AUTOMATED DOCUMENT RETRIEVAL BASED ON DISTRIBUTED PROCESSING

STEPHEN HENRY JAMIESON

Thesis submitted for the degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF ASTON IN BIRMINGHAM

OCTOBER 1981

THE UNIVERSITY OF ASTON IN BIRMINGHAM

Automated Document Retrieval Based on Distributed Processing

Stephen Henry Jamieson

Thesis submitted for the degree of
DOCTOR OF PHILOSOPHY

Summary

A personal view of bibliographic information retrieval is presented. It concludes that for the most part, the results of research in this area have tended to be ignored by the designers of large on-line retrieval systems. Several reasons for this are mooted and a number of requirements are established for overcoming the inertia of system designers. Notably, the need for large scale experimentation with retrieval methods in operational environments is highlighted.

To meet those requirements, it is described how a microcomputer system can be used to access on-line bibliographic retrieval systems rather than an ordinary computer terminal. The capability of the retrieval system plus microcomputer configuration is extensively evaluated. Firstly, potential applications are reviewed and secondly, an in depth study is carried out of the implementation of a weighted search term retrieval mechanism. A suitable microcomputer system was designed and built for the purpose of the study. Its structure is very sophisticated employing three microprocessors.

The main conclusion of the work is that a relatively inexpensive microcomputer can provide the research community with a powerful tool allowing the evaluation of experimental retrieval techniques in operational environments.

Keywords

Information Retrieval. On-line Retrieval Systems. Evaluation and Experimentation. Microcomputer. Intelligent Terminal.

1981

Acknowledgements

I wish to express my appreciation for the help and encouragement given to me by many people, including my colleagues in the Computer Centre at the University of Aston, and others whom I have met and come to know through my work in bibliographic information retrieval. In particular, I thank Dr R N Oddy, my supervisor, for allowing me to benefit from his knowledge and experience, and for always being ready to discuss problems and offer advice. I am grateful to Mr K Bowcock for the provision of computing facilities, Dr M J Walker for assistance with the hardware aspects of the work, Miss E D Barraclough for making the MEDUSA information retrieval system available and the Science Research Council for financial support. Finally, I am deeply indebted to Sharon, my wife, for her patience and understanding.

For my Parents.

LIST OF CONTENTS

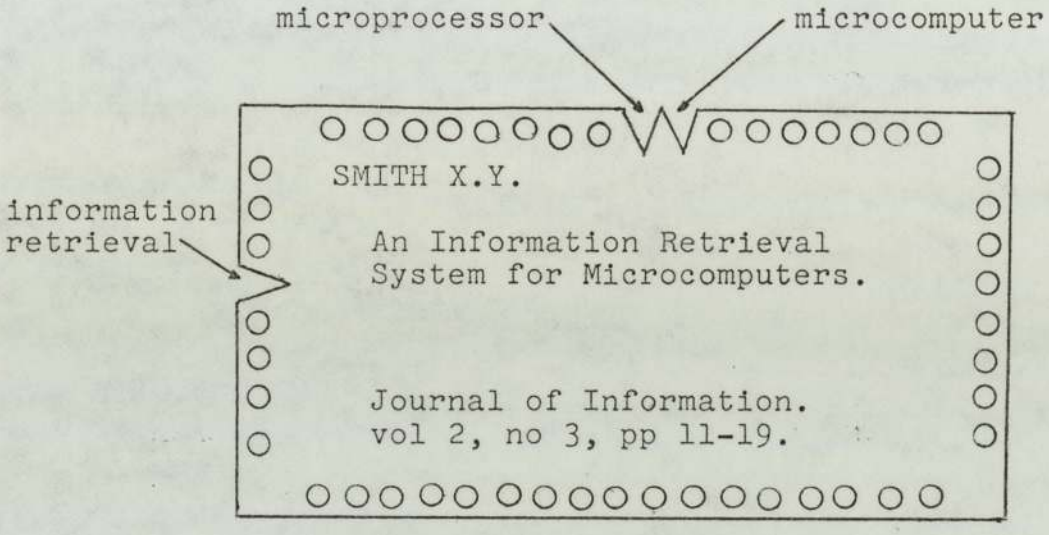
LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The availability of computers within libraries to assist those wishing to carry out a thorough literature survey is now becoming well established. Many public and institutional libraries own computer terminals and are able to offer access to on-line bibliographic retrieval systems based in the United Kingdom, the rest of Europe and in North America. Relief from the tedium of conducting manual literature surveys is welcomed by the majority of library users.

The actual mechanisation of the literature search process was begun many years ago, long before the introduction of computers. The introduction of catalogues based on edge-notched cards or optical coincidence cards (Kent 1966 and Paice 1977) represent the first attempts to ease the task of 'surveying the literature'. Card indexes of this type allow the selection of entries which satisfy queries or search statements based on Boolean logic. The Boolean search mechanism should be familiar. Each query consists of a series of search terms (taken from the set of terms indexing the catalogue) connected by any of the Boolean operators - AND, OR & NOT. Each entry within the catalogue whose index terms satisfy the Boolean expression is retrieved. Figure 1.1 illustrates an edge-notched card. The notches made around the edge of the card indicate the presence of particular indexing terms which represent the information content of the publication. To isolate all



Each notch position corresponds to a particular index term.

Figure 1.1. An Edge-Notched Card (part of an early mechanical retrieval device).

publications indexed by a specific term a long needle is pushed through the card deck in the appropriate notch position. As the needle is raised all the catalogue entries *not* indexed by the term are raised with it, thus isolating those entries which are indexed by the term. Operating successively on the subsets of cards resulting from each selection leads to groups of entries which satisfy queries expressed in the form of Boolean expressions. Examples of such query statements are:-

1) information retrieval AND (microcomputer OR microprocessor),
2) query language AND information retrieval AND NOT database.

We shall refer to queries of this type as Boolean queries.

A survey of the large commercial information retrieval systems reveals that the retrieval mechanisms they operate are all of the Boolean type. In other words the computer systems operate the same mechanism as the 'old-fashioned' mechanical systems described above. Hall (1977) and Lancaster and Fayen (1973) provide substantial surveys of the computerised retrieval systems currently in existence. The implementation and operation of systems based upon Boolean search logic is relatively simple and straightforward. So straightforward in fact that the same effect was achieved before the days of the mighty computer with simple pieces of cardboard (figure 1.1). Now that computers *are* here, we can do things that are not feasible with manual methods, so there is no longer any excuse for doing things that we know to be inadequate. (The appropriateness of the Boolean

search mechanism has been questioned in the past eg Verhoeff *et al* (1961)). The computer systems do of course have advantages compared to their mechanical counterparts: they are able to operate with larger catalogues (ie files) of records; they are easier to use, requiring less effort on the part of the user, and require no maintenance on behalf of the libraries providing access to them.

Since computerisation of the retrieval process began, research has established new and better ways to organise the data within the computer; other techniques of representing the information content of publications and more convenient methods for users to communicate their queries or information needs to the machine. However, it is evident that researchers have failed to influence the retrieval system designers and entrepreneurs within the commercial environment who are responsible for the existence of the large commercial systems. The motivation for the work described in this thesis stems from this observation.

The following chapter presents a survey and discussion of the present state of information retrieval expanding upon some of the comments we have already made. Some reasons are suggested for the lack of influence results of research have had over retrieval methods used within large retrieval systems. One major point that is raised is the lack of validity of most research to the operational environment. Research has, in the past, generally been carried out in the computer laboratory using relatively small test collections of document descriptors and small sets of test queries. There is no evidence that results obtained in this fashion can be extrapolated to the operational arena with its much larger parameters. We conject that to increase the likelihood of research influencing the kind of facilities available from the large systems experimentation should be carried out in more realistic environments.

The remaining chapters are concerned with the development of a tool which will allow the testing of some retrieval methods on a much larger scale than has hitherto been realised. We exploit the cheap processing power made available by emerging computer hardware technology. By placing a number of microprocessors within a computer terminal and providing some form of backing store the result is what has been referred to casually as an 'intelligent terminal'. Such a computer terminal, when used in conjunction with an existing information retrieval system opens up new doors for experimentation in this field. The added processing capability can make possible the operation of experimental retrieval mechanisms in conjunction with large scale, operational systems. Experimental results gleaned in this

fashion have immediate applicability to the operational environment thus overcoming the major drawback associated with earlier work. We do not propose that experimentation under laboratory conditions becomes invalid or obsolete. New and speculative ideas will often best be tested first on much smaller scales.

Chapter 3 reviews the application of new hardware technologies in the area of bibliographic information retrieval. We also discuss work carried out in related application areas such as databases. The chapter concludes with a detailed discussion of our suggestion for using an intelligent terminal in conjunction with an existing retrieval system. The potential of such a configuration for a number of different applications in the area of information retrieval is fully described in chapter 4.

A design for an intelligent terminal which was realised in practice is put forward in chapter 5. Chapters 6 and 7 describe software associated with the terminal. In chapter 6 techniques are described for operating a weighted search term retrieval mechanism within the terminal. Details of a number of different algorithms are presented .

Chapter 7 describes the actual programs executed by the intelligent terminal. Programs are required to interface the terminal to an on-line retrieval system, allow communication with a user and operate the algorithms detailed in chapter 6.

The terminal is programmed to operate in conjunction with the MEDUSA retrieval system, an experimental on-line retrieval system based at the University of Newcastle upon Tyne. We close chapter 7 with an evaluation of the suitability of the terminal for operating weighted search term retrieval. Other results are reported in chapter 6 and are concerned with the algorithms which allow the terminal to operate the retrieval mechanism based upon term weighting.

The main points raised in this thesis are summarised in the final chapter and suggestions for future work are made.

A Note on Terminology

Throughout this thesis our concern lies with on-line bibliographic information retrieval systems. We may, at times, leave out the term 'bibliographic' 'on-line' or even 'information' for the sake of brevity. In such cases no reference is being made to database or data retrieval systems. We also write about 'users' of on-line bibliographic retrieval systems. Under this collective term we include any person who uses a retrieval system, whether they be a trained intermediary carrying out a search on behalf of another, or somebody using the system to satisfy his own personal information needs.

CHAPTER 2

A VIEW OF INFORMATION RETRIEVAL

2.1 Introduction

The existence of research in any area of applied technology is founded upon the benefits it eventually brings to society. Thus, research into the design and application of computer systems should benefit the users of the systems. Sadly, there is evidence that research in the area of on-line bibliographic information retrieval systems has, to a large extent, failed to generate benefits for 'information-seekers'. We do not say that research in this area is carried out incompetently or is trivial and worthless. On the contrary, progress has been made and many experiments have lead to valuable results and conclusions concerning methods of retrieval, query formulation and information representation. We simply point out that for all the efforts of those engaged in research in information retrieval there has been little substantial change in the methods used by large public commercial retrieval systems and the service provided to their users. We table the evidence which supports this view in this chapter. We also put forward some reasons why the results of research have failed to influence the entrepreneur.

In the concluding section we mention the danger facing research in information retrieval, that of its gradual demise and eventual disappearance. The remaining chapters of the thesis are intended to offer a glimmer of hope and comfort to any disillusioned

researcher in this field: a scheme for implementing and evaluating new retrieval techniques is described which can eventually lead to the generation of convincing evidence of the value of new retrieval techniques.

2.2 The Information Retrieval Dichotomy

Activity within the area of information retrieval appears to be dichotomous. On the one hand there are commercial organisations making access to on-line bibliographic retrieval systems possible from throughout the world and supporting simple 'traditional' retrieval techniques while, on the other, the research community has seen the development of many experimental retrieval systems based upon a wide variety of retrieval techniques.

2.2.1 Commercial Information Retrieval Systems

Many public and academic libraries allow access to one or more on-line bibliographic information retrieval systems. The systems are based in either Europe or North America and are accessible through networks such as TYMNET and EURONET. Each system makes available a number of different document databases. One of the largest systems is DIALOG (Lockheed 1980) which has more than one hundred databases on-line. Usually, each database is associated with a particular subject area covering social & natural sciences, applied science & technology, and business & economics. Hall (1977) and Lancaster and Fayen (1973) provide comprehensive surveys of retrieval systems which existed at their

time of publication. Bourne (1980) describes the history of on-line retrieval systems. The majority of document databases consist of records which contain such details as authors, the title of the document, its abstract, indexing terms and citation information. The full text of the document is not normally stored within the computer. Other databases are accessible which contain census data, patent descriptions or details of newspaper articles.

Retrieval is based upon Boolean logic. Search terms are combined to form queries using the Boolean operators, AND, OR & NOT. Search terms can be entries found in many of the fields within a document record, eg an author, index term, language of publication, etc. There are limited facilities for 'full text' searching specifically within document abstracts and some systems make available adjacency operators for use in this mode, eg. two text words may be specified as having to occur next to one another or within the same sentence. Other facilities which are generally available allow searches to be stored in the computer for use in later interactions, references to be printed out on Lineprinters (such hard copy has to be mailed to the user) and even copies of individual documents to be ordered.

In some sense the systems are quite sophisticated. They can support a large number of users interacting with the system simultaneously: they can be accessed from throughout the world and make available millions of records on-line. However, there are drawbacks. In particular it can be difficult for users to formulate a query, using Boolean search logic, which adequately

represents their information need and ensures that a large proportion of items retrieved are relevant. Williams (1971) has referred to the inadequacy of Boolean query formulation:-

> " ... (there are) several stages of uncertainty in the creation, indexing, storing and retrieving of concept information. In addition, at each stage ambiguity can exist in both directions from concept to term and term to concept. ..... Boolean query statements admit only two conditions, true or false, whereas an algorithm needed for concept retrieval must admit a range of certainty or probability."

Williams has used the phrase 'concept retrieval' to distinguish bibliographic retrieval from data retrieval. Dillon and Desper (1980) relate how difficult it is for users to reformulate Boolean queries after the receipt of an inadequate reply to a search statement. Salton (1972) also adds weight to the arguments against Boolean search logic:-

> " .... to obtain a desired level of performance ...., it is necessary to find just the right kind of query formulation. For if the query formulation is too broad the retrieved set is likely to be very large, thereby imposing a great burden on the user and producing low precision even when recall is fairly high; on the other hand, a narrow formulation is likely to restrict severely the level of recall which may be attained."

Verhoeff *et al* (1961) use a mathematical analysis to argue against the effectiveness of Boolean search logic.

One virtue of Boolean queries which can be used in their defence is that they are quite easily processed by a computer given appropriate file structures, and place little burden on

machine resources. It is interesting to note that Boolean search logic was used in old mechanical retrieval aids such as optical coincidence cards (Kent 1966 and Paice 1977). Present day commercial on-line retrieval systems appear to be simple computerised versions of those earlier mechanical devices. Early computer systems were centred on batch systems employing magnetic tapes as the main form of mass storage. On-line systems followed in the wake of time-sharing operating systems and the availability of faster, more efficient backing store - disks.

Some retrieval software packages available commercially do offer a larger variety of retrieval mechanisms. These packages are mounted by organisations on their own computers and are for private use. Some of these systems allow complete document texts to be entered for example, and support automatic indexing and full text retrieval. Retrieval mechanisms offered include, coordination level and weighted term searching. (Descriptions of these mechanisms can be found in standard textbooks such as Van Rijsbergen 1979 and Paice 1977). Examples of such packages are STATUS (Teskey 1980) and the IBM product, STAIRS.

2.2.2 Research in Information Retrieval

Researchers, recognising the deficiencies in existing retrieval systems, have endeavoured to determine improved methods for analysing and representing the information content of documents, retrieval mechanisms which yield better retrieval performance in terms of recall and precision, and more

convenient ways for users to express their information needs. Williams (1971) quite categorically states that,

> " .. an interactive system is not achieved by simply adding an input terminal onto a batch searching system",

thus suggesting perhaps, that this is really all that the purveyors of on-line retrieval systems have done.

As there are many who have been, and many who are, actively engaged in research in this area we don't really wish to single out specific individuals. King (1978) presents a summary of the history of information retrieval research, while perusal of journals such as 'Information Processing and Management', 'Journal of Documentation' and 'Journal of the American Society for Information Science', will indicate the kind of work carried out. Research began as it became apparent that computers could be usefully applied to bibliographic retrieval. Since then the amount of research activity has greatly increased. The arrival of on-line computing facilities proved particularly stimulating.

Research has generally had one of two forms. Either, programs are written to run on 'local' computers to operate the retrieval mechanism under test, and relatively small 'test' document collections are used as data, or experimental work is carried out with small scale operational retrieval systems which can offer a service to a small, enclosed, user community. An example of the former is the work carried out by Sparck-Jones and her colleagues, (eg

Sparck-Jones 1972, Sparck-Jones 1973, or Robertson and Sparck-Jones 1976). She has, for example, experimented widely with search term weighting methods and feedback methods (which take account of user relevance assessments in order to modify a query automatically). The test collections she used in her experiments have been used by many others and are quite small compared to the number of documents found in databases mounted on commercial retrieval systems. Typical examples are the CRANFIELD 200 and CRANFIELD 1400 collections (the figures refer to the number of document records they contain). These were established during early experimental work carried out by Cleverdon in Cranfield, England (Cleverdon *et al* 1966). (The 200 collection is in fact a subset of the 1400 collection).
A test collection consists of a set of document records; a set of 'typical' queries and a set of relevance judgements for each query, i.e. a list of documents relevant to the particular query. An 'experiment' usually involves writing programs to process each test query according to the retrieval mechanism; ascertaining the documents retrieved by each query; assessing the relevance or non-relevance of each document 'retrieved' by checking against the list of relevant documents and thence calculating precision and recall results. The programs may be run as background jobs as there is no interaction between them and users (who, of course, do not exist). Larger test collections do exist, although they have been used less extensively than the CRANFIELD collections. Miller (1971) used 35,000 document records taken from the MEDLINE database while Barker *et al* (1972) used a set of data consisting of 27,000 items. Even these larger test collections are a lot smaller than

databases found in practice, which can vary in size from a half to one million document records.

The second strategy for experimenters is to implement small scale operational retrieval systems - 'pseudo' operational systems - which have all the facilities for interacting with users and performing on-line retrieval. A famous example is the SMART retrieval system developed by Salton and his co-workers (Salton 1971). Other examples are SIRE (McGill *et al* 1976, Noreault *et al* 1977), LEADERMART (Hillman 1973, Hillman and Kasarda 1969) and FIRST (Dattola 1979). These systems are test beds for various retrieval mechanisms. Methods for automatic indexing, weighting search terms, and improvements to the user interface are aspects which have received attention. The systems usually provide a retrieval service to small 'private' user communities within a university department or faculty. Often the service is not continuous but lasts only for the duration of an experiment and the size of the databases used are still quite small compared to operational systems.

Very recently some interesting work has been carried out within the National Library of Medicine (NLM) in the United States. Doszkocs (1978a, 1978b, Doszkocs *et al* 1979) has described the development of new facilities within the ELHILL software which constitutes the on-line retrieval system operated by NLM. This is probably the first instance of research being carried out with a full scale operational retrieval system.

2.2.3 Summary

When one looks at information retrieval from our particular standpoint one immediately reflects that research is failing to influence the kind of retrieval facilities provided by the commercial organisations. Given the stage which research in information retrieval is at, we may still be some distance from the realisation of Bush's MEMEX system (Bush 1945) or Licklider's library of the future (Licklider 1965) but with regard to commercial systems we are further away still. Some researchers may feel that it is only a matter of time before operational systems adopt some of their ideas. This is not a view that we share. Barraclough (1977) reviews the state of the art in on-line searching. She points out that present day on-line retrieval systems are essentially the same as early batch systems from the point of view of the retrieval methods they employ and goes on further to say that there is a substantial amount of experimental work that has been carried out which could prove the basis for future system design. Williams (1977) describes recent advances in on-line retrieval systems, particularly commercial systems, yet can only point to increases in size and subject coverage! Several years ago Williams (1971) made the following statement:-

> "Many machine retrieval systems have reached a plateau of effectiveness"

Little has changed.

2.3 Reasons for the Dichotomy

The major reason for the separation of the research and commercial communities lies in the nature of the experimental work carried out by researchers. As we have observed, testing of retrieval methods has generally been carried out in a 'laboratory' environment using small size test collections. It is not immediately apparent how one could justify the extrapolation of results obtained in this way and confidently predict that the same kind of performance would be obtained if the retrieval methods were implemented within large operational retrieval systems. Robertson has pointed this out already (Robertson 1976). He gives substantial consideration to the difficulties associated with the extrapolation of results obtained under certain conditions to situations where the conditions differ. He writes about the 'same difference principle';

> "if system A performs substantially better than system B under one set of conditions, then it will do so under any other set of conditions".

Robertson suggests that most testing and experimental work to date has placed a high degree of reliance on the validity of this principle but goes on to describe experimental evidence which shows it not to hold. An attempt is made to develop a basic underlying theory of information retrieval so that the effect on the performance of retrieval systems of varying conditions can be predicted, thus enabling results of evaluations to be extrapolated with a higher degree of accuracy. The resulting information retrieval model is, naturally,

quite complicated and only partly achieves the ultimate goal. He admits, 'the science of IR system design is a long way from being theory based'.

A further reason for the limited influence of information retrieval research concerns the machine resources required to operate certain retrieval mechanisms. Some retrieval methods proposed by researchers are much more complicated than simple Boolean techniques and consequently command much more of a computing resource for their operation. It is very likely that such retrieval mechanisms are much more costly to operate; leading to either fewer users being allowed to access a system simultaneously in order to maintain acceptable response times, or the need for more computing power to be purchased to maintain similar levels of access. In either case there are obvious financial consequences for commercial organisations. Salton (1971) has recognised this kind of limitation and uses it as a justification for the clustering techniques implemented in the SMART retrieval system:-

> "Since the user is responsible for providing the information to be returned to the system and therefore is expected to be present during a search, it is obviously not possible to perform a full search of all stored items except where very long search times are tolerable. Therefore, fast cluster search strategies have been implemented and tested, based on a document grouping method which restricts the actual search to only certain document groups for each given request", (page 5),

Dattola implemented 'SMART like' features in the FIRST retrieval system (Dattola 1979). He reports that the document collection needed to be clustered to improve response time. Carroll (1976) refers to

the problem as the 'computer crunch' and underlines the severe demands made by information scientists of computing resources.

The final reason why commercial organisations appear loath to take up the ideas of the research community is one of economics. Existing on-line retrieval systems are large in terms of computing power and required a large financial investment in order to establish their existence. It is unlikely that major changes will be made to them until a large proportion of the initial investment has been recouped.

2.4 Conclusions

Any research which, after having been carried out over a long period of time, does not begin to bear, or promise to bear, fruit will eventually lose the financial support coming from public and commercial purses. Consequently, the research effort will gradually shrink and eventually disappear.

I have described a number of reasons why results of research have had little effect on the thinking of retrieval system designers. In particular there has to be some testing and evaluation of ideas in a much more realistic fashion than has hitherto been the case. Of course there is still, and will always be, the need for basic research and small scale experiments as people toy with newly formed ideas and hunches. Theoretical aspects of information retrieval still need considerable attention. Robertson (1977) describes theories pertaining to information retrieval already developed, highlighting inadequacies and deficiencies and

emphasising the importance of the theoretical justification of ideas. However, ultimate testing of theories must be carried out in such a way that results prove that better performance would be obtained by their use in operational retrieval systems. Van Rijsbergen (1979) hints at this requirement in his book:-

> " . . . (I hope) . . . that some people will go on to experiment with (new techniques) and generate new, convincing evidence of their efficiency and effectiveness." (page 3)

The question that must now be examined is how should experiments be organised which will produce the 'convincing evidence'.

The remaining chapters of this thesis describes, and examines the validity of, a strategy for implementing retrieval techniques which can circumvent the problems we have pointed out.

CHAPTER 3

THE APPLICATION OF NEW HARDWARE TECHNOLOGY

3.1 Introduction

In this chapter we describe how recent advances in computer hardware technology can help to overcome problems presently associated with research in information retrieval. The problems, which were described in the previous chapter, are as follows:-

i) research to date has been carried out in a laboratory environment with very small test document collections and sets of test queries: more realistic evaluation and testing is required, preferably in an operational environment.

ii) present day, general purpose computers are not sufficiently powerful and are not particularly suited to operating complicated retrieval techniques: the implementation of some retrieval mechanisms is not an attractive proposition to commercial organisations.

iii) large financial investments have been made in existing operational retrieval systems: their replacement, or any major modifications to them, is unlikely in the very near future.

These difficulties have to be addressed in order for the efforts of researchers to begin to affect the retrieval services offered to searchers by the large systems.

We begin by briefly reviewing some of the recent developments in computer technology - microprocessors, memory devices and associative processors. In section 3.3 we describe a new philosophy for computer system design - that of designing the computer to fit the problem rather than fitting the problem solution to an existing machine. Application areas where this philosophy has been used already are reported and recent attempts to apply it in the area of information retrieval are described. We conclude that the efforts of this sort in the area of information retrieval have still failed to help circumvent all the problems we have listed. In contrast, section 3.6 describes our proposal. An inexpensive microprocessor system is used in conjunction with an existing information retrieval system allowing the implementation of some experimental retrieval techniques. By incorporating an existing operational retrieval system we arrive at a configuration which has the potential for providing practical and economical solutions to the three problems listed above.

3.2 New Technology

The rapid advances in computer technology are well known and have been referred to extensively in journals, newspapers, and magazines alike. The ubiquitous microprocessor is going from strength to strength and is approaching the performance levels of minicomputers. Developments

have also lead to faster and cheaper memory, and new kinds of memory technology including devices with intrinsic processing capability. The overall result is the availability of greater processing power at much lower costs.

New storage technology includes magnetic bubble memories and charge coupled devices (CCDs). Magnetic bubble memories are predicted by some to be eventual replacements for disks (eg. Chen, 1978). Bernhard (1980) describes the physics and method of operation of magnetic bubble devices. Bubble memories offering up to one million bits of storage capacity have been introduced by manufacturers. The devices offer non-volatile memory with slightly higher access speeds than disks. However the average cost per bit is higher. A review of bubble devices available is provided by Myers (1977). He also presents a substantial bibliography on the subject. Charge coupled devices offer higher access speeds of two or three orders of magnitude faster than bubble memories but costs are higher and data is volatile (Mackenzie 1977). The physics of CCDs is described by Hobson (1977). The volatility but relatively fast access time makes the CCD a candidate for cache memory (memory lying between main memory and disks in a storage hierarchy). The physical size of CCDs is a lot smaller than that of bubble devices.

Optical memories have been described in the literature but their development is still in its infancy and no devices are generally available from manufacturers yet. Very high storage capacities and associative access (see later) are their main characteristics.

Knight (1975) describes how holographic memories operate. D'Alleyrand (1978), points to a problem that will arise with the introduction of large capacity storage devices, such as holographic memories, within information retrieval systems: as storage capacities of computers drastically increase, search techniques must become more refined to ensure that recall is maintained without the user having to suffer unacceptable levels of precision.

General texts describing the design and application of new memory devices are those by Middlehoek *et al* (1976) and Melen and Buss (1977).

A memory device with associative properties is not a new idea but has been receiving much attention of late. Seeber and Lindquist (1962) and Kiseda *et al* (1961) are examples of early papers on the subject. Early work was held up by the high cost and the difficulties experienced with the manufacture of associative memories but the recent advances in semi-conductor technology has relieved some of the problems. Associative memories are now quite viable although still limited in size.

Data is stored in an associative memory at an unspecified location and retrieved on the basis of some knowledge of its content as opposed to conventional storage techniques in which locating data is based upon address. Overviews of associative memories and extensive bibliographies are given by Thurber and Wald (1975), Parhami (1973) and Minker (1971). Applications of associative memories are

described by Thurber and Berge (1971). The basic structure of an associative memory is outlined in figure 3.1. The bit pattern to be looked for is placed in the TESTDATA register. The MASK register is used to designate particular bit positions to be used in a matching operation. All words are selected (WORDSELECT) and the COMBINEDRESULT flag is checked for matches. If this flag is not set then no word within the memory contains the bit pattern looked for. If the flag is set then a match (or matches) did occur and the RESULTREGISTER is scanned for matching locations. This series of operations can be executed in a very short period of time. Accessing data is much faster than with conventional stores in which search keys have to be translated to unique addresses using hashing techniques. A wide variety of operations can be carried out on the data within the memory. For example, the greatest or smallest values can be isolated or data can be added to selected words (Foster, 1976). Lea (1978) describes how an associative memory can speed text compression and decompression.

Associative memories still tend to be rather small and need to be used as auxilliary memory units. Lea (1975) reports experiments with a memory of 128 words with 32 bits per word, although he says that larger devices are available (eg. 2048 words with 50 bits per word). Although searching can be carried out very quickly, their small size effectively increases search times as data has to be continually loaded into the memory during the search operation in order for a complete file of keys to be scanned. Their small size is therefore a serious disadvantage and it

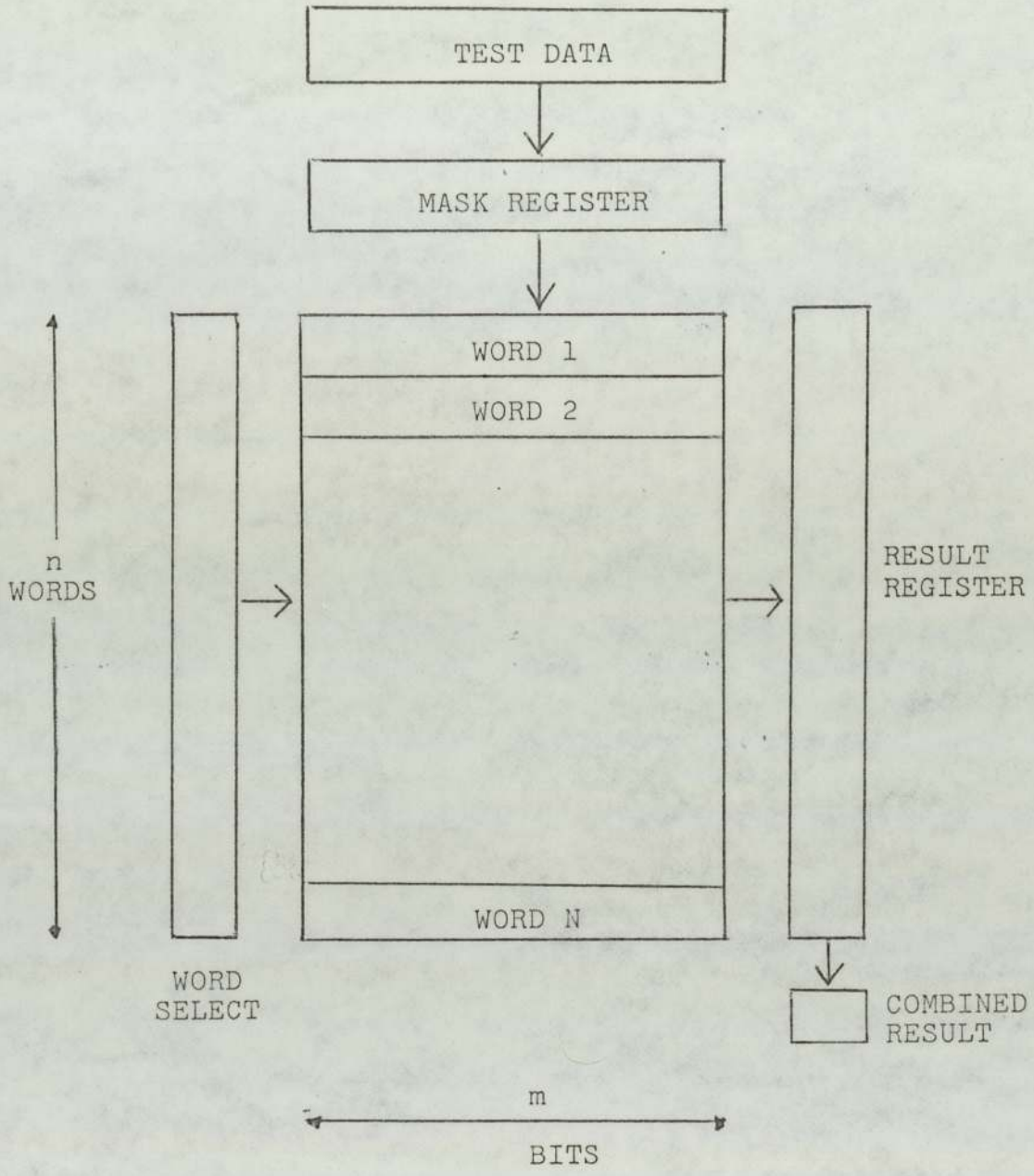


Figure 3.1. The Basic Structure of an Associative Memory.

remains to be seen whether future developments in large scale integration will bring about much larger associative memories.

A less expensive form of associative, or content addressable, memory has been obtained by placing processing capability in the read/write heads of disks. The added intelligence allows selection and matching functions to be performed outside of the computer's CPU. Instead of reading a complete data file into the CPU and selecting the required records, the selection key is sent to the read/write head and only those records which match the key are transferred to the CPU. Slotnick (1970) first put forward the proposal for devices of this nature. An example of a commercial product based on the idea is CAFS - Content Addressable Filestore (Colouris *et al* 1972, Maller 1979). The advantage of such systems is that they use 'state of the art technology' (disks) and data does not need to be moved into and out of the associative store prior to, and after, a retrieval operation. In fact the associative logic elements are in the read/write head and contain the search key as opposed to 'traditional' associative memories which contain all the data to be searched in associative elements. This is the key to the economy of 'logic per track' devices. Maller (1979) explains how sophisticated search operations may be carried out such as those required for retrieval based upon weighted search terms or Boolean expressions. The addition of search logic in this way to memory devices other than disks has been experimented with. Doty *et al* (1980) describes a magnetic bubble device which can execute a number of operations required within a relational database system.

All the devices described so far share a common attribute: they can be used in existing computers as replacements for existing components eg. simple disks can be replaced by a CAFS device. In the next section we shall consider changes in computer technology at a higher level, that of complete computer architectures.

