4	5	1
BESIDE	EDISEB	6	1	1	5	6	1
BESIDE	EDISEB	6	1	3	1	6	3
BREEZE	EZEERB	6	1	1	6	6	6
DANCED	DECNAD	6	1	3	1	6	5
DANCES	SECNAD	6	1	4	6	6	2
FLOATS	STAOLF	6	1	1	2	6	2
GLANCE	ECNALG	6	1	2	5	6	7
GOLDEN	NEDLOG	6	1	4	4	6	4
INWARD	DRAWNI	6	1	4	3	6	5
JOCUND	DNUCOJ	6	1	3	4	6	4
LITTLE	ELTTIL	6	1	3	5	6	6
LONELY	YLENOL	6	1	1	1	6	3
MARGIN	NIGRAM	6	1	2	4	6	3

OUTDID	DIDTUO	6	3	1	3	2	1
VACANT	TNACAV	6		1	4	2	2
WEALTH	HTLAEW	6		1	3	6	2
BENEATH	HTAENEB	7		1	1	5	4
BROUGHT	THGUORB	7		1	3	6	8
COMPANY	YNAPMOC	7		1	3	4	5
DANCING	GNICNAD	7		1	1	6	3
PENSIVE	EVISNEP	7		1	4	2	5
THOUGHT	THGUOHT	7		1	3	5	7
TOSSING	GNISSOT	7		1	2	6	1
TWINKLE	ELKNIWT	7		1	2	2	2
PLEASURE	ERUSAELP	8		1	4	5	6
SOLITUDE	EDUTILOS	8		1	4	4	6
THOUSAND	DNASUOHT	8		1	2	5	2
WANDERED	DEREDNAW	8		1	1	1	2
DAFFODILS	SLIDOFFAD	9		1	1	4	5
DAFFODILS	SLIDOFFAD	9		1	4	6	5
SPARKLING	GNILKRAPS	9		1	3	2	3
SPRIGHTLY	YLTHGIRPS	9		1	2	6	5
STRETCHED	DEHCTERTS	9		1	2	3	2
CONTINUOUS	SUOUNITNOC	10		1	2	1	1
FLUTTERING	GNIRETTULF	10		1	1	6	1
NEVERENDING	GNIDNEREVEN	11	5	1	2	3	4