3.3 Specialised Computer Architectures

In parallel with hardware development and invention there has also been changes in the attitudes and thinking of some computer designers. Architecturalconcepts such as distributed and parallel computing are being widely experimented with and have been applied to a limited extent in some commercially available machines. However an entirely new philosophy for the design of computer systems is gradually emerging. The underlying trend is to move away from the rule of choosing a single processor, general purpose computer for each and every application and to produce machine designs specifically suited to the tasks to be carried out. In other words, rather than forcing a system solution to fit the machine, the machine is designed to fit the solution. A computer which is perfectly suited to scientific applications such as the monitoring and control of an Apollo moon mission is not necessarily as equally suited to supporting database applications. However, this has been the pattern of events in the past. No matter the particular application in hand, a general purpose computer has been 'taken from the shelf' and software written to make the machine perform the required operations. Coercing of software in this manner

can lead to inefficient systems and the incapability of machines to cope with large workloads. Many authors have raised similar arguments. Hsiao (1979), a leading exponent of the 'tailored architecture' philosophy, has the following to say with regard to machines designed specifically for database operations:-

> "For performance, conventional computer systems were not designed for database management. These Von Neumann type computers are good for the preparation and execution of programs for numerical computations and for simple data processing. Database management is instead concerned with the storage, retrieval and management of large databases and requires quick search for concurrent access."

Kuck (1976) in support of parallel processing has the following to say:-

> " . . . a computer designer must worry about what applications are to be served by his machine. In the days of truly general purpose computers, any colour you want as long as it is black, was sufficient."

He cites the increasing costs of software and asks how machines can be better organised so that software and applications are better and less expensively handled. Iizuka *et al* (1975), Mueller (1977) and Beavin and Lewin (1973) argue similar points.

Obviously, the high cost of computer systems, particularly hardware, has required the production of 'standard models' for the sake of economy. However, now that hardware costs have fallen dramatically and computer application areas are now so

numerous and diverse, special attention given to some application areas can be much more easily justified. Databases is one of these application areas. Vick *et al* (1980), possibly worried by the economic disadvantages of specialised computer architectures, describes a design which allows reconfiguration by the programmer as the application demands. Programmers can change the system architecture into states which match the peculiarities of their algorithms. However, design ideas such as these appear speculative at present.

We shall now consider some application areas which have received special attention from computer designers resulting in architectures tailored to specific requirements. (Similar activity relating specifically to information retrieval is described in the following section). Early efforts were in the area of signal processing. Bergland (1969) describes and compares a number of hardware modules which perform Fourier Transforms of signal data in real time. He cites an example whereby a general purpose CPU would require one month of continuous processing to analyse signal data which only takes a few hours using a Fast Fourier Transform processor. The hardware tends to be stand alone processors: incoming digitised signals are stored in memory, the processor performs the analysis and stores the resulting Fourier coefficients in memory and outputs them in visual and/or machine readable form. The prime motivation for specialised signal processing hardware is that analysis is normally required in real time. Agrawal and Ninan (1976) address the same problem.

Wesley (1969) describes the use of an associative memory device to carry out Fourier analysis and Mulrooney (1978) describes a minicomputer which has been microprogrammed to carry out fast signal processing.

Watkins (1977) uses the premise that "computing hardware can now be used on a liberal basis, even in a modest system, and the design philosophy has changed from that of bringing all the problems to one large machine to taking a number of machines to the problems", to justify the use of a specially designed multiprocessor system to process radar signals. Processors specially designed for image processing are described by Duff and Watson (1977) and Meilander (1977).

The advantages of designing a computer architecture which meets the processing requirements of a particular high level language has been emphasised on a number of occasions. The MU5 computer system designed and built at the University of Manchester was motivated by a desire to match the architecture of a computer to the Algol 60 programming language (Morris and Ibett 1979). A number of registers were included for example, to make the handling of procedure calls very efficient. The Burroughs B1700 computer was designed with a number of languages in mind. The Burroughs philosophy is described by Wilner (1972). The paper is introduced with an interesting and quite appropriate anecdote:-

> "Procrustes was the ancient Attican male factor who forced wayfarers to lie on an iron bed. He either stretched or cut short each person's legs to fit the bed's length. Finally Procrustes was forced into his own bed by Theseus."

Wilner suggests that Von Neumann derived machines force programmers to lie on many procrustean beds! The final application area which we wish to refer to is that of databases which is, of course, closely related to information retrieval.

One of the largest application areas for computers is databases, and the possibility of using computers with architectures specifically matched to the requirements of database systems is now being extensively explored. Hsiao (1979) describes a database machine as one in which hardware supports basic database management functions found in contemporary software database systems. He argues for implementing basic database management functions in hardware from a reliability and performance point of view. On-line, multiuser systems demand large and complex software systems which tend to be failure prone since complete verification of such systems is not practicable yet. Hardware is less difficult to verify and building functions into hardware leads to less complex and smaller software thus leading to greater reliability. From a performance point of view Hsiao points out the mis-match between conventional Von Neumann type computers and the requirements of database systems. Typically, database management systems become CPU bound with heavy traffic to and from secondary storage. Most attention therefore, has been given

to the application of intelligent storage devices. The computer is still of a Von Neumann architecture but most of the searching, sorting and matching operations are delegated to back end devices. Most designs employ associative memories or logic per track devices. Examples of projects are CASSM (Su *et al*, 1979), RAP (Schuster *et al*, 1979), RARES (Lin *et al*, 1976) and DBC (Banerjee *et al*, 1979). Babb (1979) describes an architecture utilizing CAFS. De Witt (1979) takes a slightly different approach by suggesting the use of a multiprocessor system which can handle a number of queries simultaneously, rather than simply speeding up the processing of single queries, as most back end devices do. Maryanski (1980) presents a survey of activity in this area.

3.4 Specialised Architectures for Information Retrieval

Hollaar (1979a) reviews previous work concerned with the designing and building of specialised processors for information retrieval. The majority of projects have considered hardware which executes functions found in retrieval systems based upon Boolean search logic. An early example is reported by Parhami (1972). The device described uses a fixed head disk with additional hardware which provides basic pattern matching capabilities. Data, stored as character strings, is read from the disk character by character and matched against a search key. The matching process is implemented in the hardware attached to the disk. A marker bit, stored alongside each of the data records on the disk, indicates if a successful match occurred. Once the search and match operation is completed the records which

have their marker bits set are read from the disk. The system, referred to as RAPID, has only appeared in prototype form. Operations that can be conveniently carried out include string matching and evaluation of Boolean expressions involving search keys.

Stellhorn (1974, 1977) proposes hardware for carrying out the union and intersection operations on inverted lists. The lists of document identifiers associated with each search term specified within a Boolean query are read from backing store into a specialised hardware module. The lists are intersected or joined according to the Boolean operators used in the query. Processing time is said to be improved by a factor of between 12 and 60 depending on the capacity of the hardware module. Stellhorn points out that nearly all the mechanised document retrieval systems currently in operation employ inverted files for database organisation and suggests great potential for his system. Hardware designed for the same purpose is also described by Hollaar (1975, 1979b). However, it differs by using a number of specialised processors connected as a network. This allows the evaluation of complicated Boolean expressions without the need to store a number of intermediate results.

Rush (1980) describes a multicomputer configuration for an operational retrieval system (OCLC, Colombus, Ohio, USA). A number of computers (main frames) are dedicated to file management tasks. It is not a specialised computer architecture

as such, although Rush refers to the configuration as a 'database computer'. The retrieval system outgrew the original single CPU mainframe and the adoption of the multicomputer configuration appears to have improved reliability and response time. Radhakrishnan and Rajaraman (1975) propose a similar multicomputer configuration but with specially designed parallel processors for file searching. Their proposal has not yet been realised in practice.

Hardware for matching character strings has also been developed. Mention has already been given to Parhami's work. Haskin(1980) describes a similar device. He points out that the list merging hardware of Stellhorn and Hollaar is inadequate for textual databases unless the files are inverted on every textword. Apart from the impracticalities of doing this, list mergers cannot handle adjacency operators very easily. Haskin also criticises the work done on database computers in general from the point of view of their reliance on fixed field formats and points out that textual data cannot be adequately formatted in this way. Hollaar (1979c) reviews devices which have been proposed for handling textual databases while other papers (Hollaar1978a, Hollaar and Stellhorn 1977) describe Hollaar's own inventions which are based upon logic per track devices.

3.5 A Critique

The work carried out so far which exploits recent developments in computer technology has taken a mechanistic approach. New devices and architectures which have been proposed speed up the processing of Boolean queries or the matching of textual data. However, no one has addressed the question of whether Boolean search logic is the best means of retrieving document references from the point of view of retrieval performance, nor has anyone described how full text retrieval can be used to best effect. It seems that observations have been made of existing information retrieval systems and specific tasks such as list merging or textual string matching have been recognised as being prime candidates for the application of new hardware.

Our criticism should not be too harsh. After all, the question of what *is* the best retrieval mechanism is not an easy one to answer. Some researchers will recommend or express support for one or more different possibilities. However, such recommendations would come from the results of experiments which, as we have already pointed out, are not particularly conclusive. Small scale experimentation with retrieval methods is one thing, their operation in a real environment is quite another. So, before we turn to designing the ultimate information retrieval machine, we should redirect ourselves to the task of establishing which retrieval mechanism should be used as the blueprint for that machine. In other words we need convincing evidence of the value of

retrieval techniques which have been experimented with in the laboratory and which have shown promising results. Only when it has been determined which retrieval mechanism is really worthwhile would we be justified in presenting the specification for an information retrieval computer. Baum and Hsiao (1976) say that the era of database computers is here because of:-

1. the availability of new hardware technology.
2. a better knowledge of the algorithms required to realise database systems.
3. an understanding of data models for database systems.

In the case of information retrieval only 1. above, applies. Points 2. and 3. are still the subject of much debate. Further experimentation with retrieval techniques and the exploration of new ideas is required. New developments in computer technology will yield the greatest benefits to information retrieval if it can assist such research.

Our proposal therefore is to exploit hardware technology for research sake and help to nurture more effective research so that a better understanding of how information retrieval systems should be organised is derived.

3.6 A Route to More Effective Research

We propose to use an inexpensive microcomputer system, rather than ordinary computer terminal, to access an operational

information retrieval system. The configuration is illustrated in figure 3.2. The extra processing power made available in this way can be used to implement experimental retrieval techniques in operational environments. The microprocessors attached to the terminal can process a user's query, automatically request records from the information retrieval system and process those before displaying any to the searcher. Requests for document descriptors can be made using Boolean queries in the format expected by the operational retrieval system. In fact, the retrieval system will detect no difference between the microcomputer system and ordinary computer terminals. An example of a possible use of this arrangement might be to allow the user to submit his query in a format other than a Boolean search statement. He might be allowed to enter a description of his information need in something approaching natural English, say. We explore the range of possibilities in the following chapter.

Assuming that this arrangement allows the implementation of experimental retrieval techniques what then are the advantages? Firstly, the cost of a microcomputer system is not extortionately high. Also, any results of experiments carried out with a retrieval technique using this configuration would be immediately applicable to large operational retrieval systems. This is therefore a source of that 'convincing evidence' (Van Rijsbergen 1979) we are after. System designers may then feel justified in modifying the existing retrieval systems to operate retrieval mechanisms other than the Boolean type. On the other hand, it is quite possible to make software and hardware available which would allow searchers to make use of other retrieval methods without changes being made to existing

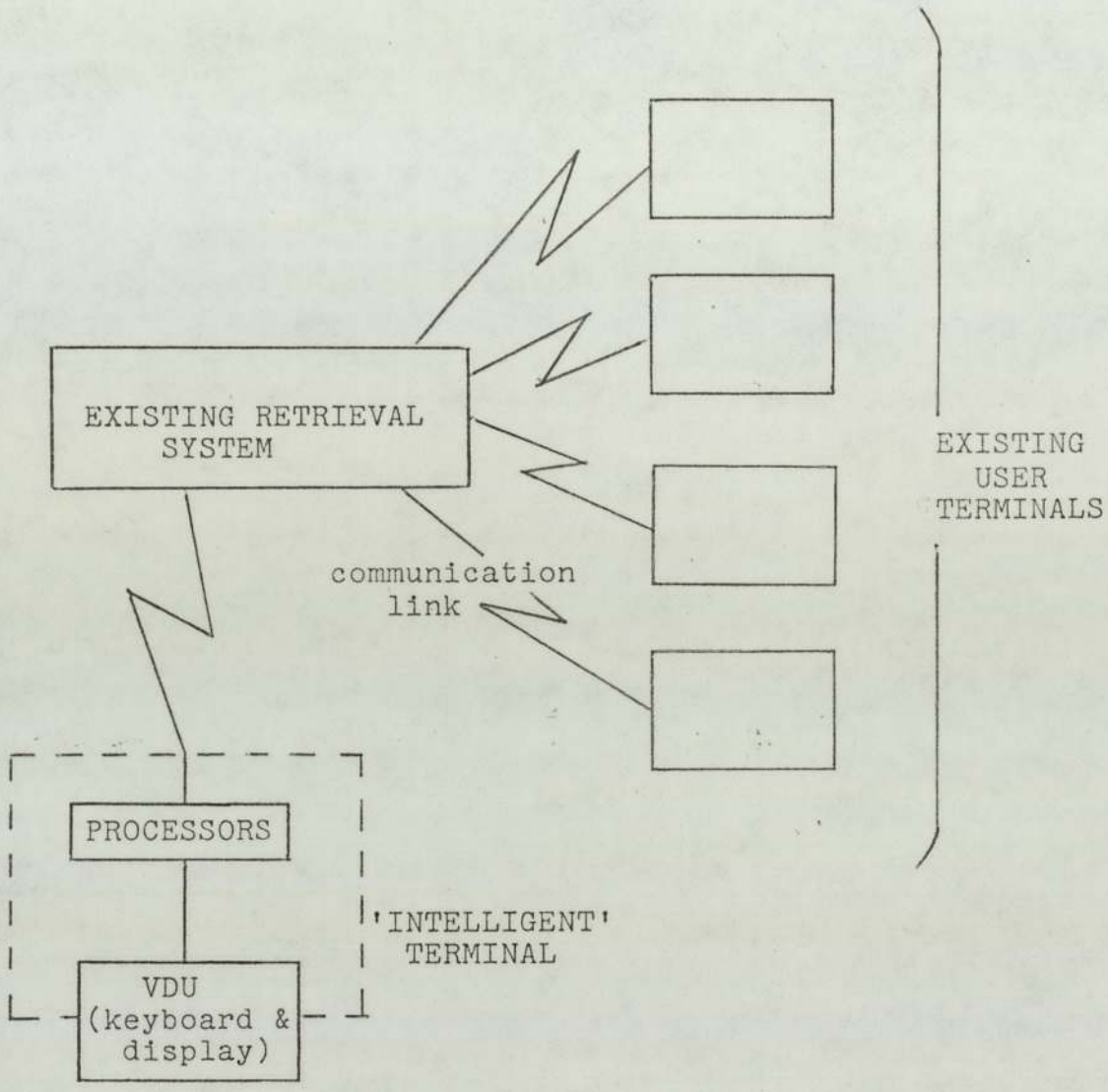


Figure 3.2. The Proposed Machine Configuration.

retrieval systems. Again, the cost would not be particularly high and the investment in the existing retrieval system would not be lost.

The microcomputer system and ordinary computer terminal combined together can be referred to as an 'intelligent terminal'. The operational retrieval system can be referred to as the 'host' system accommodating the 'parasite' retrieval mechanism operating within the terminal. Thornton (1980) has described a similar computer configuration as 'server' and 'worker' computers. In our case the host system would be deemed a 'server' and the intelligent terminal a 'worker'. We prefer the terms host and parasite as the operational retrieval system continues to offer a normal service to users with ordinary computer terminals and is therefore not totally subservient to a number of worker computers.

3.7 Conclusions

Our ultimate aim is to make possible the production of convincing evidence of the value of retrieval methods other than the Boolean type. The exploitation of new technology - microprocessors-will help to achieve that goal. This claim is validated in the following chapters. A review of possible applications for an intelligent terminal configuration illustrated in figure 3.2 is given in chapter 4. Chapter 5 presents a design for such a terminal, while chapters 6 and 7 describe a particular application, the implementation of a retrieval

mechanism based upon term weighting.

The proposed approach also has the potential for alleviating the other two factors which are considered to have induced inertia on the part of the information retrieval entrepreneur. An intelligent terminal will allow retrieval mechanisms other than the Boolean type to be made available to searchers without the need to make major changes to existing retrieval systems. Therefore the investment in existing systems will not need to be forfeited. Also, the operation of the intelligent terminal configuration will be a demonstration in itself of the ability for it to cope with the execution of more complicated retrieval methods. The need for a specialised information retrieval computer will not be such an immediate requirement. These aspects will be considered again in chapter 8.

CHAPTER 4

POTENTIAL APPLICATIONS OF AN INTELLIGENT TERMINAL

4.1 The Meaning of 'Intelligent Terminal'

Before proceeding to discuss possible applications for an intelligent terminal in information retrieval we must be quite clear about what we mean by the somewhat glib phrase 'intelligent terminal'. The term 'intelligent' has already been used to described many computer terminals, particularly by manufacturers in their advertising literature. Close examination of product specifications reveals that an intelligent terminal can range from a relatively simple device which has the capability for handling a number of different communication protocols to devices which offer sophisticated localised editing facilities. Stiefel (1977) makes the following comments:-

> " ..... the term (intelligent) becomes a nightmare to an author who tries to write on the so-called intelligent terminal whose scope is vast. It is bounded on one extreme by tiny 480 character displays that have enhanced editing facilities and are controlled by remote host computers. At the other extreme, it includes huge clustered systems that incorporate powerful central processors, disk storage measured in millions of bytes, and high speed printers."

He suggests that true intelligence comes from user programmability and the provision of peripherals eg, printers, but particularly secondary storage modules such as disks or cassettes. User programmability makes for the handling of new applications and more significantly, a distributed system can be implemented

possibly relieving the central computer of a proportion of its work load. Souster (1977) also distinguishes between those intelligent terminals which have stand-alone processing capability and those in which intelligence is used to facilitate communications. In the following chapter it will be seen that our 'intelligent terminal' is a multi-microcomputer system with local secondary storage in the form of floppy disks.

Thus, we are to consider the potential of a device which is at the far (high) end of the intelligence scale. The following section points to related work with intelligent terminals in application areas other than on-line bibliographic information retrieval systems while section 4.3 concerns itself specifically with the use of such terminals to augment the facilities available from bibliographic systems.

4.2 Related Work

The use of intelligent terminals as data preparation and editing stations within business data processing environments is surveyed by Backler (1976). Facilities provided on many widely available products include usual editing operations (character insertion and deletion etc), protection of fields against overwriting and user definable soft keys. The use of an intelligent terminal for off-loading some of the work from a central computer is outlined by Alsberg *et al* (1976). They describe a 'user-friendly' interface to the main computer system based on menu selection using a touch sensitive screen. Data retrieved from the main computer canbe

reformatted within the terminal before being displayed. The terminal was tried in conjunction with a databank containing social sciences data pertaining to an American State. Numeric data retrieved from the computer can be displayed graphically, in bar chart form for example. Provision is also made for the local storage of data within the terminal as it is retrieved so that the user may view data more than once without the need to refer back to the databank on the main computer.

The emerging home information retrieval systems, viewdata systems, have also been a target for experimentation with intelligent terminals or, more correctly, televisions. Sharpless *et al* (1977) describe such an intelligent television. Their system consists of a television equipped to communicate with Prestel (Fedida and Malik 1980) which contains a microprocessor and local secondary storage in the form of a cassette. It is predicted that such an arrangement could lead to interaction between viewers and television programmes, interaction between households, access to other computers and the availability of 'telenewspapers'. Emphasis is placed on the capability for local recording of viewdata information pages allowing for browsing at a later stage, while disconnected from the viewdata system, thus achieving a reduction in telecommunication costs. However, they fail to mention the usefulness that user programmability of such televisions would be. Volk (1974) reports on another interactive television project based on cable television and suggests similar applications.

4.3 Applications for an Intelligent Terminal in On-Line Information Retrieval

In this section we discuss possible applications for an intelligent terminal in the area of on-line information retrieval. One means of exploitation is in fact chosen for further detailed development and examination, and is the main topic of the remaining chapters of this thesis.

The British Library has already made available a number of intelligent terminals for evaluation as tools in this area. Each unit is a Hewlett Packard 2645A computer terminal which incorporates two cassette decks. Programmability is very limited as is its processing capability. Souster (1977) feels that it is a useful device for providing local storage to enable the recording of search results for example, while Livesy (1978) has made use of it as a stand alone teaching aid. Students of Librarianship and Information Science have been able to use a teaching package which assists them to develop expertise in using on-line information retrieval systems, thus saving the normal cost of using on-line systems. The limited capability of these terminals has stifled any substantial developments, particularly improvements in the performance of on-line retrieval systems, by providing a better user interface, for example.

In a speculative paper Korfhage (1978) attempts to predict the uses of home or personal computers in conjunction with

libraries. He suggests that in the future users with home computers shall be able to access the local library computer and initiate transfers of sections of the library's catalogue for browsing through in the comfort of the home. He also prophesies that increasing public pressure will ensure that the retrieval service becomes much more sophisticated, with the possible provision of full text retrieval.

Often, past studies have been concerned with the use of mini-computers as 'front-ends' to on-line retrieval systems. Now that microcomputer systems are approaching the power of mini-computers many of the conclusions of those earlier projects are applicable to the intelligent terminal approach described here. For example, the application of a network of small computers within the library environment is outlined by Mackinnon and Schuegraf (1975). Their main concern is with 'housekeeping' duties, eg circulation control and cataloguing. Mini-computers within libraries are connected to a large main computer which is shared by many libraries. Initiation of inter-library loan requests is cited as a function of the mini-computers whilst the main computer may produce catalogues. Improvement of search facilities is not considered.

Other related projects are discussed in the sections which follow - each is concerned with one potential application. The possible uses of an intelligent terminal range from those attempting to improve the interface to the retrieval system and the utilities it provides, to those enabling a search strategy to be used which differs from the

standard Boolean search mechanism. We shall consider the complete range of applications here.

We only propose to provide an insight into the numerous possibilities for an intelligent terminal. One idea is taken further and is discussed in more detail in following chapters. The application chosen for further attention is one which allows the operation of a weighted search term retrieval strategy. This choice is motivated by the apparent need (demonstrated in chapter 2) for research on the effectiveness of retrieval mechanisms in realistic environments. This thesis shows how the retrieval strategy can operate on a terminal in conjunction with a conventional on-line retrieval system. Experimentation can then begin to measure the effectiveness of the technique when applied to real queries and a real document collection.

For any application there are two limiting factors which must be taken into account when considering the feasibility of implementation on an intelligent terminal:-

i) The data that can be used is that which is available from the information retrieval system through its command language, and whatever can be elicited from the user at the terminal, such as his query and relevance assessments.

ii) Response time must be short in on-line searching. Thus, due to relatively low transmission speeds across the public data networks at present, the quantity of data

brought into the terminal from the retrieval system must not be excessive.

The second limitation will ease as time progresses. Presently, most information retrieval systems offer transmission speeds up to a maximum of 300 baud (30 characters per second). However 1200 baud lines are becoming available and the future promises greater speeds. Line speeds of 2400 to 48,000 baud are available using packet terminals and networks (Robinson 1976). Kelly (1976) indicates that computers connected to Euronet (Ungerer 1977) will be accessible in this way. Such high transmission speeds are designed primarily for computer to computer data transfers; thus there is no reason why microcomputers should not take advantage of these networks. Our motivation for developing the intelligent terminal is primarily as a research tool. For one-off experiments one can justify a slightly higher expenditure on data communication, arguing that the value of the experimental results (being applicable to an actual operational environment) would prove such expenditure worthwhile. The communication costs can be reduced by placing added processing capability at the same end of the communication channel as the host retrieval system: the user once again has an ordinary VDU (or 'dumb' terminal). The extra intelligence is now in the form of a 'front-end' processor (see figure 4.1). Similar facilities can be made available to users as with the intelligent terminal arrangement although the user has less control over their availability. The front ending processor could also have a parallel hardware interface to the main computer enabling very high data transfer rates (>100K baud) and improved data integrity since it does not travel large distances across a public data network.

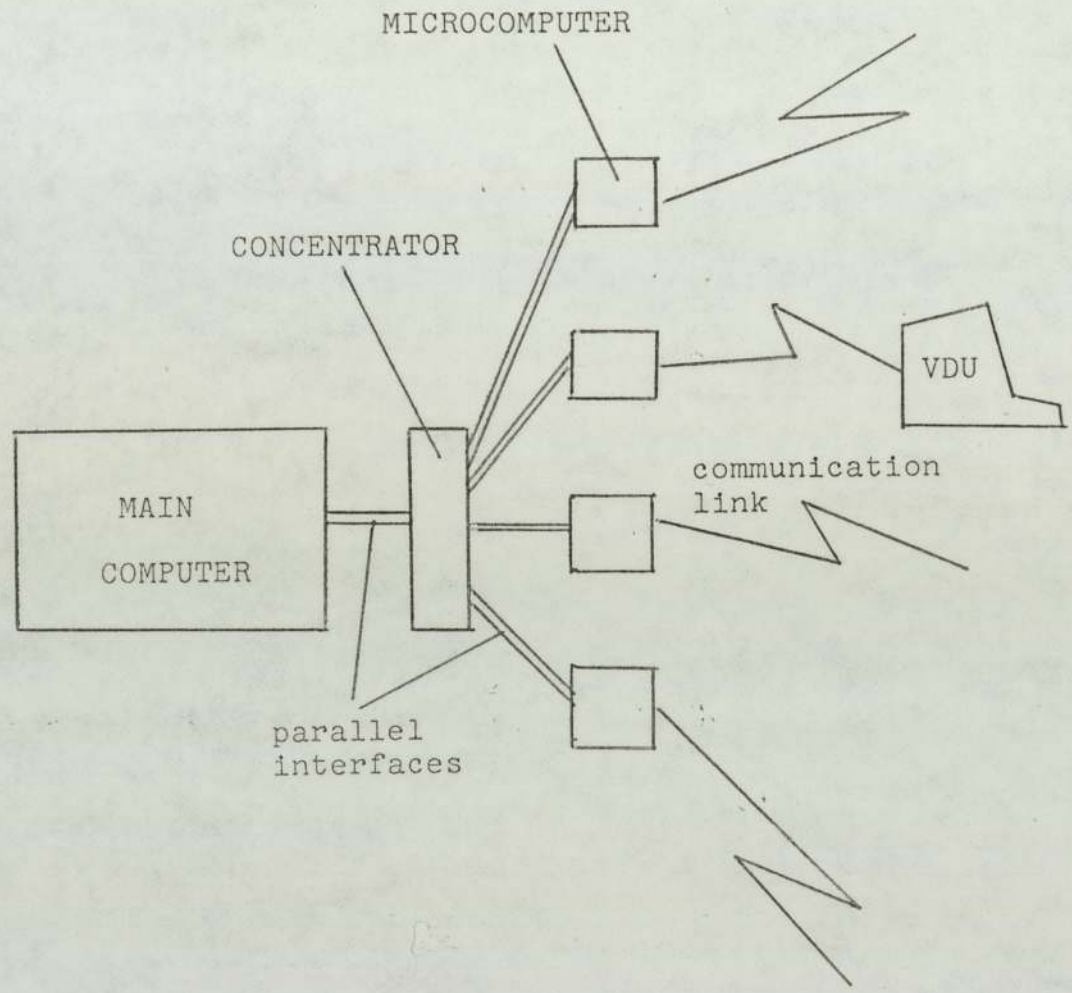


Figure 4.1. A Configuration Using Front-Ending Microcomputers.

We shall now consider specific applications for an intelligent terminal.

4.3.1 User Instruction and Assistance During Searching

Computer aided instruction has been used in many different subject areas and applications. An intelligent terminal would lend itself very conveniently to assisting new users to access on-line retrieval systems or prompting experienced users should they temporarily forget the syntax of, or facilities offered by system commands. While using the terminal to carry out an on-line search the user might request from the terminal information regarding the usage of the command language of the particular host system in use. This type of help facility was provided in the experimental retrieval system developed by Overhage and Reintjes (1974) for the 'Intrex' project: they report:-

> "The most effective user aid and the one we strongly recommend is an on-line instructional approach in which the computer, in effect, takes the user 'by the hand' step-by-step, through the basic procedures required to extract information from the system. As the dialog progresses, the machine advises the operator of his options at each point and the commands that should be entered to exercise them".

User reaction was said to be enthusiastic for this kind of help facility. Cuadra (1971) also expresses the desirability of having built in tutorial facilities within on-line retrieval systems. However, nothing of this nature is available on the

large commercial retrieval systems. Meadow and Toliver (1978) describe how they placed a computer between the user's terminal and an on-line retrieval system. The software within the intermediary computer responded to spelling errors in system commands, incorrect parameters and syntactic errors. The errors were located and reported to the user before the command was transmitted to the information retrieval system, thus removing some of the work load from the main computer. A commercial product called OL'SAM (On Line Search Assistance Machine, Franklin Institute 1980) carries out similar tasks (among others). It consists of a microcomputer and terminal with floppy disks providing backing storage: thus it is effectively an intelligent terminal.

The organisation of software within an intelligent terminal to carry out tasks such as those described here would be relatively simple (see figure 4.2). Each command typed by the user is examined. If it is a help request appropriate messages are displayed on the screen; otherwise it can be assumed that it is intended for the retrieval system to which the terminal is connected. Before transmission is effected the command is parsed and checked for syntactic and semantic correctness. Errors are reported to the user as they are located along with explanatory messages, otherwise the command is forwarded to the retrieval system. Information regarding the command structures for a number of retrieval systems could be stored within the terminal and the appopriate structure selected at the start of a search session according to the retrieval system being accessed.

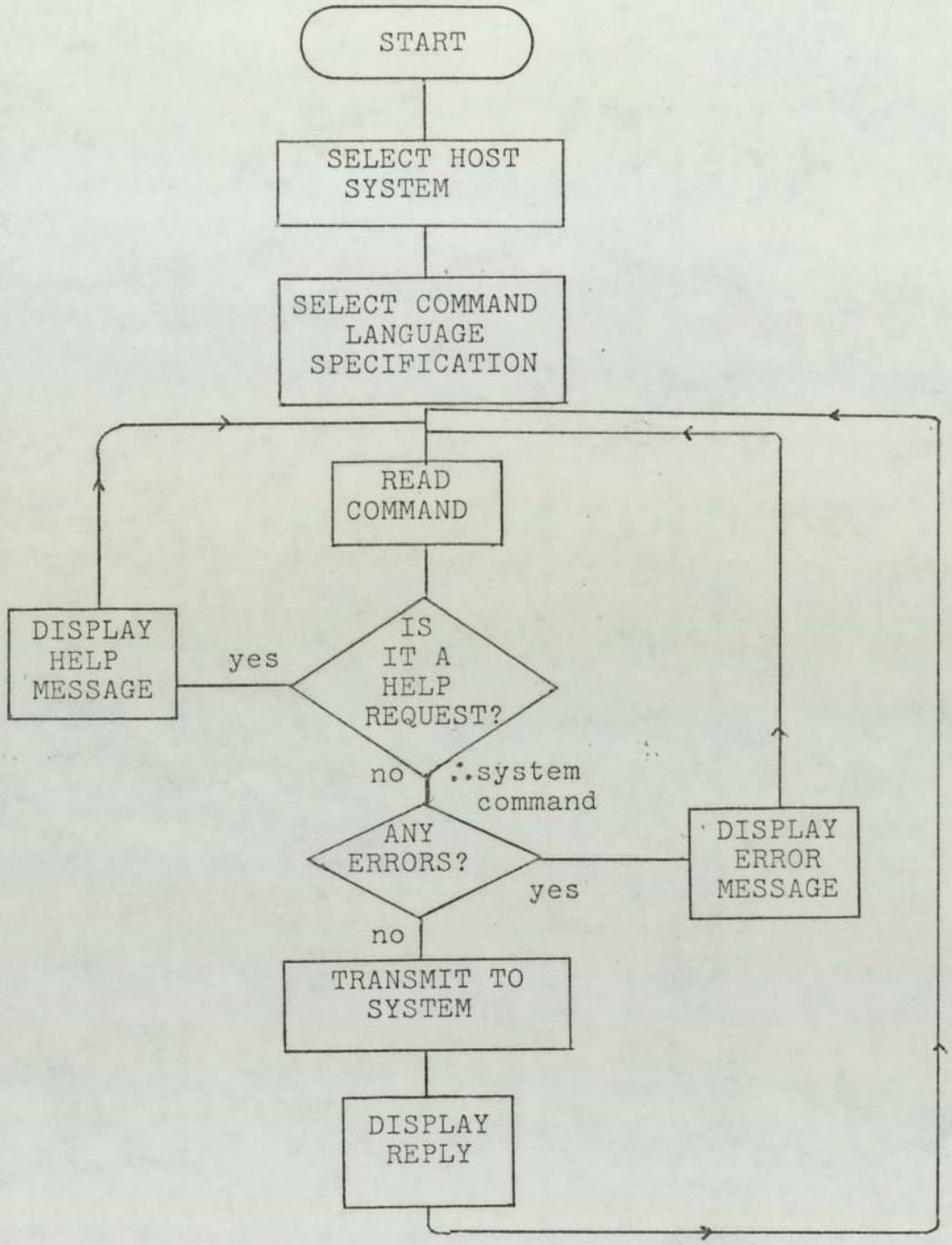


Figure 4.2. The Simple Control Structure of an Error Detection and Help Facility on an Intelligent Terminal.

4.3.2 Improvements to Command Languages

There are many on-line bibliographic information retrieval systems available for public access (see Hall, 1977, for example). Command languages are often peculiar to the system. Thus, users, especially intermediaries who typically run searches on several systems, must be familiar with a number of different command languages. Clearly this causes confusion over matters such as syntax and can lead to inefficient use of the systems. Some command languages are applicable to more than one retrieval system. For example, the British Library's on-line retrieval system (BLAISE) has the same language structure as System Development Corporation's ORBIT retrieval system. This is because the two systems share the same software (ELHILL).

A common format for all retrieval languages would be highly advantageous. Overhage and Reintjes (1974) and Bennett (1972) express the need for a common command language. Cuadra (1971) pointed to the problem caused by a "multitude of systems with a multitude of user interfaces" and posed it as a "national problem of serious proportions". Little has changed since these early papers. Salton (1980) expresses a pessimistic view: "because of the large sums invested in many systems, voluntary agreement on common formats and services is not likely". However, commercial competitiveness has meant that most of the large commercial retrieval systems offer the same kind of services. Once one organisation introduces a new facility the competing organisations follow suit although often referring to the new facility by

a different name. Thus, there is a fairly sound basis for the successful implementation of a common command language.

Work has been carried out in this area already. A set of rubrics for a common command language was suggested by Harley et al (1972) and Atherton (1978) attempts the same in a later and much briefer paper. Negus (1977) reports on a current project concerned with providing a common command or interface language to the many retrieval systems now accessible via Euronet.

Once a suitable command structure which may easily be translated into the majority of existing command languages has been devised then the translation process could be implemented on an intelligent terminal. The software within the terminal would accept the user's search statement and translate it into the format accepted by the retrieval system to which it is connected. The OL'SAM system described above has lim ted translation facilities: it has its own command language which is translated into the command languages of Lockheed's DIALOG system or SDC's ORBIT. Marcus and Reintjes (1981) describe a comprehensive interactive facility called CONIT (Connector for Networked Information Transfer) implemented on the powerful MULTICS computer complex at Massachusetts Institute of Technology (MIT). It allows users to communicate with many different retrieval systems using one command language.

The use of natural language for communicating queries to a retrieval system is an attractive proposition and has been considered on a number of occasions in the past. Lancaster (1971b) claims

"it is far more efficient to have users describe their interests in their own prose narrative terms than to impose the artificial constraints of Boolean forms". However, there are many difficulties associated with natural language processing. The English language is very ambiguous and as yet there is no way of fully analyzing text and representing its meaning in a form which is suitable for processing within a computer. Computer programs have difficulty in coping with poor syntax and making inferences given particular contexts. Bennett (1972) also points out the difficulties in analysing the semantics and syntax of natural language by computer but points to the usefulness of providing a 'natural-like' interface language, ie one that appears natural but still has several format restrictions. Kellog (1968) describes an implementation of this type. The system is essentially a data retrieval system (using census data) and allows for the use of a limited subset of English. Williams (1971) allows for 'free-form' queries in the BROWSER system. The user submits a sentence such as 'I am interested in papers on information retrieval and microprocessors', and the software scans the input, choosing any words with entries in the dictionary, as query terms. Usual conflation rules are applied. Sophisticated text processing was implemented in the LEADER system (Hillman, 1968, Hillman and Kasarda 1969). Both documents and user queries are processed by the same software to produce a representation of their 'meaning', in the form of phrases. Syntax, semantics and logic are all taken into account. Phrases extracted from the user's query are used to select documents (or just pieces of text) from the database.

It is possible for an intelligent terminal to allow the use of a command language which bears a greater resemblance to natural

language than the command languages currently in existence. It is unlikely that anything as powerful as that operating within the LEADERMART system could be achieved as this also requires the document database to have a particular organisation and information content. However, a system which does some kind of keyword extraction from the user's query statement can be envisaged. Keywords can be extracted using usual conflation rules to obtain word stems. The validity of the word stems can be established by ensuring that they exist within the retrieval system's thesaurus.

Goldstein and Ford (1978) report on a project concerned with implementing a new user interface on an intelligent terminal used in conjunction with the American National Library of Medecine's on-line retrieval system. The interface is designed to assist users to produce catalogues using the CATLINE database. The user is prompted for his search specification using a 'fill in the blank' technique. However, no reference is made to an evaluation of the interface.

4.3.3 Use of Multiple Databases

There is much overlap in the subject coverage of the different databases available for searching. Often users may require to search more than one database to ensure a high level of recall or their query may be interdisciplinary and cut across the boundaries of a number of databases. Sometimes it may be

necessary to use different retrieval systems to access all the required databases. Foster *et al* (1977) in a study they made of the need to use multiple databases found that nearly 40% of search requests fall into this class. An example of a common combination of databases is the medical and chemical abstracts databases for searches which require a comprehensive coverage of analysis, stability and manufacture of drugs. To search a number of databases requires users to have knowledge of each database - its organisation, access points and special characteristics. Hawkins (1977b) complains of "the wide variation and great inconsistency in representing the same journal titles, author names .....". Similar complaints are sounded by Artandi (1976). As reported above Salton (1980) expects little to change in the near future. It would therefore be desirable to provide software within an intelligent terminal to assist users in transporting a query from one retrieval system to another or from one database to another within the same system. Assistance could also be provided for users as they decide which databases to use. In one instance a file can be accessed which gives postings figures for every term in each of the different databases that are available on-line. The file is called the MULTIFILE INDEX and is available on the DIALOG retrieval system. An intelligent terminal could use this information to decide which databases to access for a particular query and either advise the user or actually search each of the databases automatically. Williams and Preece (1977) examine the feasibility of such a facility.

Another problem associated with multifile searching is that of retrieving duplicate records. Many documents are of course recorded in more than one database. A search run on a number of

files will retrieve citations for the same document from a number of files which are searched. The terminal could recognise duplicate citations and eliminate them.

4.3.4 Improved System Utilities

An intelligent terminal has the capability to provide a number of facilities which improve the utility of retrieval systems. One possibility is localised offline printing; others include simplified logging on sequences, pre-recording of searches prior to going on line and multiplexing users.

Localised off-line printing will become very attractive once fast line speeds are available. When the user is satisfied with his query formulation the terminal could retrieve the matching document descriptors and hold them in its backing store. Having completed the retrieval the user could log off from the retrieval system and have the terminal produce a hard copy listing of the search results. Opportunity could be given to the user for re-formatting the output and sorting it according to specified fields eg author and date. If a multifile search had been made then duplicates could be removed. Monsen (1977) considers the possibility of localised offline printing using minicomputers. He suggests that user acceptance of the product will be increased by the provision of facilities like these. Retrieving large numbers of document descriptors suggests the possibility of maintaining parts of the database locally, possibly combining the set with the users own personal collection of

bibliographic records. At present it would appear that such action may infringe copyright rules (Holmes 1980) although the true situation is unclear. A small microprocessor system is described by Williams (1980) which is connected between the users computer terminal and the line to the retrieval system. It will log in to a number of retrieval systems on behalf of the user and will store search formulations prior to going on-line, forwarding them to the retrieval system once a successful log on has been achieved. The result is a reduction in cost: charges (telecommunication and system usage) are not incurred while the user is entering his query (and possibly making one or two typographical errors). However, savings of this nature are limited since only the initial query formulation can be entered in this fashion. The user will nearly always wish to modify his query in the light of the results of the first search and, unless the user logs off after the first set of results, such query modification will be done on-line. It is impractical to consider logging on to and off from the retrieval system several times during a search as modifications are made to the query. A facility of this type would be useful for current awareness searches when the user may wish to run the same search periodically. It would then be of value for the user to be able to save his search locally between search sessions. The system described by Williams does not have this kind of storage capability (all changeable store is volatile).

The OL'SAM system already referred to above allows for the multiplexing of users. Two users may run individual searches on the same database through the OL'SAM system and use only one line

into the retrieval system. It appears to the retrieval system that there is only one user. Thus, searching costs can be almost halved as the users will receive two searches for the price of one. However, the drawbacks of this arrangement are that:-

1) at least two users or intermediaries must wish to run searches on the same retrieval system, on the same database at the same time.

2) the financial saving is due only to the current pricing policy of the commercial organisations operating the retrieval systems. Should the policy change from the present cost per unit of time basis to one of cost per retrieved unit say (which is very likely as line speeds increase) then no financial advantage will accrue from the multiplexing technique.

Thus, an intelligent terminal can offer improved utilities. Local off-line printing would appear the most useful but there is scope for saving user's searches from one search session to another. Saving searches over a period of time would be advantageous if a user wished to run the same search on a database available on another retrieval system. Local retention of parts of the database could also be useful but a facility such as this would only be practicable when line speeds are higher and the confusion over copyright rules is cleared up. A facility such as this could assist in the maintenance of catalogues within individual libraries.

4.3.5 Other Methods of Retrieval

Various methods of retrieval, which differ considerably from the standard Boolean search methods, have been devised and experimentally evaluated. An intelligent terminal will allow the operation of some of these search mechanisms in the environment of a real document collection and real users with real queries. In this section we point out the strategies which are amenable to implementation on an intelligent terminal used in conjunction with an operational retrieval system.

4.3.5.1 Weighted Term Retrieval

The idea of attaching a numeric weight to a search term in a query to represent its importance or usefulness relative to other terms is now readily accepted as a valid strategy for document retrieval. Much experimentation has been done with different methods of deriving term weights, and with various techniques for measuring the similarity of document representatives to the query. Although term weighting enjoys such a high degree of popularity there is no general agreement, and no conclusive evidence, as to how term weights should be derived and how the retrieval mechanism should operate in detail. Some weighting schemes are justified theoretically while others appear *ad hoc*. The general pattern for operating such a strategy is as follows; the user submits his query (usually just a list of search terms); weights are assigned to the search terms; document descriptors are matched against the user's

query and assigned a numerical measure of their similarity using some 'similarity function' and finally, the documents are retrieved in decreasing order of their similarity values - the document collection is said to be 'ranked'.

It is quite feasible to consider implementing some kind of weighted term retrieval on an intelligent terminal. It would also appear to be beneficial to the research community to do so. As we have already remarked, the large commercial information retrieval systems have failed to take note of results of research and, although weighted term retrieval has been widely experimented with, none of the large commercial organisations have implemented equivalent search strategies on the public bibliographic systems. However, not all weighting schemes devised so far are amenable to implementation in this way. Sparck-Jones (1973a) has provided us with a useful classification scheme for the numerous methods of term weighting which have been devised to date. There are two main classes:-

1) subjective weighting schemes,
2) statistically derived weights - objective weighting schemes.

The first class includes strategies which rely on the user to submit judgements about the importance of search terms. For example, he may be required to give a score out of ten to each term in his query: terms that he believes important are given a high score. A recent paper reporting experiments with such a weighting scheme is Salton and Waldstein (1978). Users ranked their query terms in decreasing order of importance, and the software converted rank positions into term weights. Experimental results with 41 test queries indicate that

such manually determined weights lead to inferior performance compared to statistically derived weights (class 2 above). However a combination of both was superior to all. One can easily consider implementing a subjective weighting scheme of this kind on an intelligent terminal.

Referring back to Sparck-Jones' classification scheme the second class itself can also be divided into two groups - intrinsic and extrinsic weighting schemes. An intrinsic weighting scheme takes account of the number of occurrences of a term within a particular document. This approach leads to a term having a different weight for different documents. Alternatively a weighting scheme may take account of statistical information regarding the distribution of terms throughout the document collection, ie the term postings. Such a weighting scheme is referred to as extrinsic.

Intrinsic weighting or 'within document ' weighting can be traced back to original ideas by Luhn (1958) who, concerning himself with automatic indexing techniques, proposed that the frequency of word occurrence in an article gives a useful measurement of the word's significance in representing the information content of the document. Taking the Zipfian distribution of words within a text he suggested that significant words were those lying within the middle area of the curve (see figure 4.3). Words which occur too frequently are considered too common, consisting mainly of prepositions, definite and indefinite articles etc. Terms which occur very rarely are

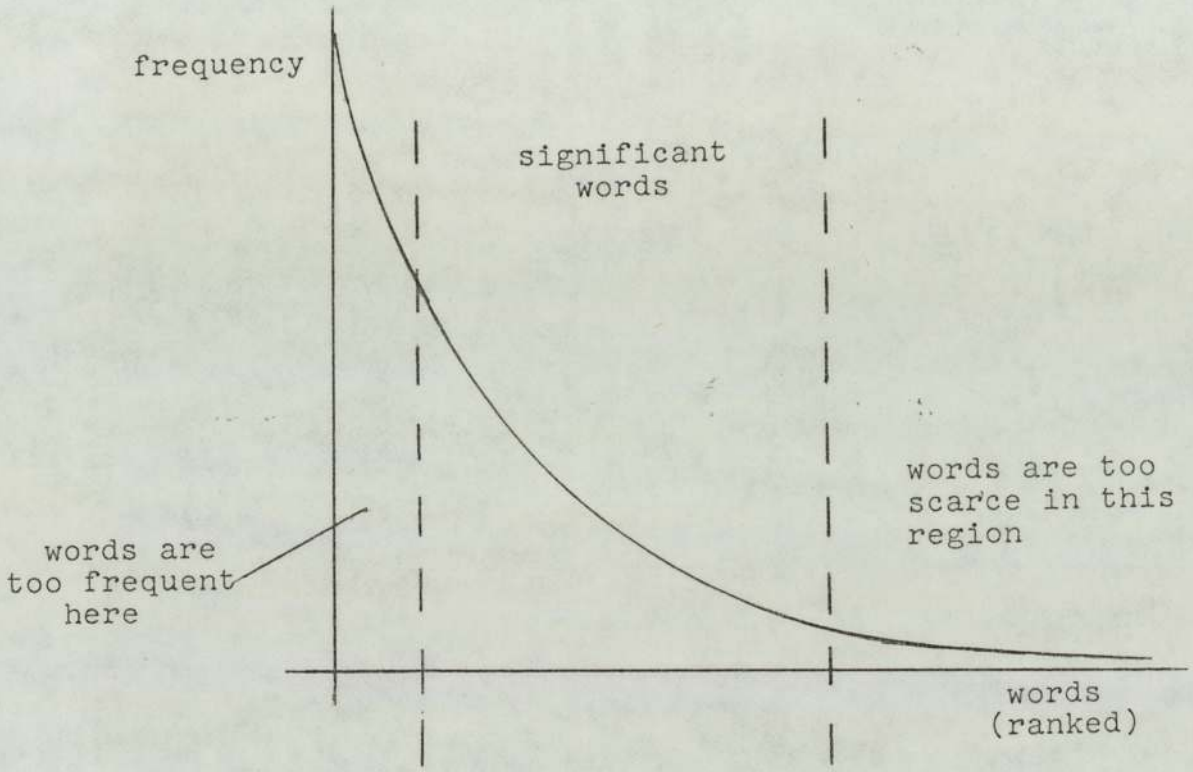


Figure 4.3. The Zipfian Distribution of Text Words and its Implications for Automatic Indexing

considered not to contribute significantly to the information content of the article. Techniques for automatic indexing along these lines have been intensely investigated within the SMART project (Salton 1971).

Substantial evidence supporting extrinsic weighting strategies is reported by Sparck-Jones (1972). In that paper she argues for term weights to be based on collection frequency in such a way that matching on less frequent, more specific terms is of greater value than matching on frequent terms. The weighting function used for individual terms is:

$$\log \frac{N}{n} + 1$$

where N is the number of documents in the collection, and n is the number of postings associated with a particular term. In fact, this strategy is giving high weights to specific terms (in other words it is providing a "statistical interpretation of term specificity"). The function used in the experiments to ascertain a measure of the similarity between the queries and document representatives is

$$S_d = \sum_{t \in (D_d \cap T)} w_t$$

where D_d is the set of terms indexing document d,
T is the set of terms representing the query,
and w_t is the weight associated with term t.

Experimental results (Sparck-Jones 1972 and Salton and Yang 1973) show such a weighted retrieval scheme to be worthwhile.

When considering which weighting schemes might be implemented on an intelligent terminal, one must take into account what information is available from the commercial information retrieval systems. Postings figures are readily available as are the lists of index terms assigned to individual documents: however, information regarding the distribution of indexing terms within full document texts is not available. Therefore it is feasible to implement a term weighting scheme based on collection frequency weights but not one based on the frequency of occurence of terms within documents. In general we conclude that extrinsic weighting schemes are candidates for implementation but not intrinsic schemes. However, we should also note that many retrieval systems make abstracts of documents available on-line, the processing of which within the terminal could lead to the realisation of some intrinsic weighting scheme on a limited basis. It may b possible to retrieve a subset of the document database via an extrinsic weighting scheme and operate an intrinsic weighting strategy using the abstracts of the retrieved set as document surrogates.

We have not discussed the difficulty of matching the query against all the documents in the database. The terminal does not have direct access to the database and it would be ridiculous to propose that all document descriptors are transmitted to the terminal for matching against the query. However a search technique has been developed by the present author which overcomes this problem. If we make a suitable choice of matching function

the intelligent terminal can retrieve document descriptors from the host retrieval system in the correct ranked order automatically, by formulating a series of queries consisting of different combinations of the user's search terms. The technique is discussed in detail in the following chapter.

4.3.5.2 Feedback Mechanisms

An extension to weighting schemes which has received attention is that of having a mechanism which changes a query automatically during a search session in the light of judgements made by the user regarding the relevance of documents retrieved from the database. As document descriptors are retrieved and displayed to the user he indicates whether they are relevant or not. Usually, the retrieval program increases the weights attached to query terms which occur in relevant documents and decreases weights of terms which occur in non-relevant documents. Terms may also be added to, or removed from, the query. Barker *et al* (1972), Vernimb (1977), Salton (1971) and Robertson and Sparck-Jones (1976) have experimented with feedback techniques. Most experimental results have supported the use of 'relevance weighting' methods (as feedback techniques are often referred by), eg Sparck-Jones (1979 and 1980). An intelligent terminal can very easily conduct a dialogue with the searcher to establish the relevance of documents whose descriptors are retrieved from the host retrieval system, and update the query term weights according to the replies given by the user.

An encouraging aspect of the work done by Robertson and Sparck-Jones is their attempt to justify the choice of formula for calculating term weights theoretically. They begin by making some simplifying assumptions regarding the distribution of index terms in relevant and non-relevant documents. The results of their theoretical work does therefore rest upon the validity of such assumptions. In particular, the validity of one assumption they make, 'term independence', has been questioned. They propose that "the distribution of terms in relevant documents is independent and their distribution in non-relevant documents independent". Van Rijsbergen and Harper (1978) support the view that there is dependence between terms and extend Robertson and Sparck-Jones' theoretical work to take account of this. Term dependence can be related to co-occurence data, ie the number of documents jointly indexed by a pair of terms, and a weighting scheme is developed to take account of this. Co-occurence data can be extracted from existing commercial retrieval systems by simply submitting a Boolean query consisting of the conjunction of two index terms. Thus, one can consider an implementation of such a weighting scheme on an intelligent terminal. However, because the number of pairs of index terms is large ('the curse of dimensionality' (Van Rijsbergen, 1977)), it is unlikely that such an implementation is practicable in its pure form without seriously reducing the size of the database operated upon, by some initial search using a different retrieval strategy. Nevertheless, the data required for the operation of quite complicated weighting schemes is available. Increased line speeds or the addition of true 'front ending' processors alongside the host retrieval system may make for a more practicable implementation.

4.3.5.3 Post-retrieval Clustering

Usually clustering is thought of as a method of organising a complete document database. Document descriptors are grouped together or 'clustered' according to their similarity with one another. Descriptors, similar in form to those attached to documents, are assigned to clusters Such cluster representatives are a selective amalgamation of keywords taken from the document descriptors contained within the cluster. Retrieval is carried out by first matching the user's query against the cluster representatives rather than individual document descriptors. Clusters are chosen according to the similarity of their representatives to the query and either all the document descriptors contained within the clusters are retrieved or further matching is carried out between the query and the document descriptors. One justification for clustering a document collection is that then searching is fast because most document descriptors are eliminated by the initial selection of clusters. However, clustering a document database is expensive in terms of machine resources and database updates lead to periodic re-clustering. Work has been carried out on fast clustering algorithms eg Croft (1977). Van Rijsbergen (1979) provides an overview of clustering theory and techniques.

Preece (1973) has proposed the use of clustering in conjunction with systems based on Boolean searching. He suggests that the output generated by a straightforward Boolean search should be clustered. He argues that relevant items will tend to be grouped

together in the same clusters and the user may scan a list of cluster representatives to identify relevant material rather than a much longer list of document descriptors. The initial search can be viewed as a filter to eliminate the majority of non-relevant items in the database. Less machine effort is required because many fewer documents are involved in the clustering process, and less user effort is required in scanning the shorter list of cluster representatives to identify relevant items.

Although such reasoning sounds plausible, Preece does not offer any experimental evidence with regard to the performance of the scheme as a retrieval tool. Such a methodology could be implemented on an intelligent terminal. The user enters his Boolean query and the terminal can capture the resulting document descriptors. Clustering of those representatives then takes place within the terminal.

Attar and Fraenkel (1977) put forward similar ideas. They also performed some experiments which indicated that better retrieval ensues from post-retrieval clustering compared to initial global clustering of the complete database. However the experiments they performed were quite limited in scope. The test database consisted of 76 patent descriptors and they used only seven queries.

4.3.5.4 Browsing Mechanisms

A method of searching commonly practiced by library users involves simply looking along bookshelves or through catalogues without a well defined specification of information need in mind. Although the searcher cannot define exactly what he wants at the outset he will nevertheless know once he finds it. 'Browsing' searches of this kind are not uncommon. Lancaster (1968) recognised their existence and pointed out the lack of opportunity to carry out searches such as these using computerised information retrieval systems. Although at the time of writing he was only able to consider the early batch retrieval systems which existed then, similar observations can be made of present day on-line systems. We have already pointed out that 'modern' retrieval systems operate traditional Boolean search strategies and therefore require a precise statement of information need from the user in the form of a Boolean query. The searcher who wishes to browse and has an information need which is ill-defined will experience difficulty in constructing a suitable query. On-line systems give the user the opportunity to modify a query in the light of search results. This is one of their main advantages over the old batch systems but it does not assist the user who has difficulty in precisely defining what he requires.

Oddy (1974 and 1977) has also considered browsing as a legitimate form of searching. He devised a retrieval mechanism which supports searches such as these. The retrieval program does not expect a query statement as such from the user, instead man and machine enter into a

dialogue during which the computer attempts to develop a representation of the searcher's information need. The dialogue begins with the user entering keywords which he feels are in some way concerned with his area of interest. The computer displays document references and their associated index words. The user indicates whether such documents appear relevant and can single out individual index terms as being connected with his information need. The representation of his information need within the computer is based on a network model. The complete document database is arranged as a network: author's names, documents and keywords form the nodes of the network and the links between nodes indicate associations between documents and authors or documents and keywords. The user's information need is represented by a small part of the complete network. As the user indicates relevant terms or documents the appropriate nodes are selected from the main database and added to the model of his information need. Nodes which are close neighbours to those cited by him are also added to his model. Documents from the sub network are displayed to the searcher. The choice as to the order in which documents are displayed depends upon the connectivity within the sub network. Thus, a representation of the user's information need is developed without placing the burden of having to formulate a precise query statement, upon the user. Experimentation performed by Oddy showed that retrieval performance was equivalent to more conventional information retrieval systems although requiring less effort on the part of the user.

Having an intelligent terminal operate a browsing mechanism such as that described above and carrying out a large scale evaluation of the

strategy is an attractive proposition. The terminal can be programmed to conduct a suitable dialogue with the user and build up a model of his information need. Documents can be retrieved from the host system by the terminal. Although the databases on existing retrieval systems are not explicitly arranged as networks there is an implicit structure. Associations between documents and terms or authors and documents, ie the links between the nodes in the network, may be established with suitably constructed Boolean queries. The terminal can do this automatically. Scale again presents problems. The number of documents associated with one term for example can be large (>10,000) in existing on-line document databases. There will therefore be a need to limit the amount of data required by the terminal to operate the retrieval strategy. Oddy has pointed out that highly posted terms could be ignored and only low and medium posted terms taken into account. The strategy could also be confined to a small section of the database by carrying out an initial broad Boolean search thus leading to a form of 'post-retrieval browsing'.

4.3.6 System Monitoring

A number of authors have pointed out the usefulness of simply monitoring the use made of retrieval systems by recording the complete dialogues between user and system. Mittman and Dominick (1973) list a number of parameters which may easily be measured

such as elapsed time per session, frequency of user errors and their type and number of search terms per query. Making measurements of this kind may lead to an improved user-interface, reducing the likelihood of errors, or to more efficient system operation by optimising commonly invoked functions. In a later paper Dominick and Penniman (1979) again emphasise the value of recording patterns of system usage indicating that statistical analysis can assist designers and managers of retrieval systems to make decisions relating to system enhancement and improvement. Salton (1970) says that evaluation of retrieval systems should not only take into account recall and precision results but also cost related parameters such as machine effort and user effort. On-line recording of dialogues with appropriate post search processing can make the measurement of such parameters possible.

An intelligent terminal can very easily make complete records of the interaction between the user and retrieval system. Each keystroke made by the user and each character received from the retrieval system for display can be stored on the secondary storage module - disk or cassette. The processing of the dialogues could be carried out within the terminal after the retrieval session or the recorded data could be transferred to a main frame for textual and statistical analysis. The availability of statistical analysis software packages possibly makes the latter strategy more appealing.

4.4 Conclusions

In this chapter we have surveyed a large number of possible applications for an intelligent terminal in information retrieval. Some of the possible uses have recently been put into practice by one or two commercial ventures as we pointed out in the individual sections. However, only those which make existing retrieval systems more convenient to use or more cost effective have been tackled. There has been no activity directed towards altering the type of seach process carried out. Such applications as these would prove extremely beneficial to the research community. A facility for testing retrieval strategies with real document collections in a real user environment appears to be extremely attractive.

In the following chapter we describe an architecture for an intelligent terminal. Our prime motivation is to enable the implementation and evaluation of retrieval strategies. We chose one retrieval method for detailed study and describe its implementation on the terminal in chapters 6 and 7.

CHAPTER 5

THE INTELLIGENT TERMINAL

In this chapter we present a full description of the intelligent terminal designed and built by the author. Firstly, the logical structure is described with particular reference to the arguments underlying the design. We continue with a detailed account of the hardware structure and conclude with some general comments about the basic system software.

5.1 Design Philosophy

The terminal communicates with a user and with the 'host' retrieval system - an existing information retrieval service. It is equipped with software to support the operation of a 'parasitic' retrieval mechanism. Communication with the host system is such that use is made of the command language appropriate to that particular retrieval system, i.e. the language normally employed by the system's subscribers for searching the document database. No other means of communication is available to us: thus the host system detects no differences between our intelligent terminal and other terminals (visual display units and teletypewriters) used by other system users. The work required of the terminal can be conveniently split into three logically separate tasks:-

i) communicate with the user.

ii) operate the parasite retrieval mechanism.

iii) communicate with the host system.

This leads naturally to a modular structure for the terminal - figure 5.1. Such a logical structure is advantageous in that each of the three

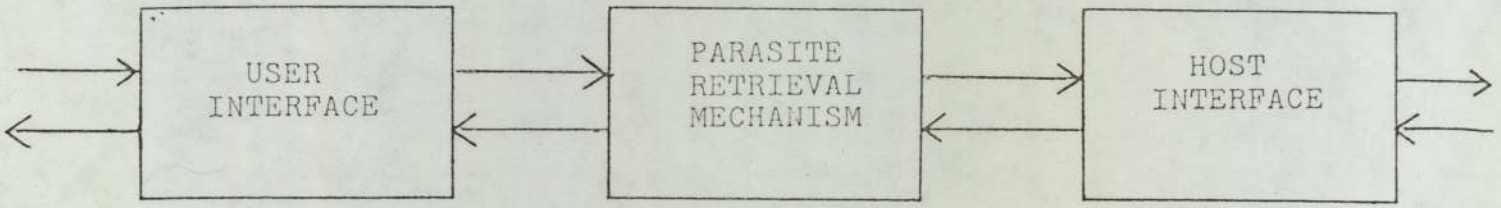


Figure 5.1. A Modular Organisation for the Intelligent Terminal.

major tasks listed above are quite separate and the design and implementation of each module could proceed independently once suitable interfaces between them had been designed.

At the core of the system is a central module which operates the parasite retrieval mechanism. Requests for information it requires from the host or user are made to either of the other modules as appropriate. Such requests may be suitably succinct for they are reformatted before being relayed to user or host. The user interface module expands a request, incorporating a degree of 'user friendliness', while the host interface structures the request according to the format of commands recognised by the host system. Input from the user, his query statement for example, and replies from the host, postings information say, are stripped of 'fluff' material by the interfacing modules and relayed to the central module in a very concise format. The central module is in fact, totally independent of protocols for communicating with the host and user. Thus, to enable the terminal to accept a different retrieval service as the host system, changes need only be made to the software of the host interface module. Similarly, modifications made to the user's command language affect only the user interface module. One might for instance base the interface on a language other than English.

Similar arguments have been used to justify the use of intermediate languages for assisting the implementation of high level languages on computers. Conventionally, a compiler accepts a program written in a high level language and produces a machine code version for some specific computer. However, when an intermediate language is used

compilation is carried out in two stages. Firstly the high level source program is processed to produce a representation in an intermediate language. The intermediate form is then translated into the appropriate machine code for a particular machine. Capon *et al* (1972) describe such a language - CTL - which has been used in the implementation of Algol 60, Fortran and PL/1 on the MU5 computer. Similarly Richards (1971) describes OCODE used in the implementation of BCPL. Elsworth (1978) points out two main reasons for proceeding in this way:-

> "i) A large and complex task (the construction of a compiler) can be split into two smaller and hopefully, more tractable problems; furthermore, given the (intermediate language) interface, these problems will be logically independent.
>
> ii) The portability of language systems can be enhanced; to implement a new language on a machine, or a language on a new machine it may be necessary to re-write only one section of the two stage compilation process."

These considerations apply equally well to the modular design of the intelligent terminal described here. Logical independence of each of the modules shown in figure 5.1 leads to a high degree of flexibility. One module may be changed with minimal repercussions for the other two. Therefore, adapting the terminal for use with other host retrieval systems is relatively easy - only the module interfacing the terminal to the host need be changed. Also, the terminal may operate various parasite retrieval mechanisms or allow the user differing command structures and only one module needs to be modified each time.

5.2 Hardware

5.2.1 Overall Structure

The terminal utilises the cheap processing power made available by microprocessors. At the time that the terminal was designed (1977) the most widely available machines were eight bit systems i.e. eight bit data bus and sixteen bit address bus. By far the most popular, and best supported chips were the Intel 8080 (Intel 1979), Motorola M6800 (Motorola 1975c) and Zilog Z80 (Zilog 1978), all of which are approximately equivalent in terms of processing power. The Motorola M6800 was used in this work because facilities for developing software and hardware based on the M6800 were more readily available. It is only very recently that more powerful machines have become available, e.g. the Motorola M68000 and Zilog Z8000. These have 16 bit data buses, 32 bit address buses and enhanced instruction sets. The Motorola company now produce an enhanced M6800 chip named MC6809 (Motorola 1979a and 1979b). This machine has 16 bit internal registers and a much improved instruction set compared to the M6800. It can be considered to be a hybrid chip situated between eight and sixteen bit products. The M6800 instruction set is compatible with the MC6809 instruction set but not with the M68000.

Although microprocessors do offer processing power on a very low cost basis they nevertheless have limited capabilities compared to standard mini-computers or mainframes. In order to overcome the limitations the present design employs a processor for each of the three logical modules described in the previous section. The overall structure of the terminal is outlined in Figure 5.2 and this should

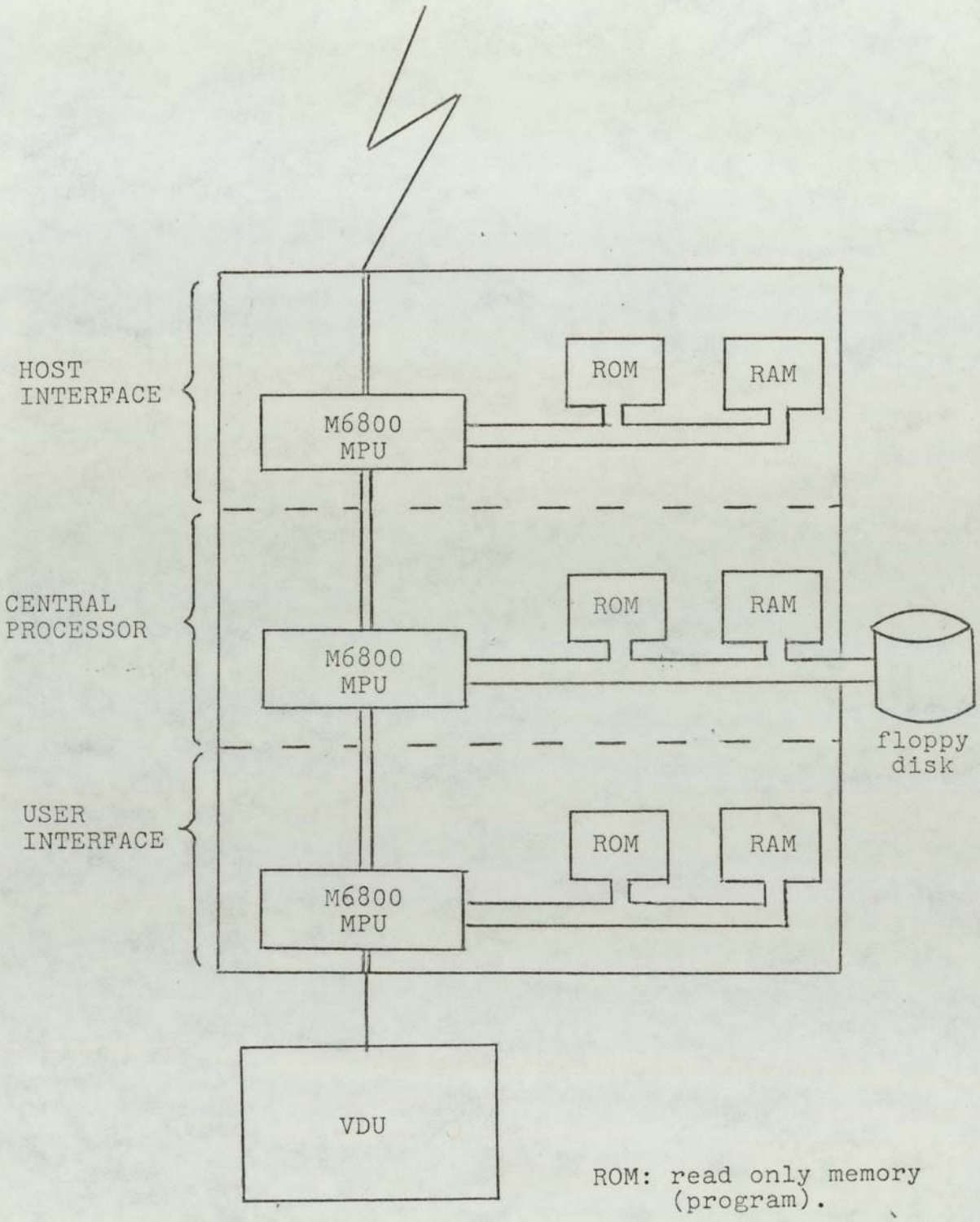


Figure 5.2. The Structure of the Intelligent Terminal.

be compared to figure 5.1. The processors are connected as shown. Mass storage takes the form of a dual drive floppy disk unit offering 500,000 bytes of storage. This unit is under the sole control of the central processor which actually operates the parasite retrieval mechanism. Each processor operates independently hence providing the capability for parallel processing. Communication between the processors is via eight bit bidirectional data buses. A simple message structure is used which is described in section 5.2.3.1. The processors are interrupt driven acting on the messages received from their respective neighbours as required. An alternative means of communication that is commonly used between processors is via common memory areas. However such a scheme introduces many problems; a more complicated hardware structure with bus and memory contentions to be considered. Designs for the terminal based on such an organisation are therefore disregarded in favour of that illustrated in figure 5.2.

The resulting configuration does not fit the generally accepted definition of a multi-processor system. In his survey article, Enslow (1977) presents the ANSI definition of a multi-processor system, "a computer employing two or more processing units under integrated control". Thus, by implication, a multi-processor system must have a single, integrated operating system. The terminal configuration is not under the control of one integrated operating system. Enslow argues for extending the ANSI definition to include the following, "a multi-processor must have the capability for the direct sharing of main memory by all processors (arithmetic/logic unit and control unit only) and the sharing of input/output devices by all memory and processor combinations". Again, the terminal structure does not fit

the bill. The processors are very loosely coupled with very limited degree of sharing. In fact, the method of communication chosen, the simple sequence of messages, would allow geographical separation of the individual microprocessors if suitable serial interfaces were attached. This is obviously not desirable and in no way would it prove to be advantageous to do so but, it is nevertheless possible. Thus the terminal is merely based upon distributed computing principles, i.e. it is a multi-computer system. Enslow supports this view - "a multi-computer system consists of several separate and discrete computers (even though there may be direct communication between them), ..."

5.2.2 The Processors

A microcomputer system manufactured by Motorola Ltd to aid development of hardware and software for M6800 systems was available for this work. Called an 'EXORciser' (Motorola 1975a), it is itself based on an M6800 processor and includes built in diagnostic and utility programs which assist system development. Mass storage is available in the form of a dual drive floppy disk unit. A disk operating system provides the usual facilities for creating, listing and editing files with further commands for assembling, loading and running programs. Normal practice is for users to develop programs on the EXORciser using its facilities for editing and debugging. When the user is satisfied that his software operates correctly he transfers it to his own hardware. The transfer is generally achieved by programming PROMs (Programmable Read Only Memories) using another EXORciser facility. The EXORciser is able to execute user programs itself.

The terminal described here makes use of three processor modules. Following normal practice, these three hardware modules would be constructed, each with one or more PROMs prepared on the EXORciser, containing their various programs. The EXORciser incorporates facilities (particularly the disk controller) which are useful for the central processor in the terminal design, and in the prototype terminal it is used to fulfill that role. The other two processor modules - the user and host interfaces are implemented in specially constructed hardware.

The two interface modules have very similar tasks, translating between different command formats, so both modules were given the same hardware structure. Each processor module is a self contained microcomputer system. All the integrated circuit chips associated with one module are contained on one circuit board and are linked together by means of wire-wrapped connections. Each board consists of:-

up to 1024 bytes of random access memory (RAM),

up to 4096 bytes of read only memory (ROM),

1 serial input/output port,

2 parallel input/output ports,

1 M6800 microprocessor,

data and address line buffers/drivers,

partial address decoding.

Inter-processor communication is achieved via the parallel input/output ports. These ports allow bit parallel transmission of eight bit bytes and provide handshake control facilities.

Further discussion of the means of communication between processors is given in the following section. More technical detail of the composition of the processor boards is presented in appendix 1. The user communicates with the terminal system using an ordinary VDU. The hardware appears as a 'black box' in between the VDU and the information retrieval system.

5.2.3 Inter-Processor Communication

The processors are connected according to the terminal structure illustrated in figure 5.2. The following two sub-sections are concerned with the means of communication between the processors. The first section describes the message structure used by the modules to relay information to one another while the second section describes how a message is physically transferred between processors.

5.2.3.1 Message Structures

Communication between processors is based upon a series of messages. The structure of the messages is simple: the first byte indicates the message type, the following bytes contain the actual information being relayed and an EOT (End Of Transmission) character terminates it - figure 5.3. The information contained therein can be numeric e.g. the number of documents indexed by a particular term, or a text string such as a query term. In a few cases the body of the message is null because the message meaning (or information content) is implicit in the message 'header', the first and only byte. For example, such a message is presently used by the host interface to indicate to the central processor that no documents were retrieved

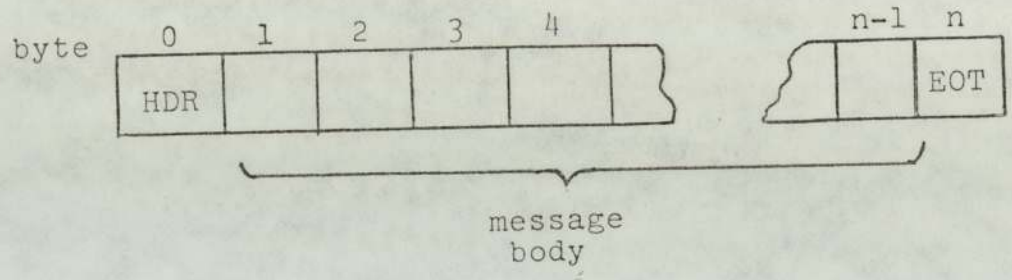


Figure 5.3 Structure of Inter-Processor Messages.

for the last search statement sent to the host retrieval system. The message types recognised by the individual processor modules are listed in appendix 2. More details regarding the information content and structure of the messages is given in the chapters which elaborate on the terminal's software.

5.2.3.2 Physical Communication Technique

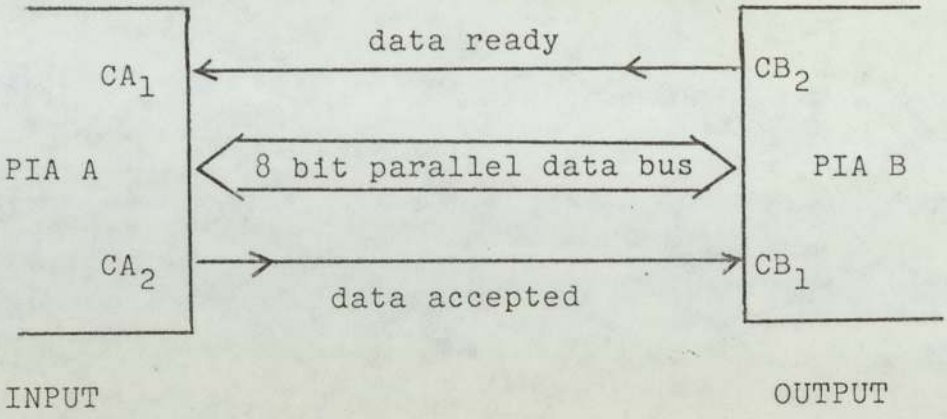
The physical transfer of data internally, between processors, and externally, to and from the user and host system, is achieved by utilizing integrated circuits produced by Motorola Ltd specifically for input and output tasks.

Use is made of the Peripheral Interface Adapter (PIA) chip for bit parallel (byte serial) communication between processors. Although a PIA is quite complex internally, it allows the system designer to handle data input and output very easily with simple programming techniques. A similar chip is used where serial input and output is required - the Asynchronous Communications Interface Adapter (ACIA). Serial input and output is required from and to the host retrieval system and the user's VDU. In both cases a small amount of additional circuitry is required to transform signals generated by the ACIA to the standard levels produced and accepted by VDUs.

Both PIAs and ACIAs are attached to the system data buses. Each chip contains registers which may be written to or read from. A microprocessor can access these registers in the same way that it accesses any memory location. Selection of the

interface chips is achieved by address decoding as with the memory chips within the system. The interface chips are said to be mapped into the microprocessor's address space. The chips contain status and control registers as well as the data registers. Status and control bits indicate, among other things, whether data is ready for reading or if the output port is ready to accept data. A PIA also allows for sophisticated 'handshaking' protocols to be implemented for controlling data transfer.

For each eight bit input/output port provided by a PIA two control lines are available for use - C_1 and C_2. Within our terminal one line is used as an output and one as an input. Figure 5.4 illustrates how two PIAs are configured for transferring data within the terminal. Although simplified to a certain degree it serves to illustrate how the handshaking protocol works. Each PIA has one output control line, C_2 and one input control line, C_1. When a PIA is programmed to output data the output control signal becomes a 'data ready signal' and the input control line becomes a 'data accepted' signal. For a PIA port programmed to receive data the roles are reversed - the input control line is 'data ready' and the output is 'data accepted'. For a transfer of one eight bit byte from board B to board A in figure 5.4 the sequence of events is as follows. Microprocessor B checks the status of PIA B. If the previous byte of data has been transferred to board A then microprocessor B writes a byte of data into the data register of PIA B. This action causes signal CB_2 to change level which is detected by PIA A on CA_1. The change of level on input CA_1 registers within PIA A, a status flag is set and an interrupt is caused in the operation of microprocessor A. Microprocessor A reads the status of



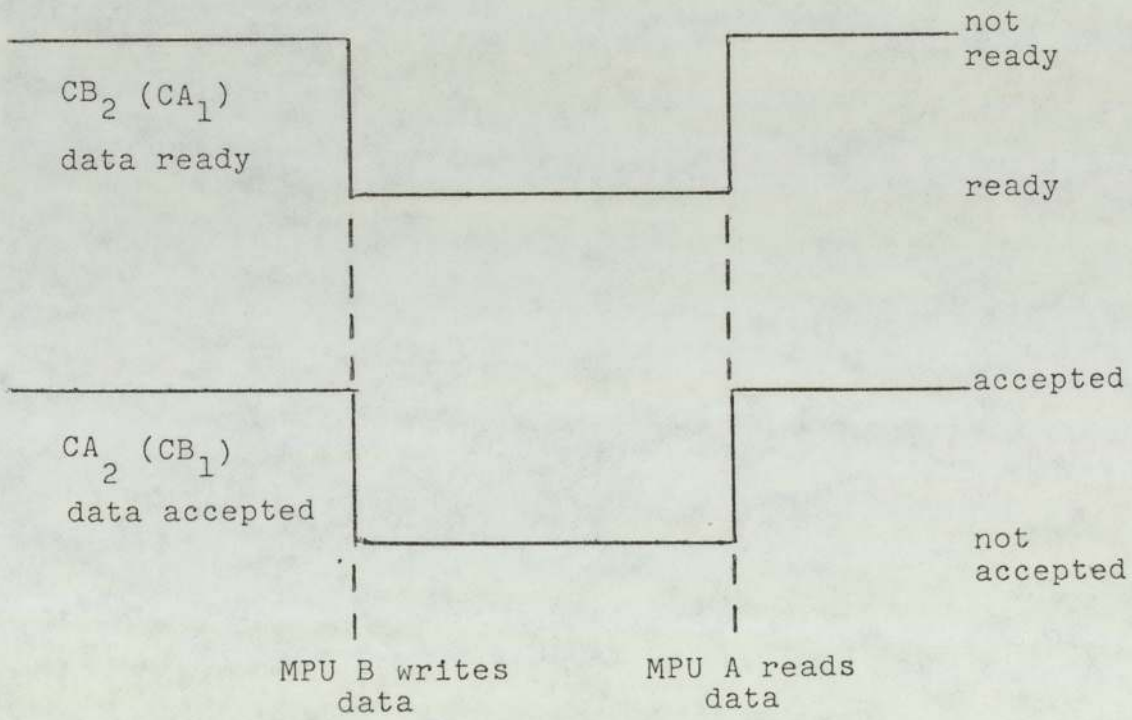


Figure 5.4. The 'Handshaking' Protocol between two Parallel Interface Adaptors (PIA).

PIA A to ensure that data is ready and then proceeds to read the data from the PIA's data register. This read operation causes signal CA_2, 'data accepted' to change level which is detected by PIA B on input CB_1. This event resets CB_2, 'data ready', and sets an internal status flag within PIA B which indicates to microprocessor B that data has been accepted. This cycle of events is repeated as many times as is required to transfer a complete message. Transfer rates can approach over 70,000 bytes per second.

Input and output via ACIA chips is of course across serial data lines with no handshaking protocol whatsoever. Data is transferred according to the baud rate (110, 300, 600 etc. bits per second) which is controlled by clock circuitry external to the ACIA. It is possible to set the baud rate for each of the ACIA chips at any of the standard values. Status flags within the chip again indicate whether data is available for reading by the microprocessor or if the ACIA is available for outputting data.

The management of interrupts generated by the PIA and ACIA chips is described in the following section.

5.2.3.3 Interrupt Levels

The processors are interrupt driven. Once power is applied to the hardware, each processor performs some initialisation of variables, clearing storage locations for example, and waits for a message from either a neighbouring processor module, the user or the host retrieval system. The arrival of a message is indicated by the generation of an interrupt by an input/output port. The interrupt signal is detected

by the microprocessor which immediately executes a piece of code to establish which input channel has enabled the interrupt. This is achieved by polling the status of the PIA or ACIA chips. The processor reads in the message and processes it. Since there are two input channels associated with each of the three processors, multiple interrupts must be handled. Also, if either of the two channels is deemed to be more important than the other and requires immediate attention then there has to be a method of assigning priority to the channels. The M6800 has facilities for handling these problems.

The M6800 microprocessor has three hardware interrupt inputs referred to as reset (RES), non-maskable interrupt (NMI) and interrupt request (IRQ). These are three different signals which can enable three different instruction sequences if their logic levels are switched. The reset interrupt is effectively unavailable for use in a system by the designer. It is only used following 'power-on' to reach an initialising program which sets up system starting conditions such as the initial values of the program counter and stack pointer. Consequently this signal will not be considered further. The other two signals are available for use: one is used for each of the two input channels.

There is a slight variation in the operation of IRQ and NMI. IRQ, interrupt request, is maskable, i.e. the microprocessor can be caused to ignore interrupts generated by this signal by setting a mask bit within the processor. Also, once an interrupt has been generated by this signal and the associated code has been entered, the mask bit is set automatically. Thus, after an interrupt has been

enabled by IRQ and it is being processed, no other interrupts can be caused by the same input (unless the mask is cleared by the user's program). On the other hand, the non-maskable interrupt, as the name suggests, is not capable of being masked. An interrupt channel which utilises this signal will always interrupt the operation of the microprocessor whatever its state. We shall now describe how priorities were assigned to the input/output channels within the terminal.

There are two directions of 'data flow' through the terminal; from user through to host and from host through to user. Each module has an input/output channel for both of these directions, i.e. two in all. There is no handshaking protocol between the host retrieval system and the terminal, so data transmitted by the host must be captured as it arrives, to avoid losing it. Thus the input/output port associated with the host system is attached to the non-maskable interrupt line of the host interface processor, which will therefore always be interrupted, no matter what it is doing, if data is received from the host. The input/output port associated with the central processor is then connected to the IRQ signal within the host interface module. This channel is not so critical due to the existence of the handshaking protocol between processors. The central processor will wait until it is 'told' that a character has been read before transferring another.

A similar strategy is followed in the design of the user interface module. The input/output channel in the host to user direction is given the higher status and connected to the NMI line. The channel in the opposite direction is assigned to IRQ. If any messages are received from the host which are unsolicited, e.g. a message from the host system's operators notifying users of impending closedown, it will be brought to

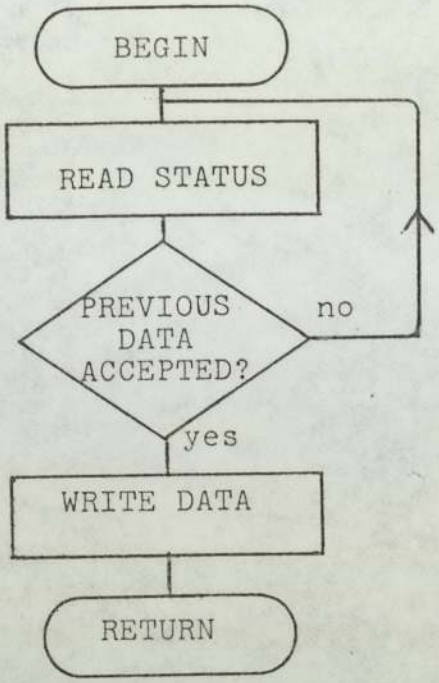
the user's attention immediately. Within the central processor similar assignments were also made, keeping it in line with the other modules. Hence, throughout the terminal data flowing towards the user is deemed critical and is always acted upon by each of the processors immediately. Data flowing in the opposite direction is considered less critical and attention requests will be kept pending by a processor until it is ready to act on them. It should, of course, be remembered that not all data received by the terminal from the host system is shown to the user and some messages passed on to him originate from within the terminal itself. The central processor generates messages as products of the parasite retrieval system operating within it.

5.3 Basic System Software

In this section we propose only to describe the low level operations required of the software for each module. The higher level functions are described in individual chapters later. The parasite retrieval mechanism presently operated by the central processor has been described from a general viewpoint in chapter 4 and its implementation on the terminal is described in chapters 6 and 7.

All software for the terminal is written in M6800 assembler (Motorola, 1975b) using the EXORciser system for its development.

i) Output:-



ii) Input:-

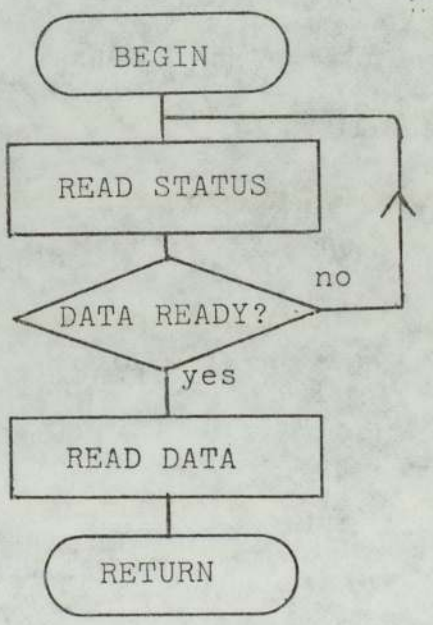


Figure 5.5. Basic Structure of Routines to Read and Write Data via a PIA.

5.3.1 Common Routines

Some basic routines were common to all processors and are concerned with the reading and writing of data. Routines are required for the following operations:-

i) read a byte of data from a specified PIA,

ii) read a byte of data from a specified ACIA,

iii) write a byte of data to a specified PIA,

iv) write a byte of data to a specified ACIA.

Figure 5.5 shows flow diagrams for routines which handle input and output from and to a PIA. Input parameters are the address of the PIA and, for the write routine, the data to be output. An output parameter, for the read routine, is the data read. The flow diagrams are equally valid for handling an ACIA. However, the actual code differs since the status flags of ACIA and PIA chips are arranged differently.

5.3.2 The Host Interface

The host interface plays largely a passive role. The main flow of control is illustrated in figure 5.6. Once the power has been switched on the processor initialises variables and simply waits for an interrupt. On arrival, the interrupt is immediately recognisable as originating from the host system or the central processor because it is either an NMI or IRQ type request. The messages received are processed (see later chapters) and the processor returns to its wait state.

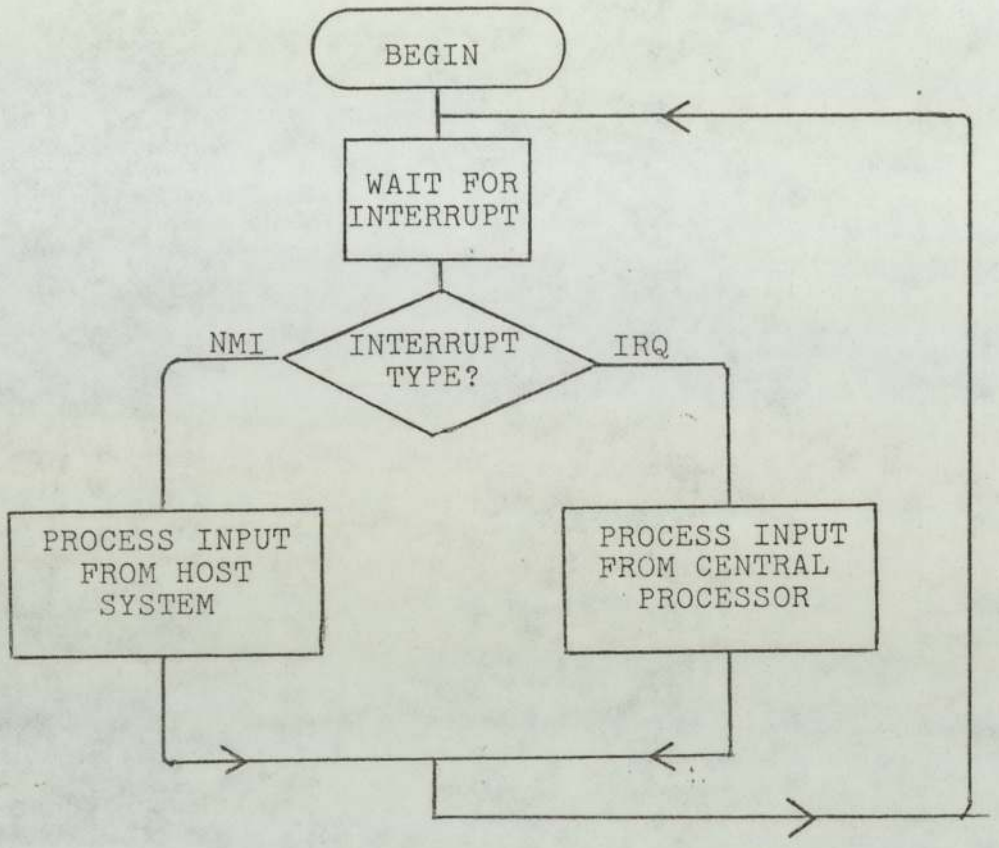


Figure 5.6. Basic Structure of the Host Interface Software.

5.3.3 The User Interface

This module also plays a passive role. The flow of control is essentially that of the host interface with central processor substituted for host (driving NMI) and user substituted for central processor (driving IRQ).

5.3.4 The Central Processor

The central processor operates the parasite retrieval system and is consequently the driving force within the terminal. This processor initiates input from the user and host system via the interface modules. The flow of control is dependent upon the retrieval mechanism implemented: the main discussion of the operation of this module is postponed until the following chapters.

5.4 Summary

In this chapter, we have described the hardware structure of the intelligent terminal and the philosophy behind its design. The prototype version which has been built allows testing and evaluation of the design in terms of the capability for supporting some information retrieval strategies and allowing them to be tested on a large scale. The modular structure of the terminal introduces a great degree of flexibility. The terminal may be programmed to work with different host retrieval systems, operate different retrieval strategies and communicate with the user through various interfaces. In most cases, changes will be limited to the

software associated with one of the modules. Parallelism is also supported by the terminal through its multi-processor configuration. An evaluation of this prototype in terms of cost and performance is given in chapters 7 and 8.

CHAPTER 6

IMPLEMENTATION OF A WEIGHTED TERM RETRIEVAL MECHANISM ON THE INTELLIGENT TERMINAL

6.1 Introduction

In this chapter we describe in detail the implementation of a retrieval mechanism on the intelligent terminal. Chapter 4 explained how an intelligent terminal communicating with a retrieval system has the potential for operating weighted search term retrieval mechanisms. We consider such a mechanism here. Term weights are derived from collection frequencies (postings) and documents are ranked according to the degree of similarity between them and the query. Laboratory experimentation with test collections has given favourable results for this type of weighting (Sparck-Jones 1972). This particular retrieval strategy was chosen for implementation because:-

i) the strategy is ripe for large scale trials as laboratory tests with it have been successful.

ii) it may be very easily adapted at a later stage to an interactive strategy which may modify the term weights during a retrieval session using information about the relevance of document descriptors displayed to the searcher.

Programs have been written to run on the intelligent terminal which allow a user to type in his query (a list of search terms); calculate the inverse collection frequency weight for each term in the query; rank the document collection according to a 'measure' of the similarity between the query and each document in the collection, and retrieve the document descriptors in order of decreasing similarity.

The following section gives an overview of the mechanism which allows the intelligent terminal to carry out weighted search term retrieval and produce a ranked output. A rigorous analysis of the algorithms implemented is presented in section 6.3 and, in section 6.4, the question of how the operation of the retrieval mechanism may be made more efficient is considered. Details of the operation of the software within the terminal, such as program structure and data structures, are contained in the following chapter.

6.2 Outline of the Method

The retrieval strategy implemented requires weights to be calculated for each term of the user's query and for documents within the collection to be retrieved in order of decreasing similarity to the query. A numerical measure of similarity is obtained using the following matching function:-

$$S_d = \sum_{t \in (D_d \cap T)} w_t$$

where D_d is the set of index terms associated with document, d;

T is the set of terms representing the user's query;

w_t is the weight associated with term, t.

Term weights are given by:-

$$w_t = \log \frac{N}{n_t} \times 100$$

where N is the size of the document collection

and n_t is the number of documents indexed by term, t.

The similarity value, S_d, given by the formula above has been referred to as 'notional coordination level' following a comparison with the generally accepted meaning of 'coordination level searching' (Sparck-Jones 1973, Robertson and Sparck-Jones 1976). The latter is simply the tally of terms which are common to both document and query. Simple coordination level values are obtained for S_d by the above formula if term weights are all set to unity (and the multiplying factor of 100 is ignored). The term weights, w_t, are referred to as inverse collection frequency weights for obvious reasons.

The information required to calculate the term weights is available from operational retrieval systems. Term collection frequencies are simply postings figures which are available to on-line users through the use of appropriate search commands. The collection size is published in system documentation and updated regularly.

Thus, the general scheme of things is as follows. The searcher types in his query (which is simply a list of search terms) into the intelligent terminal. The terminal automatically requests the postings figure for each query term and calculates the inverse collection frequency weight. Now, the terminal must present to the user a list of references, ranked in decreasing order of similarity to the query. It is of course impractical for the terminal to retrieve a large number of document descriptors and sort them. The ranked document list can, however, be produced in a less obvious but more practical way.

For any query, given the particular form of matching function above, the document descriptors which will head the ranked list will be those descriptors which contain all the terms in the user's query. These descriptors, if any exist, can be retrieved by submitting a query to the host retrieval system which is a conjunction of all the terms in the user's query. The descriptors lying at the next rank position will be those which contain every term except the one with the least weight. Again, any such descriptors can be retrieved by a suitably structured query. And so on. To summarise, the document descriptors can be retrieved from the host retrieval system in a correctly ranked order by submitting a series of queries which are conjunctions of subsets of the complete set of query terms. All descriptors which are retrieved using this method are ultimately shown to the user: there is no redundant traffic between the terminal and the host retrieval system. The problem is now one

of generating subsets of query terms in the appropriate order ready for transmission to the host system. This task must of course be carried out by the intelligent terminal.

Angione (1975) pointed out that weighted term searches may be converted to Boolean formats. No details are given as to how it is done but a submission is made that the process is complicated and lengthy. The ordering of the term subsets which we require is purely dependent upon the relative magnitudes of the term weights: there is no simple pattern as one might at first assume. This is easily demonstrated by comparing the ordering of term subsets required for two different illustrative queries each consisting of three terms. In one query the term weights are 10, 4, 1, and in the other 10, 10, 10. The two series of subsets are shown in figure 6.1 (a and b). A simple notation is used: a figure one in the n^{th} position signifies the inclusion of the n^{th} term in the subset. The subsets are ordered according to the aggregate weight of included terms. In one case the series turns out to be in strict numerical order, if the subset representations are regarded as binary numbers. In the other case, we have the combinatorial problem of selecting all combinations of m from n objects with m decreasing from n to 0. These are two extreme cases and are unlikely to occur in practice. The first pattern (6.1a) will result if the condition that $2w_i \leqslant w_{i-1}$ exists for $1 < i \leqslant n$, and the second pattern (6.1b) if $w_i = w_{i-1}$ for $1 < i \leqslant n$ (i.e. all weights are equal). Figure 6.1c depicts an example more likely to be found in practice, this time for a query of four terms. Note the lack of any regular pattern.

a) W = (10,4,1)

term subset	total weight($=S_d$)
111	15
110	14
101	11
100	10
011	5
010	4
001	1
000	0

b) W = (10,10,10)

term subset	total weight($=S_d$)
111	30
110	20
101	20
011	20
100	10
010	10
001	10
000	0

c) W = (8,7,6,5)

term subset	total weight($=S_d$)
1111	26
1110	21
1101	20
1011	19
0111	18
1100	15
1010	14
1001	13
0110	13
0101	12
0011	11
1000	8
0100	7
0010	6
0001	5
0000	0

Figure 6.1. Examples of the Ordering of Term Subsets.

Such irregularities are more pronounced for larger queries.

Algorithms for generating the subsets of terms in the appropriate order were devised by the author and are described in the following section. However, before ending this overview, it is perhaps worth noting two particular aspects of the method we have described for producing the ranked list of document descriptors. The first point concerns the number of term subsets required to produce a full ranking of the complete collection. For a query consisting of n terms, 2^n term subsets will be generated. So, for a query of 3 terms the number of subsets is 8 but for a query consisting of 10 terms the number of subsets is 1024. Obviously, such exponential dependence on the size of the query is a major drawback to our strategy. It is certainly not practical to have an intelligent terminal transmit large numbers of queries to the host retrieval system. Section 6.4 describes a technique for reducing the number of queries needed to be sent to the host system. We should also point out that it is most unlikely that the searcher will ever wish to see a complete ranking of the document collection. Usually, only the top few hundred descriptors, at most, are required and therefore, only a proportion of the term subsets will need to be sent to the host system.

Also, the final subset indicated in the examples of figure 6.1 (a and b), which is an empty set, would not be used in practice. This subset represents all the document descriptors in the collection which had not been retrieved up to that point.

6.3 The Algorithms

6.3.1 Formal Description of the Problem

We have a set of search terms $T = [t_1, t_2, \ldots\ldots, t_n]$, with associated weights $W = [w_1, w_2, \ldots\ldots, w_n]$. We wish to find all subsets of T, containing terms with weights which sum to s, where $s_{total} \geqslant s \geqslant 1$, and $s_{total} = \Sigma w_i$, i.e. the sum of all term weights. Generating subsets of terms in order of decreasing sum of weights (equivalent to notional coordination level) and transmitting queries which are conjunctions of the terms belonging to the subsets will retrieve documents in the required ranked order.

The notation used in the description of the algorithms is as follows:-

$\emptyset$ = the empty set.

T = the set of terms representing the query = $[t_1, t_2, \ldots., t_n]$.

T_{sub} = a subset of query terms ($T_{sub} \in T$).

$|T|$ = number of items in T = n.

W = the set of term weights = $[w_1, w_2, \ldots\ldots, w_n]$.

w_i = weight of term t_i, $w_i \geqslant 0$.

s_{total} = sum of all term weights = $\sum_i w_i$.

s = sum of a subset of term weights.

ncl = notional coordination level ($1 \leqslant ncl \leqslant s_{total}$).

In all cases it will be assumed that the sets T and W are ordered on increasing size of term weights, w_i, i.e. $w_{i-1} \leqslant w_i \leqslant w_{i+1}$.

6.3.2 Initial Algorithms

A number of algorithms have been developed which generate subsets of query terms in the appropriate order. The results of initial attempts are described here. At a later stage it was discovered that our problem had wider applicability and that work had already been done on algorithms for solving the same problem by other authors but in different application areas. Section 6.3.3 describes the relationship of the present author's work to that of others. Following examination of other work, further algorithms were developed: section 6.3.4 describes them.

Algorithm SJ1

ncl takes values from s_{total} to 1 in steps of -1. For each value of ncl a call is made to the recursive procedure findset - sj 1 which generates subsets of terms with weights which sum to ncl. The procedure has parameters as follows:

findset - sj 1 (ncl, s, T_{sub}, i),

s and T_{sub} refer to a set of terms under construction which may or may not eventually lead to a set satisfying the condition, s = ncl. The parameter 'i' refers to the next term and weight to be examined by the procedure.

The algorithm is as follows:-

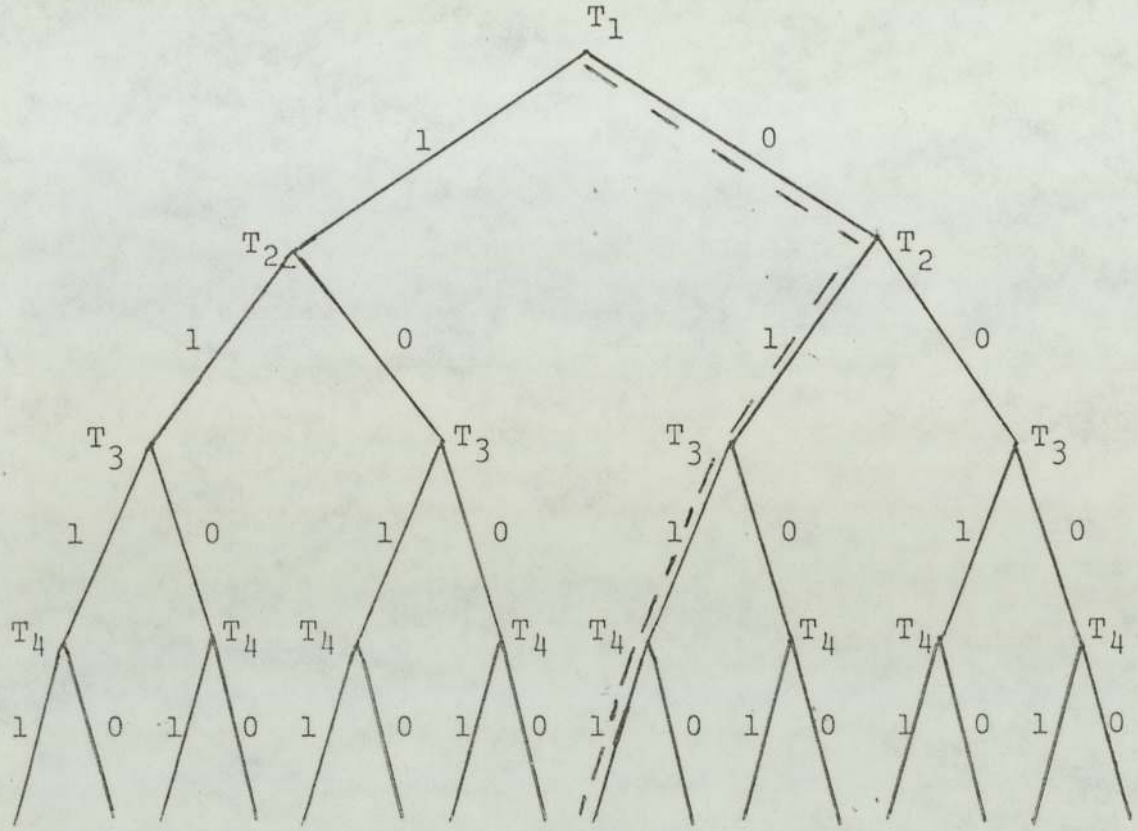
```
BEGIN

  PROCEDURE findset - sjl (ncl, s, T_sub, i);
  BEGIN
    IF s + w_i = ncl THEN
    BEGIN
      output (T_sub ∪ t_i);
      IF i < |T| THEN findset - sjl (ncl, s, T_sub, i+1);
      return
    END ;
    IF i < |T| THEN findset - sjl (ncl, s+w_i, T_sub∪t_i, i+1)
    ELSE return;
    findset - sjl (ncl, s, T_sub, i+1)
  END;

  FOR ncl:=s_total DOWNTO 1 DO
                    findset - sjl (ncl, 0, ∅, 1)

END;
```

This relatively simple algorithm is based upon tree searching. An example using four query terms is shown in figure 6.2. The dashed line shows the path taken through the tree to generate a term subset for a particular notional coordination level. The branches from each node indicate a term's inclusion (1) or exclusion (0).



'1' branch indicates term inclusion

'0' branch indicates term exclusion

subset corresponding to dashed line = (T_2, T_3, T_4)

Figure 6.2. The Tree Corresponding to Algorithm SJ1.

Storage requirement: O(n)

Time requirement = time dependence per search of tree x number of tree searches

$= O(2^n s_{total})$

Algorithm SJ2

The efficiency of algorithm SJ1 can be improved by introducing a simple heuristic. If the partial sum (s) plus the weight of the next term (w_i) is greater than the current value of notional coordination level (ncl) then we can return from the current procedure call as all w_j ($j>i$) are at least as large as w_i (since W is ordered). The heuristic prevents searching sections of the tree which are fruitless. The algorithm becomes:-

```
BEGIN
  PROCEDURE findset - sj2(ncl, s, T_sub, i);
  BEGIN
    IF s + w_i > ncl THEN return; (*new heuristic*)
    IF s + w_i = ncl THEN
    BEGIN
      output (T_sub ∪ t_i);
      IF i<|T| THEN findset -sj 2 (ncl, s, T_sub, i+1);
      return
    END;
    IF i<|T| THEN findset -sj 2 (ncl, s+w_i, T_sub ∪ t_i, i+1)
```

```
    ELSE return;
    findset - sj2 (ncl, s, T_sub, i+1)
  END;

  FOR ncl:= s_total DOWNTO 1 DO
              findset - sj2 (ncl, 0, Ø, 1)

END;
```

Storage requirement: O(n)

Time requirement: $O(2^n s_{total})$

(note: the introduction of the heuristic results in a reduction in the time constant associated with a single traversal of the tree).

Algorithm SJ3

In some instances there will be values of ncl for which there is no corresponding subset of terms. Thus algorithm SJ2 (and SJ1) will make redundant searches of the tree for those particular values of ncl. Heuristics were devised to erradicate redundant traces through the tree. The heuristics allow the algorithm to generate the $(m+1)^{th}$ value of ncl while producing term subsets which sum to the m^{th} value of ncl. The heuristics involve the noting of the highest value of s (the partial sum) generated, which is less than the current value of ncl. ncl is set to this value in the successive search of the tree. Algorithm SJ3 is as follows:-

```
BEGIN

  PROCEDURE findset - sj3 (ncl, s, T_sub, i);
  BEGIN
     IF s+w_i > ncl THEN
     BEGIN
       IF s > nextncl THEN nextncl:=s;
       return
     END;
     IF s+w_i = ncl THEN
     BEGIN
       output (T_sub ∪ t_i);
       IF s > nextncl THEN nextncl:=s ;
       IF i<|T| THEN findset - sj3 (ncl, s, T_sub, i+1);
       return
     END;
     IF i<|T| THEN findset - sj3 (ncl, s+w_i, T_sub ∪ t_i, i+1)
     ELSE return;
     findset - sj3 (ncl, s, T_sub, i+1)
  END;

  nextncl:=s_total;
  WHILE nextncl ≠0 DO
  BEGIN
     ncl:=nextncl; nextncl:=0;
     findset - sj3 (ncl, 0, ∅ , 1)
  END

END;
```

Storage requirement: $O(n)$

Time requirement: $O(2^{2n})$.

To determine the time dependency of this algorithm the number of traces through the tree has been taken to be 2^n (i.e. the number of term subsets). In the majority of cases there are not this many different values of ncl. A particular value of ncl will then result in the generation of a number of term subsets. Thus, the number of traces through the tree is in fact equal to the number of different values of ncl. However, in keeping with normal practice worst case time dependency has been given.

It is interesting to compare the time requirements for algorithms SJ2 and SJ3. $O(2^n s_{total})$ is specified for SJ2 and $O(2^{2n})$ for SJ3. The relationship then, depends on the relative sizes of s_{total} and 2^n, indicating that at times SJ2 may perform better than SJ3 (when $s_{total} < 2^n$). However, since the number of different non-zero values of ncl is always less than or equal to s_{total} then SJ3 is always better, or at least as good as SJ2.

Algorithm SJ4

The heuristic introduced in algorithm SJ2, which prevents searching through fruitless areas of the tree, comes into effect early in the tree search when the value of ncl is small in comparison to the values of the majority of term weights. This point is best

a) W = (10,4,1)

T_{sub}	$\overline{T_{sub}}$	s	$s_{ttl}-s$	
111	000	15	0	
110	001	14	1	
101	010	11	4	
100	011	10	5	$\frac{S_{ttl}}{2}$ symmetry about this point
011	100	5	10	
101	101	4	11	
001	110	1	14	
000	111	0	15	

b) W = (10,10,10)

T_{sub}	$\overline{T_{sub}}$	s	$s_{ttl}-s$	
111	000	30	0	
110	001	20	10	
101	010	20	10	
011	100	20	10	$\frac{S_{ttl}}{2}$ symmetry about this point
100	011	10	20	
010	101	10	20	
001	110	10	20	
000	111	0	30	

Figure 6.3. Symmetrical Distribution of Term Subsets and their Complements about the point given by $S_{total}/2$.

illustrated by means of an example: given the set of weights (1,2,4,6,7,9) the heuristic will be effective when ncl is 2 say, earlier than when ncl is equal to 13. Consider the following: for any subset of terms T_{sub}, the weights of which sum to s, the weights of the remaining terms, $(T - T_{sub})$ must sum to $s_{total} - s$. We shall refer to the set difference as the complement of T_{sub}, i.e. $\overline{T_{sub}}$. For large values of ncl the tree searching process may be made shorter by determining the complement subset of terms which sum to s_{total} - ncl, and outputting the set of terms given by $T - \overline{T_{sub}}$. In the following algorithm we determine complement sets of terms for large values of ncl and true term sets for low values of ncl. The point at which we switch from 'complement mode' to 'true mode' is at the stage when ncl becomes less than $s_{total}/2$. There is a symmetry to the sequence of term sets about this point since, for every T_{sub} there is a corresponding complement set $\overline{T_{sub}}$ which will sum to $(s_{total} - s)$. This property is illustrated in figure 6.3 using examples taken from figure 6.1.

The algorithm is:-

```
BEGIN

  PROCEDURE findset - sj4 (ncl, s, T_sub, i, mode);
  BEGIN
    IF s+w_i>ncl THEN
    BEGIN
      IF mode = complement THEN
```

```
    IF s+w_i<nextncl THEN nextncl:= s+w_i
    ELSE
    IF s>nextncl THEN nextncl:=s;
    return
  END;
  IF s+w_i = ncl THEN
  BEGIN
    IF mode = complement THEN
    BEGIN
      output (¬(T_sub∪t_i));
      IF i<|T| THEN IF ncl+w_i+1<nextncl THEN nextncl:=ncl+w_i+1
    END ELSE
    BEGIN
      output (T_sub∪t_i);
      IF s>nextncl THEN nextncl:=s
    END;
    IF i<|T| THEN findset - sj4 (ncl, s, T_sub, i+1, mode);
    return
  END;
  IF i<|T| THEN findset - sj4 (ncl, s+w_i, T_sub∪t_i, i+1, mode)
  ELSE return;
  findset - sj4 (ncl, s, T_sub, i+1, mode)
END;

nextncl:=0
WHILE nextncl<s_total/2 DO
BEGIN
```

```
    ncl:=nextncl; nextncl:=s_total;
    findset - sj4 (ncl, 0, ∅, 1, complement)
  END;
  IF nextncl ≠ s_total/2 THEN nextncl:=ncl;
  WHILE nextncl ≠ 0 DO
  BEGIN
    ncl: = nextncl; nextncl:=0
    findset - sj4 (ncl, 0, ∅, 1, true)
  END;
END;
```

Storage requirement: O(n)

Time requirement: $O(2^{2n})$

(note: once again the reduction in time is due to the time constant associated with a search of the tree)

The final observation is concerned with the detection of the next valid value for ncl. In complement mode it is the lowest value of s generated which is greater than the current value of ncl. In 'true' mode detection of the next ncl value is equivalent to the method used in algorithm SJ3.

Algorith SJ5

Because of the symmetry in the sequence of 'complement' and 'true' term subsets an algorithm may be designed which stores the sequence of subsets generated whilst in complement mode and simply

refers to the resulting stored list during the second stage of the process - 'true mode' - instead of re-searching the tree for each term subset. The resulting algorithm would come from quite simple modifications to SJ4 and consequently, a detailed specification is not given. The resulting algorithm would require $O(2^{n-1})$ for storage. As the intention is to implement one of the algorithms we have described on a microcomputer system in which storage capacity may be limited, consideration of algorithm SJ5 as a valid alternative was discounted.

Details of actual execution times are given in section 6.3.5 after the discussion of the relationship between these algorithms and those described in the literature.

6.3.3 The Partition and Knapsack Problems

Horowitz and Sahni (1974) describe algorithms for solving the following problem:-

> 'given r numbers $n_1 \ldots n_r$ we wish to find all possible combinations of these numbers which sum to M.'

It is clearly related to the problem described in the previous section; that of generating subsets of query terms whose weights sum to a particular notional coordination level. Equivalence is established by substituting the number of terms in the query

for r, term weights for $n_1 \ldots n_r$ and notional coordination level for M. Horowitz and Sahni state that their problem is closely related to the classical number theory study of determining partitions. Determining partitions of M would imply that $n_i=i$ and $n_r=M$. Hardy and Wright (1959) provide a strict formal definition and description of the partition problem. Motivation for Horowitz and Sahni's work in this area stems from a desire to examine solutions to another related problem; one which has numerous applications. By restricting all n_i and M to integer values; introducing a new set of integers $P = (p_1, p_2, \ldots p_r)$ referred to as profits, and an objective function, O, we have an integer programming problem called the 'Knapsack problem':-

$$\text{maximise} \sum_{i \leqslant i \leqslant r} p_i \delta_i \quad \text{(objective function, O)}$$

$$\text{subject to} \sum_{i \leqslant i \leqslant r} n_i \delta_i \leq M \quad \text{(constraint)}$$

$$\text{where } \delta = (0,1)$$

The problem's title arises from its informal resemblance to that facing a hiker who has to decide which items to pack in his knapsack, given the maximum weight that the sack will carry (M) along with the weight (n_i) and measure of the desirability (p_i) associated with each item to be packed. Solutions to the knapsack problem have many applications.

Brown (1971) discusses a number of industrial applications for the knapsack problem including optimal cutting of sheet steel or paper given a specified set of required widths and lengths, and scheduling of television commercials given their individual durations and the lengths of the commercial breaks between television programmes. Merckle and Hellman (1978) describe an application within the area of cryptology. Garfinkel and Nemhauser (1972) use the example of deciding which combination of scientific equipment to transport on a space vehicle: they attach a scientific value per unit weight for each piece of equipment. They also report that there exist complex problems which are solved by a particular model which uses the knapsack problem as a subroutine and requires the solution of a very large number of knapsack problems. Most published solutions to the knapsack problem do not fully satisfy our requirements, eg Zoltners (1978), Ibarra and Kim (1975), Greenberg and Hegerich (1969), and Kolesar (1967). They each generate optimum solutions (ie those maximising the objective function) without the need to generate all solutions which satisfy the constraint. We require the generation of all solutions which satisfy the constraint as an equality.

Mathematical analyses of the problem are given by Garey and Johnson (1978). Lewis and Papadimitriou (1978) provide a fine introduction to the class of problems in which the knapsack and partition problems lie ('NP-complete'). Aho (1976) and Aho, Hopcroft and Ullman (1974) provide further discussions of problem complexity with reference to the partition and knapsack problems. Useful descriptions of algorithmic design techniques are provided by

Tarjan (1978), Bentley (1979) and Weide (1977).

The most useful information for the present purposes is to be found in Horowitz and Sahni (1974). Further reference is made to this study in the next section where more algorithms are described combining heuristics described above with others suggested by Horowitz and Sahni.

6.3.4 Further Algorithms

Horowitz and Sahni (1974) specify an algorithm which is the same as algorithm SJ2 but for the addition of two further heuristics. A new parameter is added to the procedure calls which is equivalent to the sum of weights of the terms yet to be considered for inclusion in the subset of terms, T_{sub}. This parameter is given the name 'rem' (for remainder). The following two heuristics can then be used:-

i) if the partial sum s plus the total sum left (rem) is not enough to reach ncl then return from the procedure call.

ii) if the partial sum s plus the total sum left (rem) is exactly equal to ncl then output the union of T_{sub} and all remaining terms and return from the procedure call.

This provides us with the following algorithm.

Algorithm HS

```
BEGIN

  PROCEDURE findset - hs (ncl, s, T_sub, i, rem);
  BEGIN
     IF s + rem <ncl THEN return;
     IF s + rem = ncl THEN
     BEGIN
       output (T_sub ∪ (t_i, t_i+1, ....., t_n);
       return
     END;
     IF s + w_i > ncl THEN return;
     IF s + w_i = ncl THEN
     BEGIN
       output (T_sub ∪ t_i);
       IF i < |T| THEN findset - hs (ncl, s, T_sub, i+1, rem-w_i);
       return
     END;
     IF i < |T| THEN findset - hs (ncl, s+w_i, t_sub ∪ t_i, i+1, rem-w_i)
     ELSE return;
     findset - hs (ncl, s, T_sub, i+1, rem-w_i)
  END;

  FOR ncl:=s_total DOWNTO 1 DO
          findset - hs (ncl, 0, ∅, 1, s_total)

END;
```

Storage requirement: $O(n)$

Time requirement: $O(2^{2n})$

This algorithm is essentially the same as algorithm SJ2 except for the introduction of the two new heuristics. These lead to a lower execution time as they serve to further reduce the number of searches of fruitless areas of the tree. The heuristics were also added to algorithms SJ3 and SJ4 to give algorithms HS-SJ3 and HS-SJ4. Detailed specification of these algorithms will not be given.

6.3.5 Results of a Comparison of Execution Speeds

Measurements were made of the execution time of a number of the algorithms described in the previous sections using a number of different sets of data. The BCPL programming language was used (Richards 1974) and the programs run on an ICL 1904S computer. The algorithms chosen for examination are HS, HS-SJ3, HS-SJ4 and SJ4. It is evident that the introduction of the heuristics suggested by Horowitz and Sahni will improve the performance of the SJ series of algorithms. Hence the choice of HS-SJ3 and HS-SJ4. Algorithms HS and SJ4 were chosen so that the effect of the heuristics devised by the author and those devised by Horowitz and Sahni could be compared. (SJ4 is the most evolved of the SJ series of algorithms).

The tests used the following sets of data:-

a) sequential data,

$w_i=i$, $w_i=2i$, $w_i=5i$, $w_i=10i$, $w_i=20i$, $w_i=50i$

with $1 \leq i \leq 8$ and $1 \leq i \leq 12$

b) random data,

w_i = random number in (1 ... 100)

n = 8 (5 sets)

and n = 10 (5 sets)

The results are given in figure 6.4.

An immediate observation that can be made about the sequential data is that algorithms SJ4, HS-SJ3 and HS-SJ4 show execution times which are fairly constant for a particular set size. On the other hand the execution time of algorithm HS increases as the difference between values within a set increases. This is explained by the fact that algorithm HS does not detect the next valid ncl value and thus attempts to determine non-existent subsets of terms. Algorithm HS also shows poorer performance amongst the sets of random data for the same reason.

Of the other three algorithms, SJ4 exhibits the best performance. SJ4 is up to 20% faster than HS-SJ3 and HS-SJ4 in one case (RAN-8-2). Clearly, algorithm SJ4 is the superior algorithm of those described

	SJ4	HS	HS-SJ3	HS-SJ4
SEQ-8-1	2195	2227	2329	2233
SEQ-8-2	2198	3665	2333	2236
SEQ-8-5	2200	7980	2335	2239
SEQ-8-10	2203	15254	2337	2241
SEQ-8-20	2204	29811	2338	2242
SEQ-8-50	2207	-	2341	2245
SEQ-12-1	34253	34085	36978	34692
SEQ-12-2	34257	50873	36982	34697
SEQ-12-5	34259	-	36985	34699
SEQ-12-10	34262	-	36987	34702
SEQ-12-20	34268	-	36993	34707
SEQ-12-50	34270	-	36994	34709
RAN-8-1	7306	23407	7951	7679
RAN-8-2	6371	19941	7853	7558
RAN-8-3	8340	15447	9998	9441
RAN-8-4	6615	12346	8170	7862
RAN-8-5	6415	20549	7502	7213
RAN-10-1	28059	42331	35610	32498
RAN-10-2	39408	44059	43904	40316
RAN-10-3	30850	34021	34615	31653
RAN-10-4	43373	-	48711	44912
RAN-10-5	35558	44807	39929	37984

All times in milliseconds

'-' indicates that the program failed to execute within 60 CPU seconds.

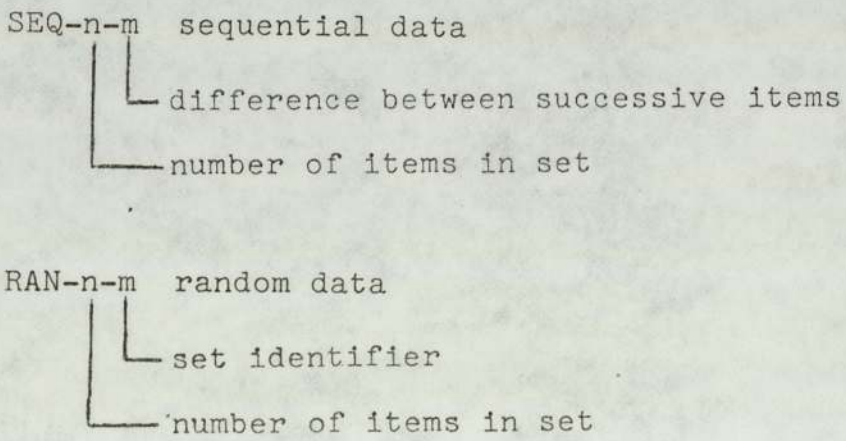


Figure 6.4. Results of a Comparison of Execution Speeds of the Algorithms for Generating Term Subsets in the Appropriate Order.

and the likely choice for implementation on the intelligent terminal. However, the simplicity of the algorithm was also taken into account when deciding which algorithm to implement. Of the three fastest algorithms the simplest is in fact algorithm HS-SJ3. Therefore, to speed implementation and reduce the memory requirements of the programs, algorithm HS-SJ3 was translated into M6800 assembler language and implemented on the intelligent terminal.

6.4 An Enhancement to Improve the Efficiency of the Method

6.4.1 An Inefficiency in the Method so far Proposed

For a query consisting of n terms the total number of term combinations generated will be 2^n-1. Thus, for a query containing 5 terms, 31 different term combinations will be generated while, for a query containing 10 terms, 1023 term combinations will result. As each term combination represents a query sent to the host retrieval system, clearly the large numbers of term combinations resulting from queries consisting of 6 or more terms cannot be entertained: the process would take too long, and would prove too expensive. However there are some techniques which go some way towards overcoming this difficulty.

6.4.2 Solutions

Two suggestions were considered to alleviate the difficulty highlighted above. The first is to retrieve a set of documents by forming a query consisting of a disjunction of terms appearing in the user's query. The similarity function is then used within the terminal to rank the retrieved set of descriptors. This possibility was discounted due to the large number (O(100)-O(1000)) of document descriptors likely to be retrieved. The amount of storage that the terminal has at its disposal may not be adequate in some cases and the time required to capture the data also places the method at a disadvantage. The second, following, suggestion appears more reasonable.

Figure 6.1 defines the order of term combinations for some example queries. It can be seen that a number of combinations contain sets of common terms, eg t_1 and t_2. Now if the conjunction of terms t_1 and t_2 alone does not lead to document references being retrieved then neither will any term combination which contains terms t_1 and t_2 as a subset. Thus, if it were established early in the dialogue that the terms t_1 and t_2 combined together produce no documents then many other queries could be eliminated within the terminal and need not be sent to the host system.

An investigation of this technique will now be presented. We shall first outline the operation of the method and then report results of experiments carried out with some test document collections.

We shall use the phrase 'pre-searching' to refer to the practice of submitting searches out of sequence, simply to determine whether they retrieve anything from the host.

6.4.3 The Pre-Searching Method

The purpose of submitting a pre-search is to decrease the total number of queries or term combinations submitted to the host. Nothing is gained if the pre-search itself retrieves documents; only those pre-searches which score no hits are of any benefit. If possible therefore, only those combinations of terms which are thought unlikely to retrieve documents should be submitted. The probability that two or more terms are assigned to the same documents can be related to the weights assigned to query terms. The probability, P_i, that term t_i is assigned to any document in the collection, chosen at random, is given by:-

$$P_i = \frac{n_i}{N}$$

where, as before, n_i is the collection frequency of term t_i, and N is the size of the collection. The probability that terms t_i and t_j, are assigned to the same document is given by:-

$$P_{ij} = P_i * P_j = \frac{n_i}{N} \times \frac{n_j}{N}$$

assuming term independence. The conjunction of terms t_i and t_j should be chosen as a pre-search if the resulting combined probability is below some threshold, T.

$$\text{ie } P_{ij} < T$$

Inverting this condition and transforming to logs we obtain,

$$\log (1/P_i) + \log 1/P_j) > T'$$

Now, the frequency weights assigned to terms are,

$$w_i = \log (N/n_i) = \log (1/P_i)$$

Therefore a conjunction of terms is chosen for submission to the host system as a pre-search if the sum of the inverse collection frequency weights of the two terms exceed some threshold, T'. Of course, a similar argument applies to pre-searches of more than two terms. However, the more terms there are in the pre-search, the fewer queries the pre-search can eliminate. We therefore use pre-searches with few terms.

6.4.4 The Pre-Searching Algorithm

Once a term subset has been generated by one of the algorithms described earlier, it is passed to a search procedure which operates

the pre-search mechanism. The procedure is invoked by the following statement:-

search (T_{sub})

The notation and variable names used in the following outline of the algorithm are similar to those used in the earlier sections with the following additions:-

PS = a pre-search (a subset of search terms)
threshold = pre-search threshold weight
pssizelimit = maximum number of terms allowed in a pre-search.

The algorithm is as follows:-

```
PROCEDURE search (T_sub);
BEGIN
  PROCEDURE downtree (T_sub, PS, noofterms, nextterm, wt, send);
  BEGIN
    IF wt ≥threshold THEN
    IF not alreadysent (PS) THEN
    BEGIN
      transmitpre-search (PS, send);
      IF not send THEN return
    END;
```

```
    IF |PS| <pssizelimit THEN
    FOR i:= nextterm DOWNTO 1 DO
    BEGIN
      downtree (T_sub, PS⌣t_sub_i , nooftermss +1, i-1, wt+w_sub_i, send);
      IF not send THEN return
    END
  END  (downtree);
  PROCEDURE pre-search (T_sub, send);
  BEGIN
    FOR i:=|T_sub| DOWNTO 2 DO
    FOR j:=i-1 DOWNTO 1 DO
    BEGIN
      downtree (T_sub, t_sub_i⌣t_sub_j, 2, j-1, w_sub_i+w_sub_j, send);
      IF not send THEN return
    END;
  END  (pre-search);
  IF not blocked (T_sub) THEN
  BEGIN
    send:=true;
    pre-search (T_sub, send)
    IF send THEN transmitquery (T_sub)
  END;
END (search);
```

Some procedure and function calls have been used in the outline without stating explicitly what they do. The procedures, 'transmitquery' and 'transmitpre-search' handle the transmission of queries and pre-searches to the host retrieval system and process the replies from the host appropriately. 'blocked' and 'alreadysent' are Boolean functions. If a query is prevented from being sent to the host system by existing successful pre-searches then this is detected by the function 'blocked'. The function 'alreadysent' checks a pre-search specification against a list of unsuccessful pre-searches which have already been sent to the host system thus preventing subsequent transmissions.

Pre-searches are formed as and when necessary. If a search term subset is not 'blocked' by existing pre-searches then others are formed. Terms are selected for pre-searches in order of decreasing weight to take advantage of the relationship between term weights and probability of term co-occurence explained earlier. If a pre-search fails (retrieves document records) then a subsequent term is added (through the recursive procedure 'downtree') and the resulting larger set of terms is then tried. The process is continued until a successful pre-search is located, thus blocking the query represented by the term subset, or all possible pre-search formulations are exhausted.

6.4.5 Results of Tests

Experiments were conducted with three document collections to examine the effect of the introduction of pre-searching. The collections used were the CRANFIELD 200, KEEN 800 and CRANFIELD 1400 collections (for details of these see, for example, Sparck-Jones 1973 and Sparck-Jones & Webster 1980). The experiments were designed to try and provide answers to the following questions:-

i) how many search statements are required to retrieve the first 1, 5, 10, . . . etc. documents in the ranked list, for queries of different sizes?

ii) what effect does pre-searching have on the figures obtained in i)?

iii) does changing the value of the pre-search threshold weight have any noticeable effect?

iv) does the size of the document collection have any bearing on the number of search statements required?

Programs were written in the BCPL programming language (Richards 1974) and run on an ICL 1904S computer. Tables of results are presented in appendix 4. The main conclusions that can be made from them are summarised here.

		SIZE OF QUERY (no. of terms)										
		3	4	5	6	7	8	9	10	11	12	13
NUMBER OF QUERIES IN COLLECTION	C200	1	4	6	10	8	3	3	5	2	0	0
	KEEN	11	8	14	8	4	6	1	2	1	0	2
	C1400	10	10	31	22	30	38	20	25	14	8	10
		NO. OF SEARCH STATEMENTS										
1 DOC	C200	1	3	6	11	47	19	257	298	854	-	-
	KEEN	2	4	11	21	47	173	188	797	1369	-	5061
	C1400	1	2	4	9	26	62	127	304	509	1068	2306
5 DOCS	C200	5	8	16	34	77	194	433	499	1644	-	-
	KEEN	4	10	22	48	83	229	383	921	1967	-	7446
	C1400	3	5	10	21	53	125	275	583	1090	2425	5546
10 DOCS	C200	5	9	20	41	92	213	459	721	1798	-	-
	KEEN	5	11	26	53	92	237	446	975	1991	-	7906
	C1400	3	7	15	28	69	152	331	692	1355	2858	6632
15 DOCS	C200	5	11	24	47	98	224	480	829	1875	-	-
	KEEN	6	12	27	54	108	241	461	996	2000	-	8059
	C1400	3	8	17	32	77	167	356	753	1553	3069	6976
20 DOCS	C200	5	12	25	51	104	235	486	885	1924	-	-
	KEEN	6	12	28	56	112	243	473	1002	2015	-	8072
	C1400	4	8	18	36	83	177	377	791	1621	3231	7144
30 DOCS	C200	5	12	27	55	111	240	491	949	1973	-	-
	KEEN	6	13	29	58	116	245	488	1011	2025	-	8131
	C1400	5	9	20	40	91	190	401	838	1736	3429	7507
ALL DOCS	C200	7	15	31	63	127	255	511	1023	2047	-	-
	KEEN	7	15	31	63	127	255	511	1023	2047	-	8191
	C1400	7	15	31	63	127	255	511	1023	2047	4095	8191

C200 = CRANFIELD 200; C1400 = CRANFIELD 1400;

Figure 6.5. Results of Trials with test collections: The number of search statements required to retrieve proportions of the ranked list of document descriptors.

Firstly, the numbers of search statements to retrieve the first, 1, 5, 10, ... etc. documents in the ranked list are shown in figure 6.5. The figures for queries of size 3, 4 or 5 terms are not particularly great as expected. The number of term subsets generated to produce the complete ranked list for a query of size n terms is 2^n-1: thus for n=5 the total is 31. The figures for larger queries increase rapidly with n due to the exponential dependence, and this is seen in the results. Each value shown is an average for all queries of a particular size within one document collection.

It should be remembered that these figures represent the number of search statements transmitted by the terminal to the operational retrieval system to which it is connected. They therefore have a direct bearing on the time and cost of the search process. A large number means a longer connect time and thus higher costs. If the average response time for each search statement is 5 seconds then, to retrieve the first ten documents for a query of size ten terms, taking the figure for the CRANFIELD 1400 collection, would take approximately 58 minutes. To retrieve the complete ranked list would require approximately 85 minutes. A response time of 5 seconds may be somewhat optimistic so in practice these times could be longer. The operation would therefore be quite expensive and suggests that any trials in an operational environment would have to be limited to relatively short queries. The size of query permitted would be determined by the size of the experiment's budget and other practical and ergonomic factors.

Pre-searching was introduced and the programs re-run so that its effect could be examined. The threshold above which the sum of weights of terms in the pre-search formulation must lie was varied to determine its effect. The number of pre-searches and number of actual queries required to retrieve the first, 1, 5, 10,... etc. documents in the ranked list was again recorded for each query of each collection. Complete results will be found in appendix 4. Figure 6.6 illustrates a typical set of results. The set illustrated is that for the CRANFIELD 1400 collection with a pre-search threshold weight of 400. This threshold relates to the size of term weights. A search term weight is derived using the formula log (N/n), as already described. To increase the significance of the values given by this formula each result is multiplied by a factor of 100. Hence the thresholds for pre-searches reflect this transformation also. For each query size the table shows the number of pre-searches and number of full queries required to retrieve a particular number of documents. Since a pre-search is effectively another search statement sent to the retrieval system, the values are added together to produce a total which can be used to compare with the totals given in figure 6.5 (when no pre-searching was carried out).

The immediate observation that can be made is that pre-searching can have a dramatic effect on the number of queries required to retrieve the ranked list of document references. For instance, whereas a query of size 10 terms required 304 search statements to be forwarded to the retrieval system to locate the first document in

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 400.

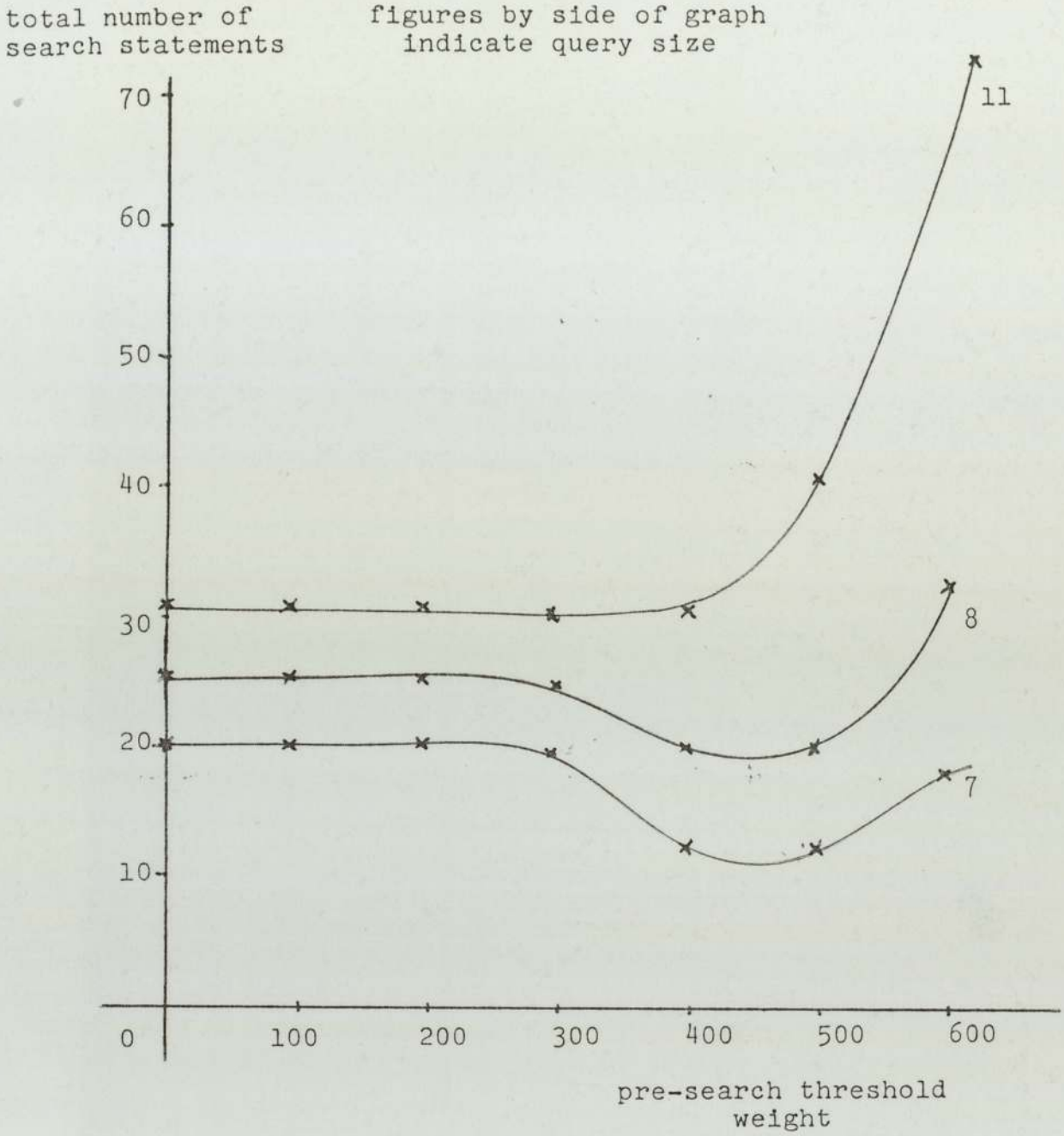
No. DOCS ↓		QUERY SIZE (and population)												
		3	4	5	6	7	8	9	10	11	12	13	14	15
		10	10	31	22	30	38	20	25	14	8	10	5	1
1	S	1	1	3	2	1	1	1	1	1	1	1	1	1
	P	0	0	0	2	11	19	26	25	31	36	41	46	63
	T	1	1	3	4	12	20	27	26	32	37	42	47	64
5	S	3	4	9	10	14	20	24	38	64	185	355	146	74
	P	0	0	0	3	16	34	50	56	74	96	154	193	193
	T	3	4	9	13	30	54	74	94	138	281	509	339	267
10	S	3	5	14	17	25	35	49	72	128	278	471	213	224
	P	0	0	0	3	18	40	59	73	105	131	220	246	260
	T	3	5	14	20	43	75	108	145	232	409	691	459	484
15	S	3	6	15	21	32	46	68	100	207	342	554	259	328
	P	0	0	0	3	18	41	62	80	124	153	256	283	311
	T	3	6	15	24	50	87	130	180	331	495	810	542	639
20	S	4	7	17	25	38	56	84	121	244	408	611	339	360
	P	0	0	0	3	18	41	63	86	133	169	275	310	321
	T	4	7	17	28	56	97	147	207	377	577	886	599	681
30	S	5	8	19	29	46	67	105	154	320	506	733	373	658
	P	0	0	0	3	18	41	66	90	141	188	322	350	408
	T	5	8	19	32	64	108	171	244	461	694	1055	723	1066
ALL	S	7	15	31	52	82	132	210	332	610	1076	1252	788	3850
	P	0	0	0	3	18	41	65	93	147	214	371	359	711
	T	7	15	31	55	100	173	275	425	757	1290	1623	1147	4561

S = no. of true searches; P = no. of pre-searches;
T = total.

Figure 6.6. The effect of pre-searching on the number of search statements required to retrieve proportions of the ranked list of document descriptors (CRANFIELD 1400 collection).

the ranked list, the figure is reduced to 26 when pre-searching is introduced. Nevertheless, for larger numbers of documents for the same query size this figure grows considerably, to finally yield 425 search statements for retrieval of the complete ranked list. However, it should be borne in mind that a user is very unlikely to want a complete list in a real-life search.

The effect of varying the pre-search threshold can be best seen graphically. Figure 6.7 shows the effect of varying the threshold on the number of search statements required to retrieve the first document for a number of different query sizes in the CRANFIELD 1400 collection. Beginning with low threshold values the numbers of search statements remain fairly constant, eventually they decrease and then increase dramatically. This suggests that there is some optimum value of pre-search threshold. The explanation for this effect is as follows. When the threshold is set at a relatively high value, a pre-search has to contain relatively large numbers of terms, so that their combined weight exceeds the threshold. The larger the number of terms in a pre-search the fewer queries it can 'block' (ie, prevent from being sent to the host retrieval system). As the theshold value decreases small pre-searches become possible, and these are more effective in terms of the number of searches they block. However, when the threshold becomes very low, pre-searches are generated which tend to be unsuccessful, ie, they actually retrieve documents. This is due to the relationship between the threshold value and the probability of co-occurence of terms in document descriptors.



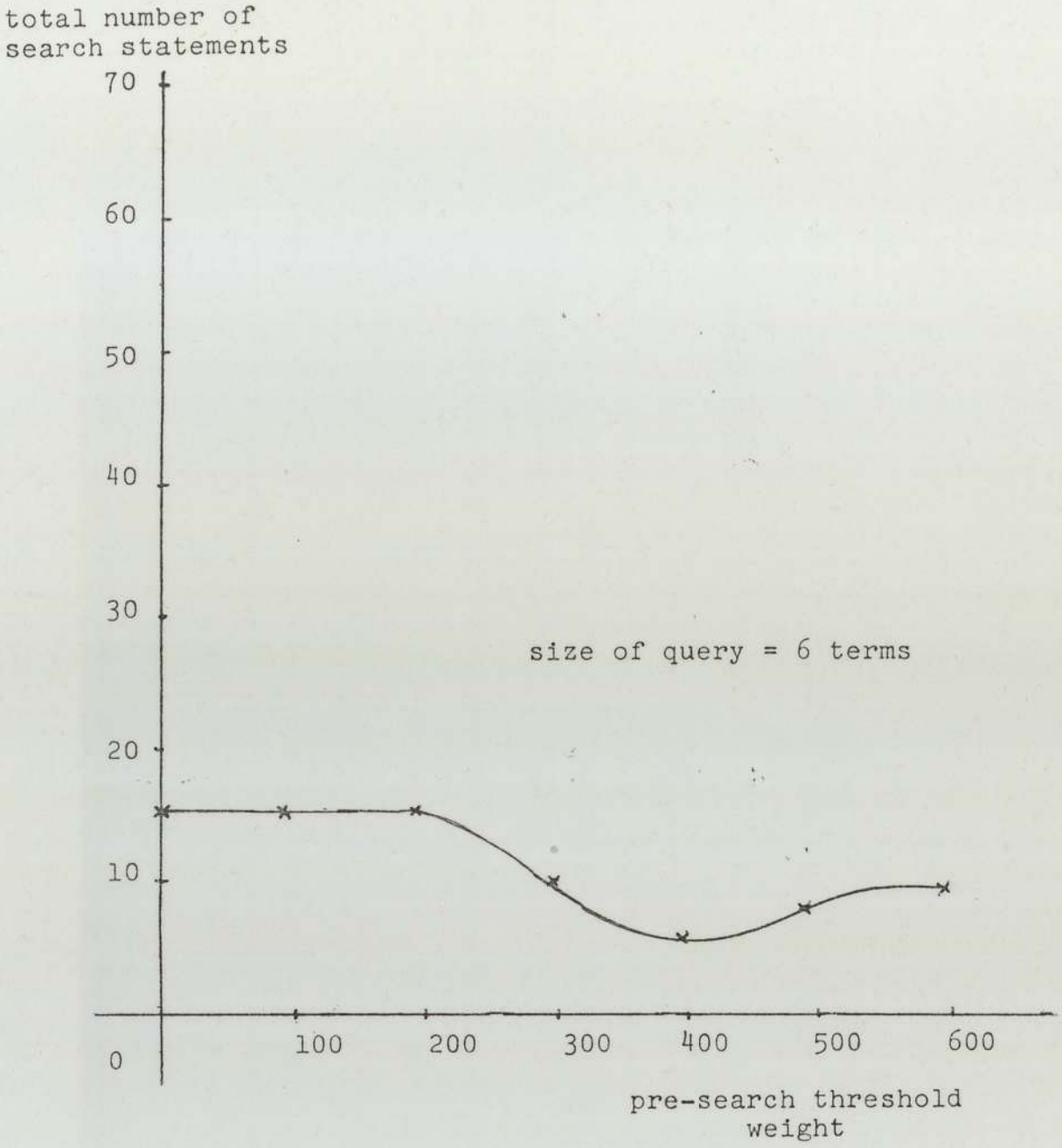
CRANFIELD 1400 COLLECTION: Average number of search statements required to retrieve the first document descriptor in the ranked list.

Figure 6.7. The Effect of Varying the Pre-Search Threshold Weight.

An increase in the number of unsuccessful pre-searches leads to the generation of more pre-searches which causes the rise in the graph at low threshold values.

For some small queries the rise at low values of the pre-search threshold can lead to more search statements being sent to the retrieval system than when no pre-searching was carried out at all. Figure 6.8 shows an example taken from the CRANFIELD 1400 collection for queries of 6 terms. This is because a number of subsets of terms are used unsuccessfully as pre-searches early in the retrieval process, in attempts to block queries, and used again in the later stages of the process as actual queries.

The optimum value of pre-search threshold varies for different collections and different query sizes: the range for the collections used is 100 to 400. Figure 6.9 is a table of results which shows the lowest number of search statements known to be required to retrieve various proportions of the ranked document list for each of the collections. The figures are selected from the complete set of results presented in appendix 4. In practice the pre-search threshold would have to be set to some value prior to processing a query so that these figures are somewhat idealistic since various threshold values have been used; selected with *a priori* knowledge of the complete spectrum of results. However, they are useful in the sense that they present the best known performance of the retrieval process we propose. The phrase 'best known' is used as the figures could be lower for various intermediate values of pre-search



CRANFIELD 1400 COLLECTION: Average number of search statements required to retrieve the first document descriptor in the ranked list.

Figure 6.8. The Effect of using Pre-Searching with Queries Containing Few Terms.

threshold, eg, 150, 250, etc., which were not used in the experiments.

Figure 6.9 may be used to try and answer the fourth question in the above list, i.e. does collection size have any bearing on the number of search statements required? The number of documents in each collection is 200, 797 and 1400 for the CRANFIELD 200, KEEN and CRANFIELD 1400 collections respectively. One analysis suggests that fewer search statements will be required to retrieve the ranked document list (or portions of it) from large collections. This is because although the probability that a number of terms co-occur in one document record is low, a larger population of document records makes it more likely that actual instances of co-occurence exist. Queries containing large numbers of terms, which tend to be generated early in the retrieval process, are therefore more likely to retrieve document descriptors. The results presented in figure 6.9 fail to show any strong trend of this nature. The CRANFIELD 1400 collection tends to require fewer search statements but, overall, the evidence is not particularly conclusive.

To summarise the results, we make the following statements:-

i) Pre-searching is a very effective mechanism for reducing the number of search statements sent by the intelligent terminal to the host retrieval system in order to retrieve the ranked list of document records, or parts of it.

		SIZE OF QUERY (no. of terms)										
		3	4	5	6	7	8	9	10	11	12	13
NUMBER OF QUERIES IN COLLECTION	C200	1	4	6	10	8	3	3	5	2	0	0
	KEEN	11	8	14	8	4	6	1	2	1	0	2
	C1400	10	10	31	22	30	38	20	25	14	8	10
		NO. OF SEARCH STATEMENTS										
1 DOC	C200	1	1	4	4	10	19	20	36	25	-	-
	KEEN	2	4	5	8	13	19	18	34	42	-	50
	C1400	1	1	3	4	12	20	27	25	31	36	40
5 DOCS	C200	5	6	13	25	27	132	59	85	169	-	-
	KEEN	4	8	13	24	30	36	83	58	105	-	145
	C1400	3	4	9	13	30	54	73	90	134	275	503
10 DOCS	C200	5	7	16	32	39	149	73	179	234	-	-
	KEEN	5	9	16	29	37	43	123	80	120	-	239
	C1400	3	5	13	20	43	75	104	136	223	397	677
15 DOCS	C200	5	9	20	38	46	160	87	247	280	-	-
	KEEN	6	9	18	30	49	47	139	94	128	-	280
	C1400	3	6	15	24	50	87	123	168	315	478	791
20 DOCS	C200	5	10	21	42	52	170	92	287	319	-	-
	KEEN	6	10	18	32	51	49	150	98	138	-	291
	C1400	4	7	16	28	56	96	139	191	355	555	864
30 DOCS	C200	5	10	24	46	59	176	97	342	364	-	-
	KEEN	6	11	19	34	56	51	167	107	148	-	326
	C1400	5	8	18	32	64	108	161	223	424	668	1021
ALL DOCS	C200	7	13	27	54	74	190	117	411	434	-	-
	KEEN	7	12	21	38	66	60	190	119	170	-	378
	C1400	7	14	29	55	100	172	264	393	703	1233	1538

C200 = CRANFIELD 200; C1400 = CRANFIELD 1400;

Figure 6.9. The Lowest Known Numbers of Search Statements Required to Retrieve Proportions of the Ranked List of Document Descriptors.

ii) Experience with the test collections suggests that pre-searching is only needed with queries of seven terms or more. (The number of search statements generated for queries smaller than this is not disadvantageous).

iii) An optimal threshold weight can be chosen to help eliminate pre-searches which are likely to be unsuccessful, i.e. are likely to retrieve document records. (The sum of weights of terms within the pre-search must sum to a value greater than or equal to this threshold in order for it to be sent to the host system).

iv) The number of search statements for queries containing large numbers of terms is quite high, even when pre-searches are invoked. The size of queries used with the intelligent terminal system may need to be limited, the maximum size depending upon the response time of the host retrieval system.

Before ending this close examination of the concept of pre-searching further aspects of the work require highlighting:-

i) the limitation on query size is not a serious drawback. Firstly, Holmes (1978) reports the results of surveys made of the use of on-line retrieval systems

from within the United Kingdom: the data suggests that a great majority of searches consist of less than eight terms. Secondly, large Boolean queries tend to contain many terms 'ORed' together: such terms could be treated as synonyms for a single concept within the retrieval process and the frequency weight derived from the postings figure obtained from the disjunction of them all.

ii) some of the pre-searching results are derived from small numbers of queries. The total number of queries of different sizes are shown in the results illustrated in figures 6.5 & 6.9 and appendix 4. The test collections used for the experiments contained just one or two queries of some sizes. Thus, some of our results need approaching cautiously.

iii) in our examination of the relationship between term frequency weights and the probability of term co-occurence we made the assumption that index terms are assigned to documents independently. This is not quite true (Van Rijsbergen (1977) examines the dependence of index terms and the relationship to co-occurence data). However, the fact that a low pre-search threshold weight leads to the generation of unsuccessful pre-searches lends some justification to making this basic assumption.

iv) further experimentation could be carried out with pre-searching methods. The effect of varying the limit on the size of pre-searches could be examined. In the experiments described here it was set to $|T_{sub}|-2$, i.e. two terms less than the size of the query statement to be 'blocked'. This ensured that a pre-search had some potential for blocking other search statements. But perhaps this potential is not quite enough: a lower limit of say, two or three terms may be better. A pre-searching algorithm which tried all pairs of terms first, then all triples, and so on, as it attempted to locate a successful pre-search, could also be tried. The present algorithm steadily adds terms to pre-searches as they fail.

6.5 Summary

In this chapter we have described the methods and algorithms required for the terminal to operate a particular 'parasite' retrieval mechanism. A number of algorithms were devised which allow the terminal to retrieve a ranked list of document descriptors. The relative efficiency of the algorithms has been presented.

The concept of 'pre-searching' was introduced as a means of increasing the efficiency of the process. The effects of pre-searching on the number of search statements which would need to be sent to a host retrieval system, has been examined through a series

of tests with some familiar test collections.

We are satisfied that we have an adequate method of implementing a retrieval mechanism, based on weighted search terms, on an intelligent terminal used in conjunction with an operational retrieval system. The following chapter presents a description of some software for the intelligent terminal, which implements the methods described.

CHAPTER 7

THE TERMINAL SOFTWARE

7.1 Introduction

The intelligent terminal is programmed to communicate with an on-line retrieval system and operate the weighted search term retrieval mechanism using the methods described in the previous chapter. The terminal's software is outlined here.

All programs were written in M6800 assembly language using utilities on the EXORciser development system for assembling and debugging programs (Motorola 1975a and 1975b). The algorithms which allow the terminal to operate the weighted search term retrieval mechanism (chapter 6) were initially programmed in BCPL (Richards 1974) and tested on a large computer (ICL 1904s). After logical errors in the programs had been removed and the programs operated correctly they were translated ma nually into M6800 assembly language. This tactic proved to be beneficial as the high level language programs were easier to debug and serve to document the assembly language versions after the translation process has been completed.

It was argued in chapter 5 that because of the modular structure of the terminal, the software operating within the two interface modules ('host' and 'user') can be independent of the parasite retrieval mechanism operating within the central processor. However, in order to expedite the development of a functional terminal, only

those facilities which were required for the operation of the weighted search term retrieval mechanism have been implemented.

The following section describes the on-line retrieval system that the terminal is programmed to communicate with. Sections 7.3 to 7.5 describe software associated with each of the three processor modules within the terminal; the user interface, the host interface and the central processor (see chapter 5). Only the main aspects of the software are reviewed, i.e. the program structure and some data structures.
Details of the physical means of communication between the processors within the terminal were given in chapter 5.

A broad evaluation of the terminal and aspects of the practical experience gained while using the terminal with the on-line retrieval system are reported in section 7.6.

7.2 The Host Retrieval System

The terminal is programmed to work with the MEDUSA information retrieval system running on an IBM 370/168 computer at the University of Newcastle Upon Tyne. The system was developed some time ago as a vehicle for experimental work in on-line information retrieval. It is not a commercial retrieval system. However, it does support Boolean search techniques in a similar fashion to many of the existing commercial systems and enabled the feasibility of the intelligent terminal approach to be examined without incurring the

burden of the high charges made by commercial organisations for the use of their systems. Barber *et al* (1972 and 1973) describe the MEDUSA system and the initial experiments carried out with it. Their aim was to compare the level of performance obtained from the system by trained searchers (intermediaries) and users searching alone. The conclusion was that the two retrieval performance levels were similar. A second experiment (Barraclough *et al*, 1975) examined current awareness needs of users. Users' interactions with the computer system were monitored over a period of eighteen months. The main results were concerned with the rate at which users changed their queries; the number of different queries they ran and the number of times the user returned to run his query.

The MEDUSA system was originally developed on an IBM 360/67 computer and subsequently, adapted to run on an IBM 370/168 machine. Citations were obtained from the producers of the MEDLARS database (the National Library of Medecine in America). The early experiments used citations published over a period of three months and the databases were continually updated. For the trials with the intelligent terminal described here one month's citations were used (approximately 15,000 entries) which dated back to 1974.

Instructions for using the MEDUSA system are reproduced in appendix 3. A search session begins with the user entering his query terms individually. The reply from MEDUSA indicates if the term is in the thesaurus. If a term does appear there, the retrieval system assigns a code number and informs the user of it. Once he

has entered all his query terms and noted their associated code numbers he is able to formulate search statements. The usual Boolean operators may be used but term codes must be specified rather than the terms themselves. Each search statement is introduced by the command 'COMBINE'. An example of a MEDUSA search statement is as follows:-

COMBINE M1 AND M2 OR M3 AND M4.

The codes M1 to M4 are those assigned by MEDUSA to query terms previously entered. The response to a search statement is yet another code number which is used to refer to the search statement in later stages of the dialogue. To display document descriptors which are retrieved by a search statement the user must enter the command 'SEARCH' followed by the code number for the statement. The reply to the SEARCH command can be either a list of document descriptors or an indication that no documents match the search specification. Other commands exist which enable the contents of the thesaurus to be examined and details of these may be found in appendix 3.

7.3 The User Interface

The interface is programmed to accept commands from the user, recognise their meaning and relay them on to the central processor in coded form. Similarly, coded messages received from the central processor are intercepted and displayed in a suitable format for

```
<command>::=<query command>|<host command>|<terminate
                                                command>

<query command>::=<query signifier><term list><end
                                                signifier>

<host command>::=<host signifier><host list><end
                                                signifier>

<terminate command>::= E<newline>

<query signifier>::= Q<newline>

<host signifier>::= H<newline>

<end signifier>::= ↑A (control and 'A')

<term list>::=<search term><newline>|<search term>
                                    <newline><term list>

<host list>::=<string><newline>|<string><newline>
                                                <host list>
```

Figure 7.1. Terminal Command Language Specification.

the user to view.

The functions required of the interface software for the operation of the weighted term retrieval mechanism are:-

i) allow the user to input his query (a list of search terms).

ii) allow the user to communicate directly with the host retrieval system - enabling him to browse through the thesaurus for example, or to log in to and out from the host computer system.

iii) display document records which have been retrieved from the host retrieval system.

iv) allow the search session to be terminated.

The next section describes the command language while others discuss how the interface software meets each of the requirements listed above.

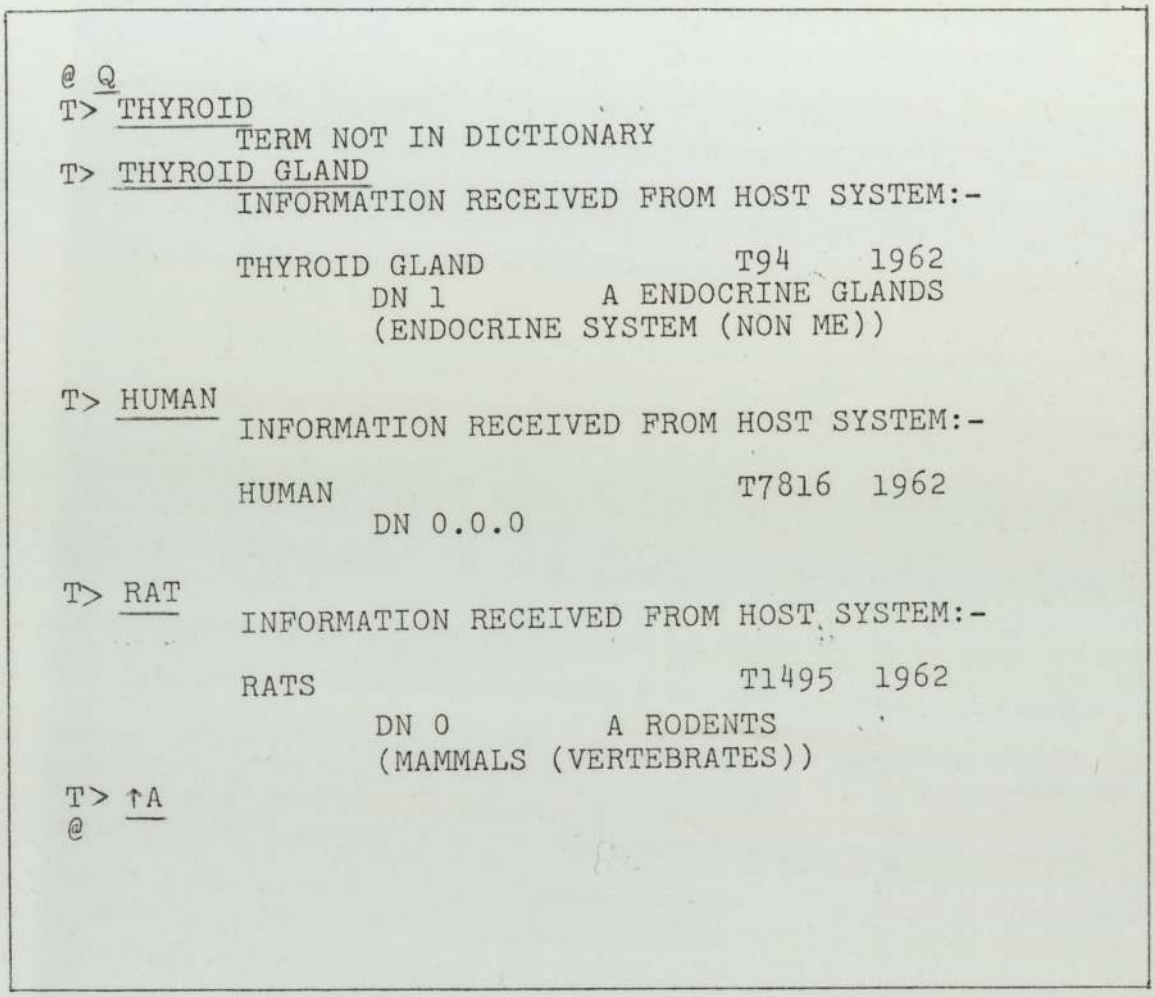
7.3.1 Command Language

A very simple command language has been implemented - each command consists of a single character followed by any parameters it may require. Figure 7.1 defines the command language in BNF

notation. For example, the character 'Q' indicates that the user wishes to enter his query. The query, which is simply a list of search terms, is entered by typing each term followed by carriage return. The end of the query is indicated by the user typing the control and A characters simultaneously. The 'H' character indicates that direct communication with the host retrieval system is desired while 'E' indicates that the search session is to be terminated. The command language could be made more pa latable very easily with the further application of programming effort. However the feasibility of the intelligent terminal approach can be tested with the simple command language described. The operation of the software for each command is described in the following sections.

7.3.2 Entering a Query

The character, 'Q', indicates that the user wishes to input his query. He enters each search term on a new line. After each term a report is displayed by the terminal which informs the user if the term is in the host system's dictionary. Any other information received from the host is also shown. For the current host system (MEDUSA) this includes related terms taken from the MeSH (Medical Subject Headings) thesaurus. The user must wait for this reply from the terminal before entering the next search term. During this waiting period the terminal has to transmit the search term to the host and receive the reply.



'@' and 'T>' are user prompts issued by the user interface.

Characters typed by the user are underlined.

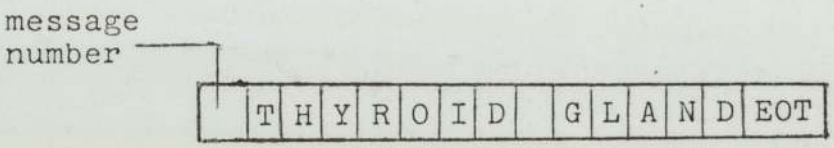
Figure 7.2. A Typical Dialogue between the User and the Terminal as a Query is Entered.

Figure 7.2 illustrates a typical dialogue between the user and the terminal. The corresponding messages exchanged by the interface and the central processor are illustrated in figure 7.3. If a term is found not to be in the host's dictionary a single byte message is received from the central processor. Upon termination of the query the interface passes a single byte message to the central processor (the central processor then begins the retrieval process). Actual message numbers and their meanings are listed in appendix 2.

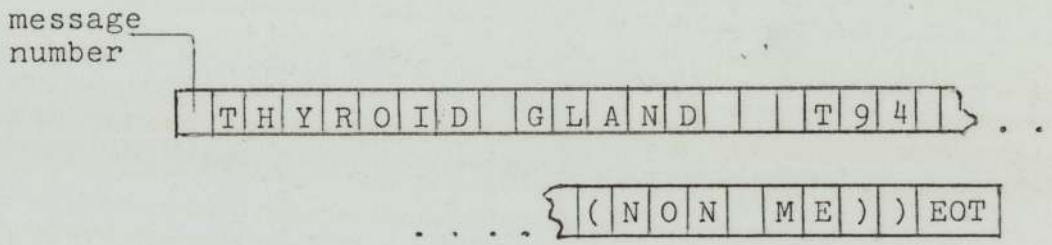
7.3.3 Direct Communication Between the User and Host Retrieval System

Following the receipt of the character 'H' all characters typed by the user are passed on to the central processor for transmission to the host system. The appropriate message number is attached. The user may wait for a reply from the host or continue on a new line with a further string of characters. The characters are passed to the central processor on a line by line basis - each line forming a separate message. This mode of interaction with the terminal is terminated by the control and 'A' characters typed simultaneously. The message structures are indicated by figure 7.4. Only messages passed to the central processor are shown since the messages received from the central processor have exactly the same structure.

This command can be used at the start of a search session to log in to the host system, during a search session to browse through



i) to Central Processor



ii) from Central Processor

Figure 7.3. Messages Exchanged between the User Interface and the Central Processor as a Query is Entered.

the thesaurus prior to entering a query and at the end of a search session to log out from the host system.

At any time during a search session the terminal may receive 'broadcast' messages from the host. These are often initiated by the system operators to notify users of impending system closedown. Any such messages, recognised by the host interface, are received by the user interface via the central processor. Upon receipt of the message the user interface displays the contents to the user. This form of direct communication between host and user can occur at any stage during the time the terminal is logged in to the host system. The user need not be using the 'H' command. The message structure is the same as that illustrated in figure 7.4.

7.3.4 Termination of the Search Session

The user terminates the search session by typing the character 'E'. A single byte message is passed to the central processor (which terminates the retrieval process). The user is now able to log out from the host system or begin a new query.

7.3.5 The Display of Document References

Messages containing document records are received from the central processor as they are retrieved from the host system. The document records are displayed to the user immediately. The structure of the messages are quite simple: a single character

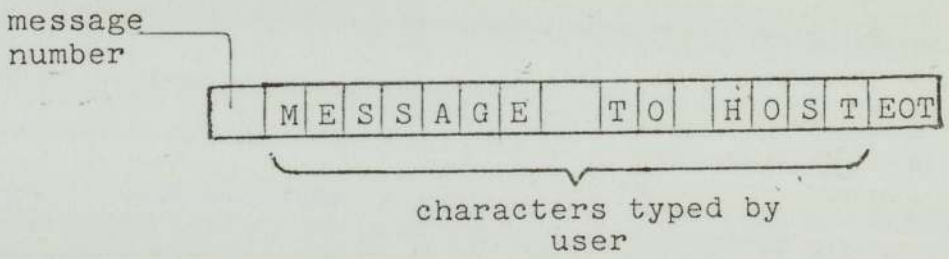


Figure 7.4. Structure of the Messages Transmitted to the Central Processor by the User Interface when the Searcher wishes to Communicate Directly with the Host Retrieval System.

string which includes title, authors and keywords. A more desirable arrangement is to have the individual fields of the document records identifiable by the user interface, through the inclusion of field separators within the messages arriving from the central processor. This would enable the interface to generate a display which was independent of the format used by the host system.

7.4 The Host Interface

The host interface has to carry out the following tasks:-

i) accept requests for data from the central processor and forward them to the host retrieval system in an appropriate format.

ii) intercept replies from the host system, isolate the required data and relay it to the central processor.

Since only those functions which are pertinent to the operation of the weighted search term retrieval mechanism have been implemented we have the following list of requirements for the host interface:-

i) to establish postings figures for terms specified by the central processor.

ii) to transmit a query (a conjunction of search terms) and retrieve matching document descriptors.

iii) to transmit a query and simply determine if there are any matching document descriptors (no retrieval is required) - a 'pre-search'.

iv) to allow direct communication between the user and the host retrieval system.

v) to report unsolicited or broadcast messages received from the host, to the user.

Each function will be considered separately.

7.4.1 Establishing Postings

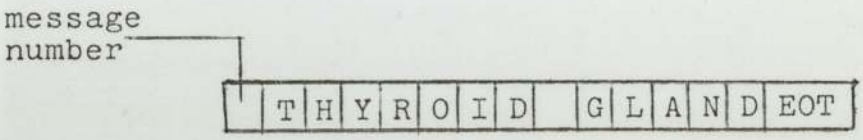
The operations required are:-

i) receive request for postings and the search term in question from the central processor,

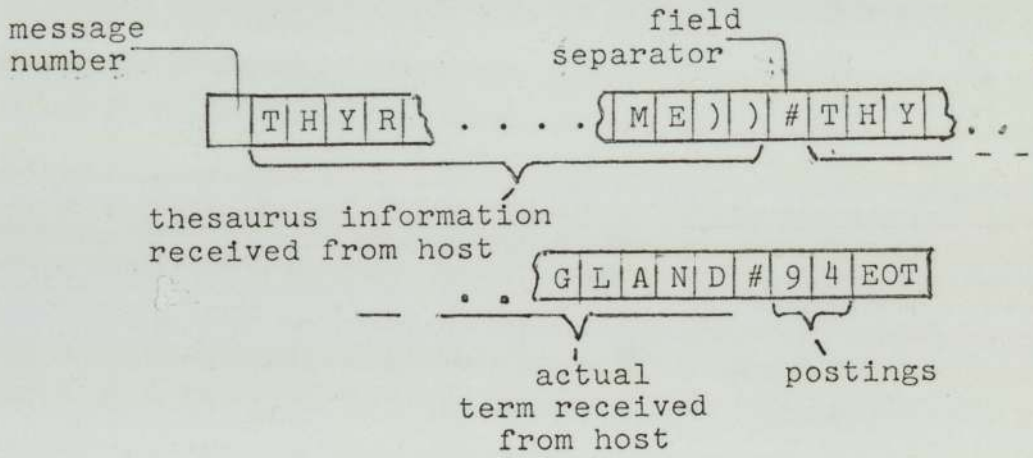
ii) output the term to MEDUSA,

iii) receive reply from MEDUSA and isolate the postings figure (including recognition of a 'no postings' reply),

iv) forward the postings figure on to the central processor.



i) from Central Processor



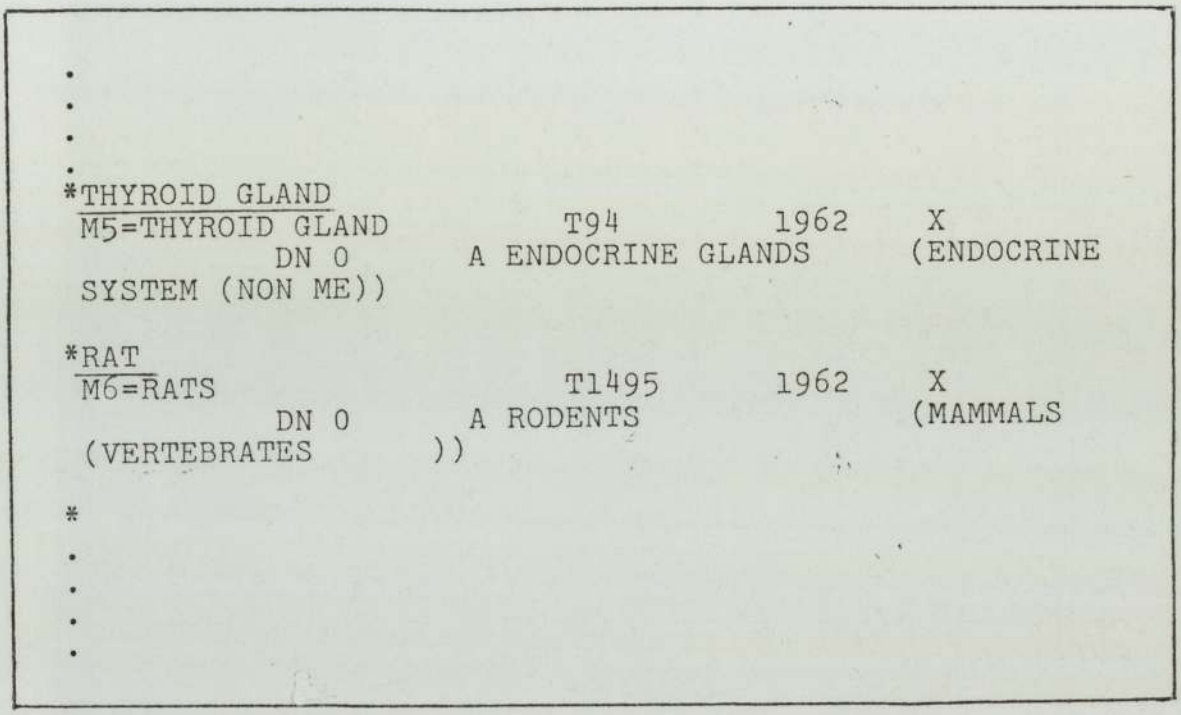
ii) to Central Processor

Figure 7.5. Messages Exchanged by the Central Processor and Host Interface when Establishing the Postings Figure for a Search Term.

The structure of the messages used to communicate with the central processor are shown in figure 7.5 and an example of the dialogue invoked with MEDUSA is given in figure 7.6.

The message number received from the central processor indicates a postings request. The host interface simply transmits the character string received after the message number and terminated by an 'end of transmission' character, to MEDUSA. On receipt of the reply from MEDUSA the interface distinguishes between standard replies for a term which is not in the MEDUSA dictionary (a term which does not index any documents) and a term which is in the dictionary.

If a term is not found to be in the dictionary, MEDUSA reports the fact by returning the string 'NOT IN DICTIONARY'. In this case the message relayed to the central processor by the interfacing module consists of one byte only containing the message number, 5. The reply for a term which is in the dictionary is more complicated. Firstly, the term actually returned by MEDUSA and the associated postings figure are isolated from the rest of the reply. The remaining part of the MEDUSA reply details thesaurus information pertaining to the term, eg related terms. The thesaurus information is passed on to the central processor intact (which in turn relays it to the user interface). The term returned by MEDUSA follows the thesaurus information and this may differ from the original term specified by the user. This occurs when the user's query term is not found in the thesaurus. Before reporting that the term was



'*' is a prompt issued by MEDUSA

The output from the terminal is underlined

Figure 7.6. A Typical Dialogue between the Terminal and the MEDUSA Retrieval System when Establishing the Postings Figure for a Term.

not found in the dictionary, MEDUSA attempts to return a synonym of the user's term which is in the dictionary. The central processor may need to be aware of the synonymy and is therefore informed of the term returned by MEDUSA. For the sake of simplicity the term returned by MEDUSA is forwarded to the central processor in all cases. The final part of the message passed to the central processor is the postings figure for the term, which is transferred as a character string (and not as an integer value).

The code numbers returned by MEDUSA for each term are not relayed to the central processor but are stored locally within the host interface. All references to terms within search statements despatched to MEDUSA must be made using the codes. A table (called TERMTABLE) is used to record the codes - figure 7.7. The ordering within the table corresponds to the order in which terms are received from the central processor. Terms which are reported as not existing within the MEDUSA dictionary are not given entries in the table and are ignored once reported to the central processor. The central processor is unaware of the existence of term codes. The constituent terms of queries to be sent to MEDUSA during later stages of the dialogue are communicated to the host interface by the central processor, using bit strings. This process is described in the following section.

TERMTABLE

M	1	'_'	'_'
M	2	'_'	'_'
.			
.			
.			
M	3	5	'_'

'_' = space character

Figure 7.7. The 'TERMTABLE' used by the Host Interface to Store Search Term Codes returned by MEDUSA.

7.4.2 Processing Queries

Actions required of the host interface upon receipt of a query specification from the central processor are:-

i) receive query specification from the central processor.

ii) output conjunction of terms (codes) to MEDUSA.

iii) receive search statement code and estimate of size of retrieved set from MEDUSA.

iv) request all document descriptors from the retrieved set.

v) forward document descriptors to the central processor.

Figure 7.8 shows an example of a typical dialogue between the terminal and MEDUSA. The query specification from the central processor takes the form of a bit string, figure 7.9. The n^{th} bit of the string refers to the n^{th} term received by the host interface from the central processor during the earlier phase of the process when postings figures were being established. A bit set to one indicates the presence of the appropriate term in the query. The host interface refers to the TERMTABLE (figure 7.7) and outputs the appropriate term code to MEDUSA. The central processor is not aware of the existence of term codes. (Another host interface, designed for use with a different retrieval system, might use a table of actual terms).

output from terminal is
underlined.

a previous
search statement
which retrieved references

```
.
.
.
.
*COMBINE M1 AND M2 AND M3 AND NOT R2

 R4= M1 AND M2 AND M3 AND NOT R2
 EXPECTED RETURN : SMALL

*SEARCH R4

 SEP 73 CITATIONS

 FIRST CITATION FOUND IN 4 SECS

 CIT NUM 00143014
 LILLIBRIDGE CB SMITH AD
 Observations on the ultrastructure of cells in alcoholic
 fed dogs.
 Am J Dig Dis VOL18   443-54 Jun 73
 *ALCOHOL, ETHYL                *GASTRIC MUCOSA
 DOGS                           MICROSCOPY, ELECTRON
 .
 .

 CIT NUM 00143020
 RIDLEY PT
 .
 .
 .
    etc.
 .
 .
 .
 .

*
```

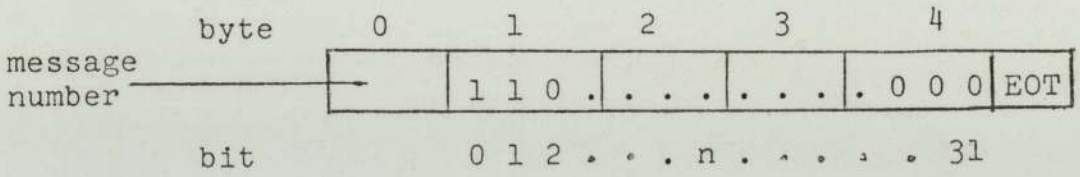
Figure 7.8. A Typical Dialogue between the Terminal and MEDUSA when Processing a Search Statement.

Having received a query, MEDUSA returns the search statement code and an estimate of the size of the retrieved set. A second command must be sent to MEDUSA which will either retrieve actual citations or elicit an indication that there are no hits. If no document descriptors are retrieved by the query a one byte message is returned to the central processor. Retrieved references are processed as they are received by the host interface. Individual document descriptors are isolated and forwarded to the central processor.

Search statement codes of successful searches are recorded by the host interface. Document sets represented by each of these search statements must be removed from all document sets which satisfy queries occurringlater in the dialogue. This is achieved by negating (with 'AND NOT') each successful search statement code in all later queries sent to MEDUSA - see figure 7.8. Such statement codes are recorded in a table similar to that which contains term codes.

7.4.3 Processing Pre-Searches

Chapter 6 explains the concept of a pre-search: it is a query, consisting of a small number of terms, for which the central processor requires only to be informed whether it retrieves any documents. The document descriptors, if any, need not be retrieved. Actions required of the host interface are therefore similar to those, described in the previous section, required for processing queries. The actions are:-



n^{th} bit indicates inclusion or exclusion of search term n

'1' = inclusion

'0' = exclusion

Figure 7.9. The Structure of a Query Specification from the Central Processor.

i) receive pre-search specification from the central processor.

ii) output conjunction of terms to MEDUSA.

iii) receive search statement code and estimate of the size of the retrieved set from MEDUSA.

iv) request documents from the retrieved set.

v) receive and recognise first document descriptor or 'no documents retrieved' message.

vi) if documents are returned by MEDUSA curtail output by transmitting 'BREAK'.

vii) inform central processor of result.

The central processor is informed of whether documents were retrieved by means of a single byte message. Upon receipt of 'BREAK', MEDUSA stops transmitting characters at the end of the current document descriptor. We should point out that the method of establishing if there are any documents in the database which satisfy a particular query would be far less cumbersome in most other on-line retrieval systems. The majority of other systems report the exact number of hits after receiving a search statement and consequently there would be no need for actual document descriptors to be retrieved. Search statement codes associated with pre-searches are not recorded by the host interface (as they are for actual queries) because citations in the retrieved sets are not processed.

7.4.4 Direct Communication between the User and the Host Retrieval System

The user may communicate directly with the host system. This facility allows the user to examine the thesaurus, for example. The actions to be carried out by the host interface are quite straightforward:-

i) to receive user communication from central processor,

ii) forward complete message to MEDUSA.

iii) receive reply from MEDUSA,

iv) forward complete reply to central processor (for transfer to the user interface module).

The structure of the messages passed between the processor within the terminal are quite simple - figure 7.4 . The first byte contains the message number as always, and indicates the message type. The remaining bytes up to the 'end of transmission' character contain the character string to be sent to MEDUSA, or the character string received from MEDUSA.

7.4.5 Broadcast Messages from the Host System

At any stage during the dialogue the host computer may issue 'broadcast' messages. These may originate from the system's

operators for example, giving notice of impending closedown. Any messages received from MEDUSA which are 'unsolicited' are forwarded to the central processor for transfer to the user interface. The messages have the same structure as those depicted in figure 7.4.

7.5 The Central Processor

The operation of the 'parasite' retrieval mechanism is carried out by the central processor module. This section describes the software which enables the central processor to execute the algorithms described in chapter 6. The requirements are:-

i) to receive the query from the user interface and calculate term weights, i.e. request postings figures for query terms from the host retrieval system via the host interface.

ii) to initiate the retrieval of documents from the host in the correctly ranked order.

Also it has:-

iii) to support direct communication between the user and the host retrieval system.

Descriptions of the structure and contents of the various inter-processor messages have been given in the two previous sections. Consequently we shall say little about them here:

instead, we shall concentrate on the operation of the software.

7.5.1 Receipt of the User's Query and Calculation of Term Weights

Search terms are received by the central processor from the user interface module individually, as soon as the user has completed typing each one into the terminal. Upon receiving a query term the central processor passes it to the host interface for transmission to the host system in an appropriate format. The host interface eventually returns a postings figure for the term in the form of an ASCII character string or indicates that the term is not part of the host's thesaurus. In the latter case the central processor indicates the result to the user interface, ignores the term and waits for the arrival of the next search term. If a postings figure is returned by the host interface, the character string is converted to an integer value and stored in a table. The host interface also returns other information which it receives from the host system in answer to the request for a postings figure. This forms another field within the message received by the central processor. The information contained within the field is passed on to the user interface which in turn, displays it to the user.

While the searcher is digesting data displayed to him by the user interface relating to a search term, and is typing in his next term, the central processor calculates the term weight. It uses the postings figure received from the host interface and a figure for the total number of documents in the host's document file

which is (presently) written into the software. Real arithmetic is realised using a set of floating point routines written in M6800 assembly code which were provided with the EXORciser development system as part of a software utility library. Once calculated, the term weight is added to a table. Each term is represented by a term identifier. The actual term itself is not stored anywhere as the central processor assumes that the host interface has a record of it. The term identifiers are integer values beginning with zero and increasing in steps of unity. The numbers simply represent the order in which the user inputs his query terms (ignoring those terms which do not appear in the host's thesaurus). A single byte message arriving from the user interface indicates that the user has completed entering his query. The central processor, upon receiving this message, begins to generate the ranked list of document descriptors.

7.5.2 Retrieval of Documents

Once term weights have been established, the terminal proceeds to retrieve the ranked list of document descriptors using the algorithms described in the last chapter for generating the subsets of terms and pre-searches. The search terms are sorted according to their weights and the 'knapsack algorithm' is entered.

Queries and pre-searches are represented by bit strings. Bit position 'n' indicates the inclusion or exclusion of term 'n' in, or from, a subset of terms (which can be an actual query, or a

pre-search). The digit, '1' indicates inclusion and '0', exclusion. A bit string is 4 bytes in size so that up to 32 terms may be accommodated. Specification of a query or pre-search to the host interface processor is done via these bit strings. The host interface software compiles the queries/pre-searches for transmission to the host system (section 7.4). Successful pre-searches (those which do not retrieve documents) and unsuccessful pre-searches (those which do retrieve documents) are stored in tables (as bit strings). As queries are produced they are checked against the list of successful pre-searches. If any successful pre-search is a subset of the query then the query need not be forwarded to the host as it will not retrieve documents; the query has been 'blocked'. As pre-searches are produced they are checked against the list of unsuccessful pre-searches, so that such pre-searches are not repeatedly transmitted to the host system. The bit string format for queries and pre-searches is very convenient as the operations just described, eg comparing a query specification against the list of successful pre-searches, can be handled very efficiently using the assembler instructions available on the M6800 processor.

If the user interface indicates that the searcher wishes to terminate the search session an interrupt is generated within the central processor. An interrupt handling routine picks up the message from the user interface and sets a flag. The flag is regularly checked within the main program so that processing ceases as required.

A degree of parallelism is present. Once a query or pre-search has been despatched to the host interface the central processor continues to find the next query or pre-search specification whilst waiting for a reply. Once the next query or pre-search is ready to be sent to the host interface the central processor waits for the reply to the previous request (which the host interface may already be waiting to transfer). This action exploits the architecture of the intelligent terminal. A considerable amount of work can be carried out by the central processor while the host interface module processes the previous request. For example, many term subsets may be generated and blocked by existing successful pre-searches in the meantime. A number of flags are used to control the transfer of queries or pre-searches to the host interface and the processing of replies to them.

7.5.3 Direct Communication between User and Host

It has already been pointed out that the terminal will allow the searcher to communicate directly with the host retrieval system. Also, that any broadcast messages received from the host are displayed to the user. Since there is no direct communication link between the host and the user interfaces any message from one to the other has to travel via the central processor. The central processor therefore recognises such messages and simply transfers them byte by byte, from one input/output port to the other.

7.6 Evaluation of the Terminal

To gain an idea of the effectiveness of the terminal hardware and software, a small number of searches were conducted using the terminal in conjunction with the MEDUSA retrieval system. The searches are taken from a set which were used in early experiments with MEDUSA. Oddy (1974) has also made use of them in experiments with his browsing system.

We shall consider each of the processor modules within the terminal separately and then consider the terminal's architecture as a whole.

7.6.1 The User Interface

SOFTWARE

Only a very basic interface has been implemented. Our concern was not with providing a 'user-friendly' interface which was robust enough to withstand mistreatment and have good error reporting or help facilities. Instead we wished only to test the feasibility of the intelligent terminal approach to implementing a weighted search term retrieval mechanism. The interface we have described was adequate for that task.

The memory requirements of the software are:-

program (read only memory - ROM): 1.0K bytes

data (read/write memory - RAM):.75 K bytes

HARDWARE

The processor hardware configuration (chapter 5) is adequate. The memory provision matches the requirements of the software and the execution of each command is carried out within satisfactory time periods (ie there is no detectable delay in processing any of the commands).

7.6.2 The Host Interface

SOFTWARE

The memory requirements of the host interface software is as follows:-

program (read only memory - ROM): 1.5 K bytes

data (read/write memory - RAM): ·75K bytes

The software met the stated requirements.

A difficulty came to light during the trials which was not accounted for during the design stages: that is the problem of line noise within the telephone link between the terminal and the host retrieval system. In all, the effects of line noise caused the terminal to fail twice during trials (excluding instances during

system tests). Reasons for the resulting failure are obvious. The host interface has to indentify data items in the replies from the host system. If line noise corrupts a character string then the interface can fail to recognise the presence of a sought after data item. This problem has to be overcome if the terminal's reliability is to be made acceptable for an operational environment. The difficulties caused by line noise may be worse than we have portrayed here, for the following reasons:-

i) our trials are limited in scope (only 10 searches were conducted).

ii) other parasite retrieval mechanisms may require finer details of messages received from the host system to be recognised, eg individual index terms.

A number of methods for overcoming this problem can be suggested:-

i) if noise is detected by the host interface then it could repeat the inquiry to the host system.

ii) the searcher could be enlisted to help resolve any difficulties that the host interface may have.

iii) techniques for matching character strings which allow for the loss or the corruption of individual characters could be employed. (Wagner & Fischer 1974 and Hall & Dowling 1980).

Probably more than one of these possibilities would be required.

HARDWARE

Conclusions are similar to those for the user interface hardware. The hardware configuration is adequate and execution speeds satisfactory.

7.6.3 The Central Processor

In this section we shall consider the operation of the parasite retrieval mechanism, ie weighted search term retrieval and ranked output. A number of searches were conducted and the number of search statements required to retrieve portions of the ranked list of document descriptors were recorded as in the experiments with test collections described in chapter 6. The results are shown in figure 7.10. The pre-search threshold weight was set at a low level. (We did not examine the effect of varying its value at this stage). The queries, taken from those used in earlier experiments with the MEDUSA retrieval system, contain between six and twelve terms. Smaller queries were ignored as it has already been shown that they can be adequately handled while larger queries were discarded as these have been shown to generate too many search statements already (chapter 6).

No. DOCS		QUERY SIZE (and population)						
		6	7	8	9	10	11	12
		2	1	4	1	1	0	1
1	S	2	3	5	4	1	-	2
	P	10	15	19	21	32	-	15
	T	12	18	24	25	33	-	17
5	S	12	5	15	11	6	-	9
	P	13	17	24	28	45	-	33
	T	35	22	39	39	51	-	42
10	S	13	15	21	20	17	-	15
	P	13	19	24	32	48	-	56
	T	26	34	45	52	65	-	71
15	S	17	16	31	21	42	-	30
	P	13	19	24	32	49	-	73
	T	30	35	55	53	91	-	103
20	S	18	19	32	29	56	-	?
	P	13	19	24	32	49	-	?
	T	31	38	56	61	105	-	?
25	S	19	20	33	35	?	-	?
	P	13	19	24	32	?	-	?
	T	32	39	57	67	?	-	?
50	S	22	28	37	48	?	-	?
	P	13	19	24	32	?	-	?
	T	35	47	61	80	?	-	?

S = no. of true searches

T = total

P = no. of pre-searches

Figure 7.10. Results of Trials with the MEDUSA Retrieval System: The number of search statements required to retrieve proportions of the ranked list of document descriptors.

A comparison of these results with the results reported in the previous chapter for the test collections indicates that fewer search statements are required to retrieve similar proportions of the ranked list of document descriptors. This may be due to the larger number of document records in the subset of the MEDUSA collection which was made available for the trials (approximately 15,000 records). However, we cannot draw concrete conclusions due to the limited evidence we have presented. Actual timings were not considered worthwhile recording since the terminal was not communicating with an operational retrieval system but only a computer offering a general purpose computing facility. The numbers of search statements listed in figure 7.10 are an indication of the time required to perform the retrieval operation. If the average response time of an on-line retrieval system is taken to be 5 seconds then, for a query of ten terms, it would take approximately 5.5 minutes to retrieve the first ten document references. This is quite acceptable. Again, it has to be pointed out that the totals listed in figure 7.10 are based on small numbers of queries so they must be treated cautiously. It is felt however, that they prove the feasibility of the terminal operating a weighted search term retrieval mechanism for queries containing less than some maximum number of search terms, eg 10.

The memory requirements are:-

program (read only memory - ROM): 8K bytes

data (read/write memory - RAM): 2 Kbytes.

The calculation of log (x) in determining term weights is quite slow, taking nearly 5 seconds for each value. This is due to the complexity of the floating point package supplied by the Motorola company as part of a utility library. Improvement here is desirable.

HARDWARE

The execution times of the algorithms were adequate (except for the calculation of log (x)). The EXORciser development system formed the central processor module within the intelligent terminal. This system was able to meet the memory requirements of the software.

7.6.4 The Architecture of the Terminal

Although the architectural modularity of the terminal was not fully exploited in the design of the software (ie the host and user interface software is not independent of the parasite retrieval mechanism) the parallelism which resulted from the multiprocessor configuration was beneficial. While the central processor is waiting for a reply to a search statement from the host interface it is able to determine the next search statement to be sent to the host. Also, the calculation of a term weight could be carried out by the central processor while the user is viewing the response from the host system to the term and is typing in the next one.

A modification to the terminal architecture which would lead to greater efficiency is to place a direct link between the user and host interface modules. This would relieve the central processor of the burden of transferring messages from one interface to the other at the times when the user wishes to communicate directly with the host system or the host initiates a 'broadcast' message which is to be displayed to the user.

7.7 Summary and Conclusion

The terminal's software has been described which allows it to operate a weighted search term retrieval mechanism in conjunction with the MEDUSA retrieval system. Although the software is quite basic, especially the user interface, and only a small number of searches were conducted using the terminal, we are nevertheless able to demonstrate the feasibility of this approach to implementing weighted search term retrieval in an operational environment.

CHAPTER 8

FINAL REMARKS AND CONCLUSIONS

We shall begin this chapter by making some general conclusions about the work described in this thesis as a whole. We reiterate some of the motivations, highlight the advantages of our approach and point to some drawbacks. The sections which follow the general discussion concentrate on particular aspects of the work. Section 8.2 considers the intelligent terminal and section 8.3 is concerned with the implementation of the weighted term retrieval mechanism. Finally, we suggest some directions for future work to take.

8.1 General

The work described here grew from the initial premise that research in an area such as computerised bibliographic information retrieval needs eventually to influence the kind of retrieval facilities made available to users of information systems. The observation was made that research information retrieval was failing to do this. Reasons for this failure have been put forward.

One means of influencing the designers of computerised retrieval systems is by providing evidence that certain retrieval methods yield improved performance or show advantages in other ways, e.g. are easier to use, over the traditional Boolean search techniques. Such evidence can only be established by actually experimenting with

retrieval techniques in an operational environment. It has been demonstrated how the cheap processing power made available by microprocessors allow this to be done economically. Two points need to be emphasised.

Firstly, we are referring to retrieval methods. That is, techniques for allowing the user to express his 'information need' and algorithms operated by the computer to determine which document descriptors shall be retrieved in response to the specification of that information need. We do not wish to refer to studies which examine aspects of on-line searching such as the role of the intermediary or searching costs: studies like these have been carried out in operational environments already, but have little to say about retrieval algorithms. The second point is that our emphasis has been placed upon the need for an experimental tool which enables retrieval methods to be tested on a large scale, under realistic operating conditions. There is still a need for laboratory testing as researchers toy with new ideas. It would be ridiculous to suggest that all experimentation should be carried out on the scale we have described. It is more reasonable to suggest that only when laboratory tests have shown promising results should the methods undergo full scale trials.

The retrieval technique chosen to demonstrate the feasibility of the proposed experimental tool - the intelligent terminal - is the one that is ripe for large scale testing. Retrieval based upon

weighted search terms has received much attention from researchers in the past and there is evidence of good performance in laboratory tests. The work described simply demonstrates that an intelligent terminal used in conjunction with an operational retrieval system is capable of operating a particular method of retrieval and allows evaluation of that method within a realistic operational environment. We have not considered how large scale experiments should be conducted. There are many difficulties: examples are the measurement of recall and the effect of human factors, such as the role played by the search intermediary (who usually operates a terminal on behalf of the user of an on-line retrieval service). Again, if two retrieval systems are being compared then one must decide whether the same query should be run on both systems. If so, it is likely that the user's relevance assessments for the second system used would be affected by the output of the first system, and this would have to be taken into account in the design of the experiment. If different queries are used with the two systems then one must ensure that the two populations of queries have similar characteristics. The problem of experimental design in real-life information retrieval testing is very complex. This thesis has been concerned solely with technical problems associated with making it possible for large scale evaluation to be carried out. But this is a prerequisite to any such evaluation.

We have also pointed out that even after conclusive evidence in favour of some retrieval technique has been gathered, system

designers may still be loath to introduce it into existing retrieval systems. This is due to the excessive load that it might place on the existing computer's resources, and to the fact that the financial investment in present systems must be recouped before more is spent on major changes in software and hardware. We pointed out that the intelligent terminal approach may once again help. Should a particular retrieval method be deemed worthwhile in an operational environment, that retrieval facility could be made available to users by providing them, or allowing them to provide themselves, with intelligent terminals and appropriate software. No significant changes would be needed in the existing retrieval systems and the financial investment they represent would not be lost. The only added expense would be for the terminal itself, a modest cost which could be borne by any user who wished to take advantage of a proven advance in retrieval technique. The question of whether existing computer configurations can cope with more complicated search procedures can still not be immediately answered. Queries based upon Boolean search logic can be handled very efficiently within the machine given appropriate file structures. The extra work required for other retrieval mechanisms is obvious but how expensive they are in terms of the demands placed on a machine's resources is unknown. An intelligent terminal operating a weighted term retrieval mechanism as we have described, is placing a greater burden on the computer than the user with an ordinary computer terminal. As we have shown, the intelligent terminal generates many search statements thus demanding more CPU time to process them. Although the terminal

was adequately accommodated by a large interactive computer (IBM 370/168) executing a retrieval program, the effect of many such terminals operating simultaneously with an existing retrieval system is not known. However, it is obvious that, on occasions, the response time will be degraded due to the number of search statements the retrieval system is asked to handle. Thus, it may be that an intelligent terminal can only justifiably be used as a research tool, that is, just one device connected to the operational retrieval system for the duration of a particular experimental project. The problem of retrieval systems coping with more complicated retrieval techniques, demanding greater processing resources, may only be solved by employing new machine configurations which exploit the developments in hardware - technology and architecture - that were described in chapter 3.

8.2 The Terminal

Although the terminal, in its present form, has been adequate to explore the feasibility of our proposals, it was nevertheless not possible to make it generally available to users of a retrieval system and allow them to employ a weighted term retrieval mechanism. There are a number of reasons for this. The software needs to be made much more robust and the hardware properly packaged. The 'EXORciser' development system has played the role of the central processor and it is necessary to replace this by appropriate hardware. The user interface is fairly basic, supporting only a simple command language:

there is poor error reporting and there are no help facilities to guide the user. Finally, the host interface is programmed to communicate with the MEDUSA retrieval system which is not now in use as an operational retrieval service.

It must be remembered that our original aim was not to produce a well packaged product for use by an operational service, or by naive users. Instead, we wished to test the feasibility of our proposal for a tool for experimental work. This we feel we have done.

The one aspect of the terminal which has not been fully evaluated is that of cost. Figure 8.1 lists the items of hardware required for the terminal (assuming that appropriate hardware is used to replace the EXORciser development system) and approximate costs. The cost for a basic system which excludes a floppy disk drive unit, but is nevertheless capable of operating a 'parasite' retrieval mechanism, is £2,300. This is not particularly expensive. The addition of disks would greatly enhance the capability of the system and enable software development facilities to be provided: the extra cost is approximately £1,000. Costs of hardware are continually decreasing, particularly semi-conductor memory chips so overall costs are unlikely to rise in future years (1981 on). Software costs have not been included as these are difficult to assess, but any further development work would need to take such costs into account. It is likely that they would be substantially higher than the costs for hardware.

	£
Visual Display Unit	800
3 Processor Boards (including memory)	1200
connectors, cables and cases etc.	300
TOTAL	2300
Dual Floppy Disk Drive	1000
TOTAL	3300

Figure 8.1. The Cost of the Intelligent Terminal.

8.3 The Implementation of the Weighted Search Term Retrieval Mechanism.

The algorithms for operating the retrieval mechanism based upon weighted search terms within the intelligent terminal appear to be adequate (see chapter 7). The only drawback is with regard to the number of search statements required to be sent to the host retrieval system in order to retrieve the ranked document list. This relates directly to the time taken to complete a search session and therefore the cost of the search (given the current charging policies of system vendors). The concept of pre-searching has been shown to be particularly effective in reducing the number of search statements generated. However, the results indicate that the size of a user's query should be limited to between 8 and 10 terms. The enforcement of such a restriction is purely dependent upon the level of search costs which can be tolerated. Above average costs could be justified within an experiment by referring to the value of the research results. Arguments were put forward in chapter 6 to show that a query size restriction is not a significant disadvantage. Studies have shown that the average size of user's queries is less than ten terms. Also, large Boolean queries tend to have many terms 'ORed' together. It was suggested that these can often be taken as representing a single concept and thus treated as a single search term, with several synonyms: the term frequency would be the postings figure for the disjunction of the terms. (Small scale exploratory work may be required to establish the nature of the effect this is likely to have on retrieval performance).

8.4 Future Work

There are a number of different directions which future activities may take. On a broad level other potential application areas for the terminal within on-line information retrieval (see chapter 4) could be explored. An exercise of this nature would require effort to be directed towards the design and development of software for use on the existing terminal hardware. Further investigation into how new hardware technology (see chapter 3) could be exploited to the benefit of information retrieval is another relatively broad avenue which could be followed.

An avenue more specifically related to the retrieval mechanism that has been implemented is the investigation of different methods for selecting pre-searches. Such experiments could be carried out on a mainframe computer using test collections similar to the work described in chapter 6. One possible algorithm is to select all pairs of terms from the search statement, then all triples and so on, up to some maximum pre-search size. (The present algorithm selects a pair of terms and, if they fail as a pre-search, i.e. they retrieve document references, adds another term to the pair to form a triple etc). Some preliminary work has suggested that it is possible to establish the value of a particular pre-search by calculating how many search statements it is likely to prevent from being sent to the retrieval system (if it is a successful pre-search) given the list of pre-searches which have already been used. However, it is unlikely that the computing effort required to

find all possible pre-search formulations, determine each of their potentials, and chose the 'best' one, would be justified by the returns i.e. any further reduction in the total number of search statements sent to the retrieval system.

A final possibility for future work is to conduct an experiment to evaluate a retrieval mechanism operating in a realistic environment. Our original justification for the terminal was to make such an experiment possible so it is a step that must follow soon. However, as we have already pointed out the terminal would need to be developed further to improve the quality and robustness of the hardware and software. Large scale evaluation of retrieval techniques is a necessity so this would be a most worthwhile path to take.

Appendix 1.

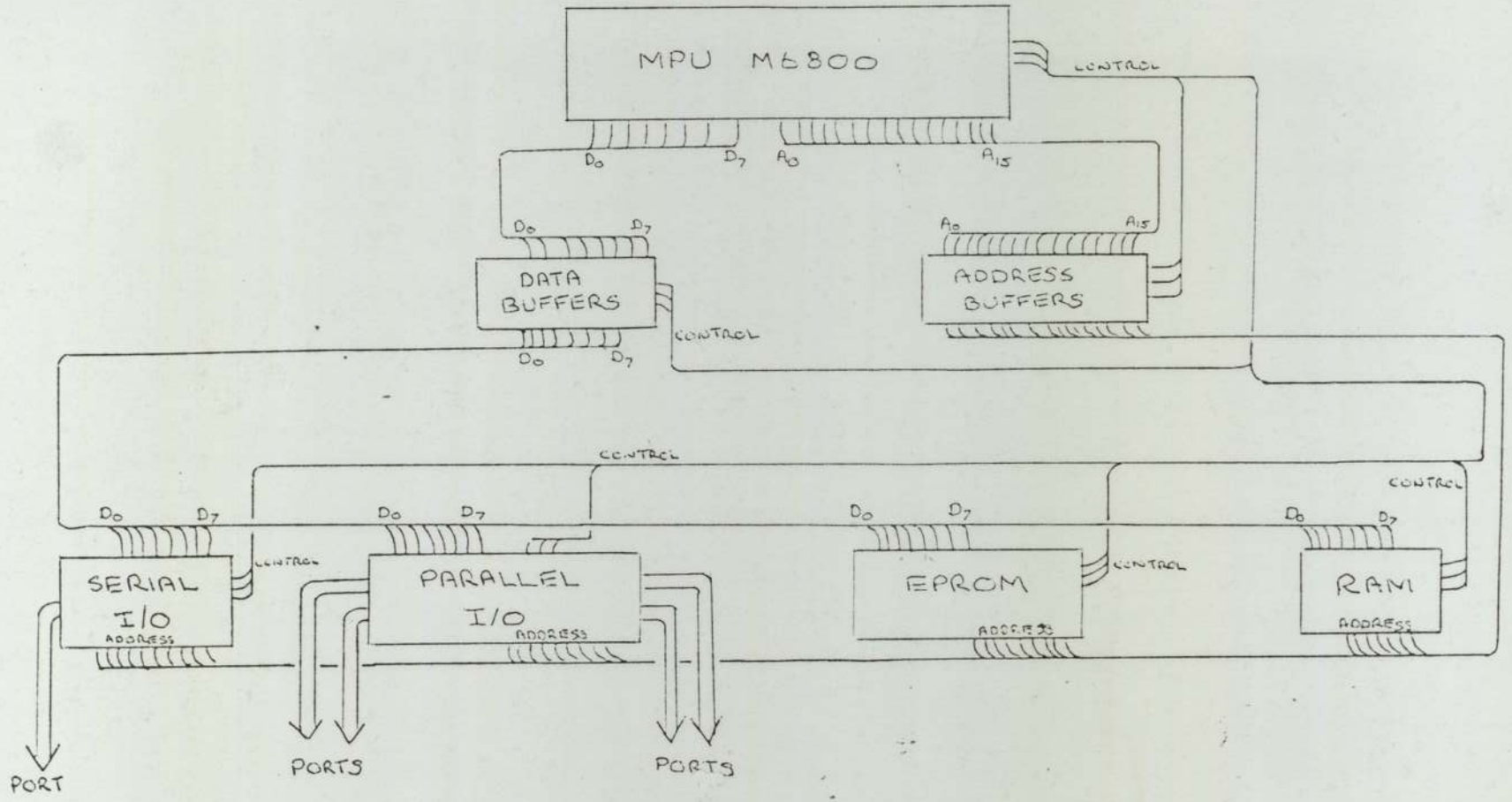
The Processor Hardware

The following pages document the microprocessor systems used for the user and host interface modules of the Intelligent Terminal (see chapter 5). The 'EXORciser' system (Motorola 1975a) played the role of the central processor. Each processor was constructed using 'wire-wrap connection' techniques and all constituent parts of one processor reside on one circuit board. (Therefore there are two boards in all).

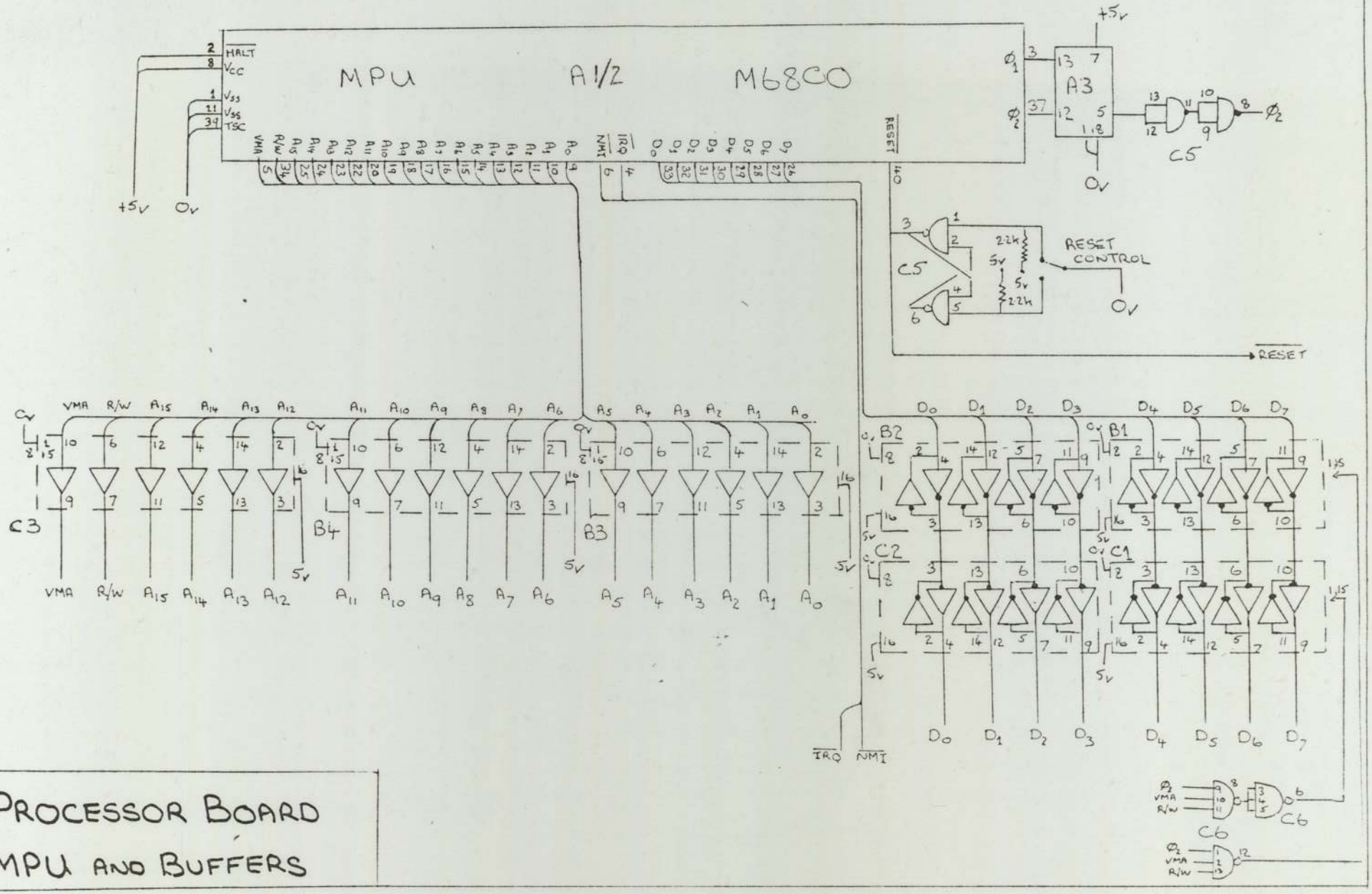
Hardware was also designed and constructed which interfaces the two processor boards to the EXORciser system. All the constituent parts again reside on one circuit board which fixes into the EXORciser's card-cage allowing access to the signals present on its backplane. The two processor boards connect to the interface board through flat ribbon connectors.

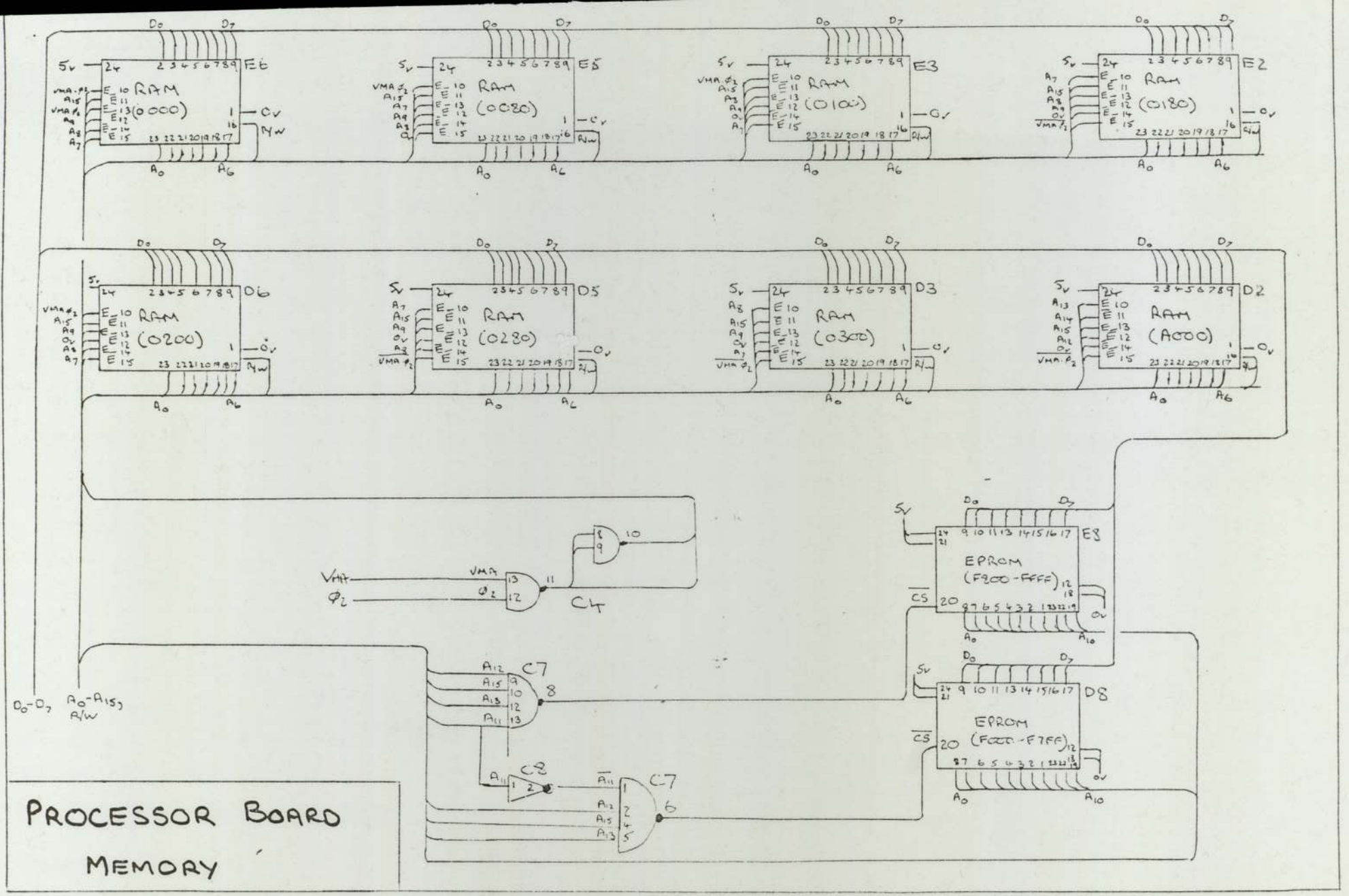
Each processor board provides:-

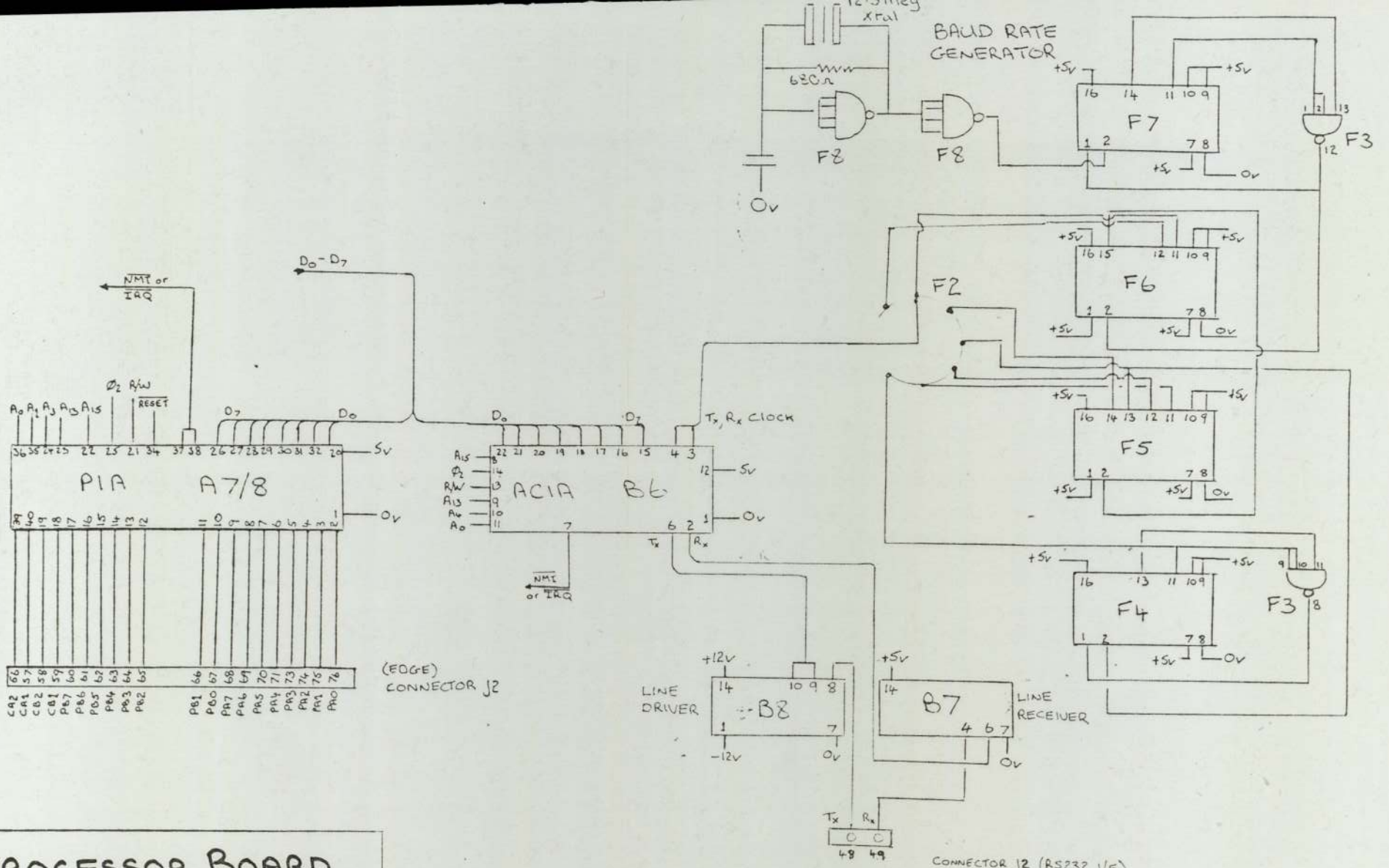
1024 bytes of random access memory (RAM)
4096 bytes of read only memory (ROM)
1 serial input/output port
2 parallel input/output ports
1 M6800 microprocessor
data and address line buffers/drivers
partial address decoding.



PROCESSOR BOARD BLOCK DIAGRAM


MPU A1/2 M6800
HALT
Vcc
Vss
Vss
TSC
VMA
R/W
IRQ
NMI
RESET
+5V
0V
A3
C5
RESET CONTROL
2.2k
C3
B4
B3
B2
B1
C2
C1
C6
IRQ
NMI
PROCESSOR BOARD
MPU AND BUFFERS

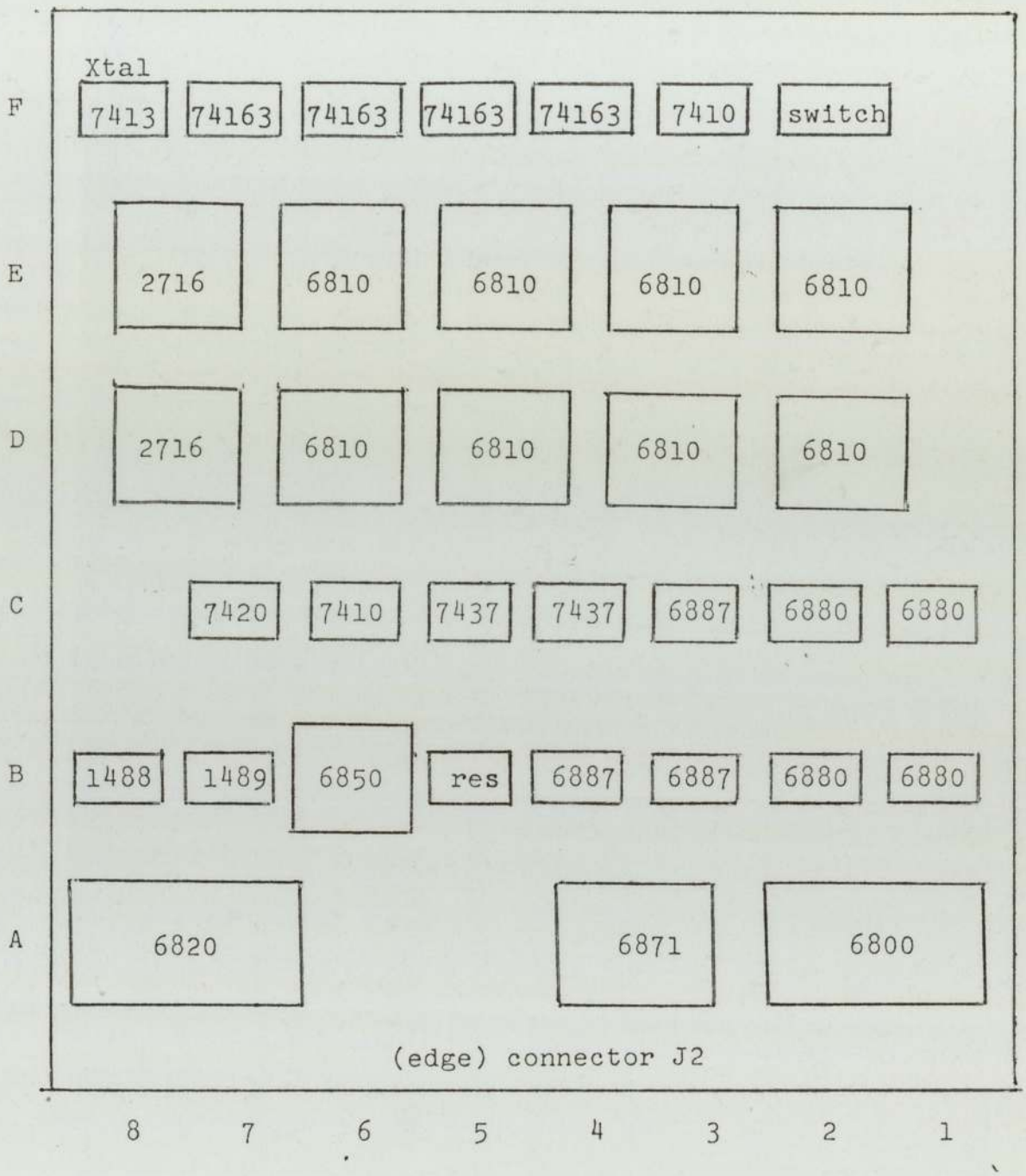

E6
RAM
(0000)
E5
RAM
(0080)
E3
RAM
(0100)
E2
RAM
(0180)
D6
RAM
(0200)
D5
RAM
(0280)
D3
RAM
(0300)
D2
RAM
(A000)
E8
EPROM
(F800-FFFF)
D8
EPROM
(F000-F7FF)
C4
C7
C2
C7
VMA
Φ2
D0-D7
A0-A15
R/W
PROCESSOR BOARD
MEMORY



PROCESSOR BOARD

PARALLEL AND SERIAL I/O

PROCESSOR BOARD LAYOUT



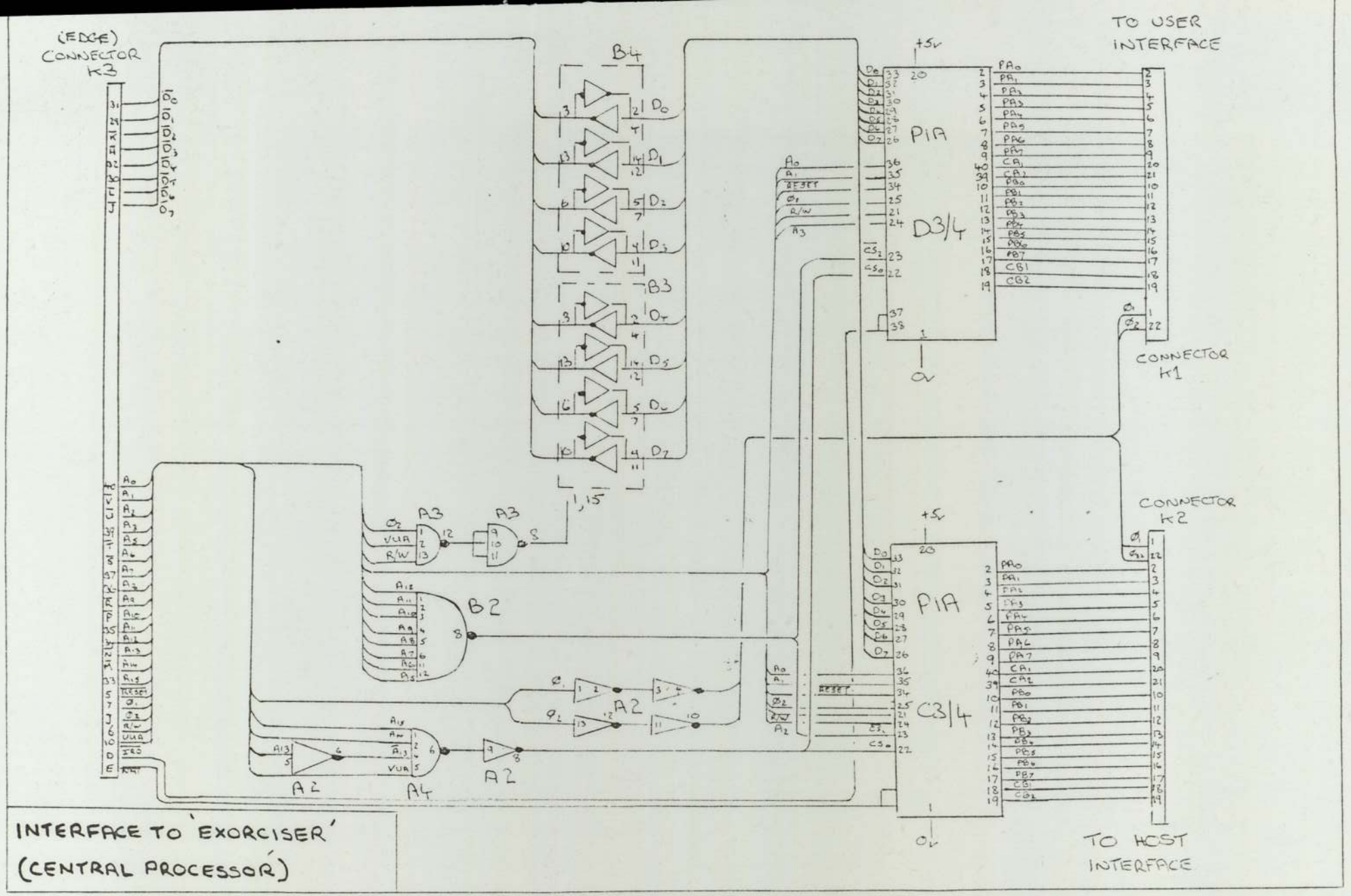
PROCESSOR BOARD PARTS LIST

A1/2	MC6800	MPU
A3/4	MC6871	Two phase clock
A7/8	MC6820	PIA (parallel interface)
B1,B2 C1,C2	MC6880	Bus transceiver (data buffers)
B3,B4 C3	MC6887	Buffer/inverter (address buffers)
B5	resistors	
B6	MC6850	ACIA (serial interface)
B7	MC1489	Line receiver
B8	MC1488	Line driver
C4,C5	SN7437	2-input NAND gates
C6	SN7410	3-input NAND gates
C7	SN7420	4-input NAND gates
D2,D3 D5,D6 E2,E3 E5,E6	MCM6810	128 byte static RAM
D8,E8	2716	2k byte EPROM
F2	dil switch	baud rate select (serial interface)
F3	SN7410	3-input NAND gates
F4,F5 F6,F7	SN74163	Synchronous 4-bit counter (serial interface)
F8	SN7413	4-input NAND gates
	Xtal	12.5 Meg crystal

PROCESSOR BOARD

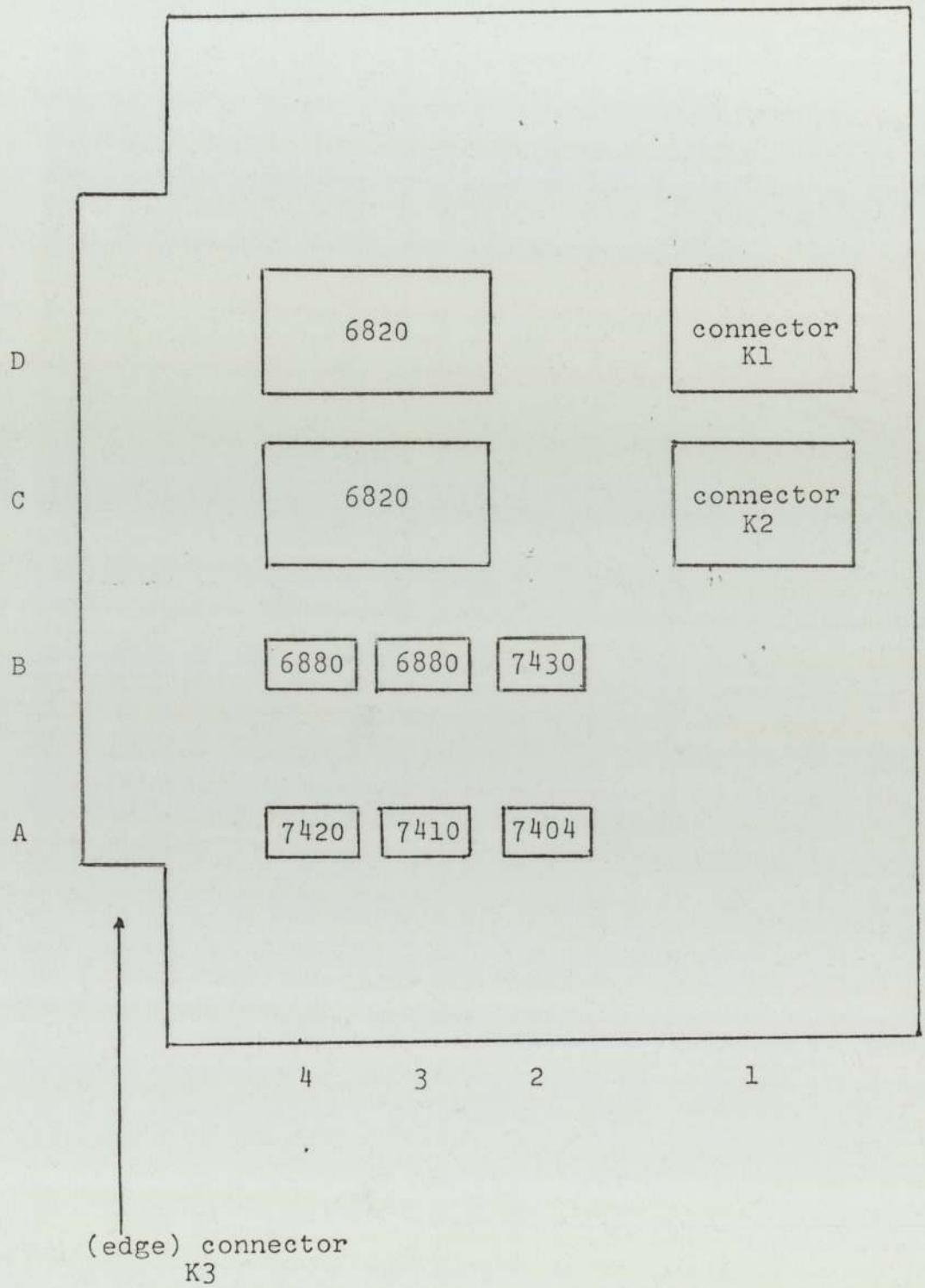
EDGE CONNECTOR J2

2	-	0v	
.			
.			
48	-	Tx data	serial interface
49	-	Rx data	
56	-	CA2	
57	-	CA1	
58	-	CB2	
59	-	CB1	
60	-	PB7	parallel interface port B
61	-	PB6	
62	-	PB5	
63	-	PB4	
64	-	PB3	
65	-	PB2	
66	-	PB1	
67	-	PB0	
68	-	PA7	parallel interface port A
69	-	PA6	
70	-	PA5	
71	-	PA4	
73	-	PA3	
74	-	PA2	
75	-	PA1	
76	-	PA0	
77	-	+5v	



INTERFACE TO 'EXORCISER' (CENTRAL PROCESSOR)

EXORCISER INTERFACE BOARD LAYOUT



EXORCISER INTERFACE BOARD PARTS LIST

A2	SN7404	Inverters
A3	SN7410	3-input NAND gates
A4	SN7420	4-input NAND gates
B2	SN7430	8-input NAND gates
B3,B4	MC6880	Bus transceiver (data buffers)
C1,D1	24 pin DIL connector	Processor board connector
C4/3 D4/3	MC6820	PIA (parallel interface)

EXORCISER INTERFACE BOARD

CONNECTORS K1/2

24 pin DIL sockets.

1	- $\emptyset_1$	13	- PB3
2	- PA0	14	- PB4
3	- PA1	15	- PB5
4	- PA2	16	- PB6
5	- PA3	17	- PB7
6	- PA4	18	- CB1
7	- PA5	19	- CB2
8	- PA6	20	- CA1
9	- PA7	21	- CA2
10	- PB0	22	- $\emptyset_2$
11	- PB1	23	- nc
12	- PB2	24	- nc

'nc' = not connected.

EDGE CONNECTOR K3

This edge connector matches the backplane signals of the Motorola EXORciser system (Motorola 1975a, page 7.83).

Appendix 2.

Inter-Processor Messages.

MESSAGE IDENTIFIER (BCD)	valid channel(checked)				MESSAGE MEANING
	HOST IF to CENTRAL PCESSOR	CENTRAL PCESSOR to HOST	CENTRAL PCESSOR to USER	USER IF to CENTRAL PCESSOR	
1		✓		✓	host system command
1	✓		✓		reply to host system command
2		✓			query specification
2				✓	search term
3		✓			postings request
4		✓			pre-search spec
5	✓		✓		term not in thesaurus
6	✓				reply to postings request
6			✓		data received from host concerning a search term
9	✓				no references retrieved by query or pre-search
10	✓				references retrieved by pre-search
11	✓		✓		document reference
12				✓	terminate session
13	✓		✓		broadcast from host

message identifiers $\geq$ 50 = system errors

Appendix 3

The MEDUSA Retrieval System

The document reproduced here describes the features and commands of the MEDUSA retrieval system; it was distributed to users who took part in the experiments carried out with the MEDUSA system (Barber et al 1972 and 1973).

MEDUSA INFORMATION RETRIEVAL SERVICE

USER MANUAL

----------oOo----------

University of Newcastle upon Tyne,
Computing Laboratory.

November, 1975.

MEDUSA INFORMATION RETRIEVAL SYSTEM

INTRODUCTION

MEDUSA is an on-line information retrieval system which allows users unfamiliar with either computers or information science to access the U.S. National Library of Medicine MEDLARS database. By doing this they can keep abreast of developments in their fields, or investigate activity in a possible new region of research.

MEDUSA is run as a British Library project by the Computing Laboratory of the University of Newcastle upon Tyne using their IBM 370/168 computer. This computer is of course based in Newcastle, but it can be accessed from any part of the U.K. by means of the G.P.O. telephone network from a library or research centre possessing a suitable communications terminal. Researchers dialling in to MEDUSA are able to search the Medlars citations for a subset indexed under headings relevant to their interest by conversing with the computer in a simple command language described later in this document. Citations found by this process are printed out on request allowing the user to modify his search strategy to obtain the most relevant set of references. A typical search on-line takes about half an hour, and an example is shown in the appendix.

Two means of accessing the system are provided; Current Awareness and Retrospective Medusa. Both systems allow the user to formulate an identical search, but differ in the manner of searching the data available.

CURRENT AWARENESS MEDUSA is intended for those users who want to keep up to date with the current literature; they will expect to return to the system each month, or at least every three months, and search the data acquired since they last used the system. The database kept for the Current Awareness system is the the latest three months of the file. This is updated as new information arrives, the oldest month being dropped and the new months citations added. There are about 45,000 citations indexed from 2,200 journals for papers written in English, French and German. Users running current awareness searches may retain up to four different profiles from session to session as it is anticipated that they will wish to modify their search criteria as their work progresses. Users are prevented from searching the same month's citations twice with the same search statement.

RETROSPECTIVE MEDUSA is intended for users requiring a simple search on a particular topic from as large a database as possible. Some 110,000 citations are available for searching taken from 1,150 journals over the past year. The citations are restricted to those written in English. A Retrospective session is self-contained; that is, any search formulation is lost when the session ends. A special SAMPLE command, q.v., is supplied to permit checking of a search against the latest block of citations before it is used to access the whole database.

Medusa citations are indexed with terms selected from a thesaurus of 10,000 medical subject headings. The user has to formulate his search using terms from this thesaurus. The main object of the Medusa system is to enable the user to find the correct terms for his subject. The task of finding all relevant terms is made easier by their organization into categories - e.g. neoplasms, musculoskeletal system, vertebrates, surgery. The general term is at the "top" and the more specific terms appear below, down to four levels - e.g. vertebrates - mammals - rodents - mice. The terms above and below a particular term can be displayed easily on the terminal. In fact, going "up", "down" and "across" the category structure is the best way of finding which terms are available - start by typing in a fairly general term, e.g. "respiratory tract diseases", and then move down the category displaying more specific terms. Some of the terms in the dictionary are not used by N.L.M. indexers; their purpose is to clarify the structure of the category, and they can be identified in the print out by the word "non-mesh" and a tally of zero.

Papers are indexed under an average of ten main headings. An important point about the selection of terms is the use of the most specific term for a subject. An indexer would index an article dealing with rats and mice with the terms "rats" and "mice", not the general term "rodents". The only articles to be indexed with "rodents" will be those on rodents in general or on any particular variety of rodent which does not have a main heading to itself in the vocabulary. Because of this indexing practice, the most specific terms must be used for a search. Other points of relevance to searching are discussed later in the SEARCH command description.

When you come to use the service the librarian in charge of your centre will give you a user identifier and its associated password. You will also be shown how to sign on to the system and how to master the mechanical peculiarities of your centre's terminal. When signed on the version of Medusa to be run is selected by typing "c" or "r" in response to the prompt printed out by the system. When you have finished your session merely type SIGNOFF and the system will terminate your run.

MEDUSA COMMANDS

It is essential that this section be read in conjunction with the appendix.

Initial entry to Medusa involves finding entries in the dictionary of medical terms. To do this type in the name of a possible term, and the system will check it against a list of 17,000 entries to 10,000 medical subject headings. If Medusa cannot find it, you will be informed and asked to try a synonym or a broader term, which do. When the system is given a known entry, one of several things may happen.

If the term is a main heading, language or qualifier you will be given a code which is a letter followed by a number. The letter indicates the type of term and is M, L or Q. Future

references to these terms are made using these character groups. A display is given of information about the term - in the case of an M term, the category or categories to which it belongs are printed, together with information on whether more specific terms exist and the tally of the expected number of citations indexed.

As an alternative, sometimes a concept may not be indexed under a single term, in which case either a suggestion of how to retrieve under it is printed, or a search statement will be automatically produced for you, identified by the R code letter.

Once a term has been found which the system recognises, the Medusa dictionary may be explored at will to find related entries. This can be done using the following commands. All should be followed by the term's code group; this should also be followed by the category letter if the term was found in several categories.

DOWN will reveal the more specific terms, if any. If successful, this command will also generate the category term, identified by the C prefix. This refers to all of the terms in the relevant category below the original term.

UP will reveal the broader term.

ACROSS will reveal related terms at the same level.

XREF will reveal any cross references to different categories. These are indicated by an X printed in the display of a term.

Qualifiers are sub-headings which may be linked to main headings and categories to restrict the context in which they retrieve references.

QUAL followed by the code group will print a list of those qualifiers which may be legally linked to a term when forming a search statement.

LIST followed by a character string of three or more letters will cause the system to print out dictionary entry names which start with those letters. These are either main heading names or one of the 6,000 synonyms which are held to assist users. This should help avoid difficulties with misspelling or transatlantic spelling of terms. It is worth remembering that many American spellings contract "ae" and "oe" to a single "e". Some, but not all, of these alternatives are held as synonyms, e.g. FOETUS for FETUS. Note that the list command does not generate terms; if it is desired to search on any of those listed the English name must be typed in.

The next step in the process, having found a set of suitable headings, is to form a search statement, and use it to retrieve citations.

COMBINE followed by a term code, or by a group of codes separated by one of the operators AND, AND NOT, OR, LINK, will form a search statement. The system will print out an R code number and give you a rough estimate of the number of citations liable to be retrieved by its use.

The operators work in the following way:

M11 AND M31 would form a statement which retrieves only those citations indexed under both the terms labelled M11 and M31.

M9 AND NOT M15 retrieves citations indexed under M9 provided that they are not also indexed under M15

M3 OR M12 would retrieve citations indexed by either M3 or M12 or both.

M12-17 is a convenient way of forming:
M12 OR M13 OR M14 OR M15 OR M16 OR M17

M5 LINK Q4 retrieves terms indexed under M5 only if they are restricted to the subheading Q4.

It is wise not to use more than one kind of operator in one search statement. As an R term itself may be used in a COMBINE command, more complex search strategems may be set up by repeated use of the command.

```
COMBINE M1 AND M4     (forming R1)
COMBINE M7 AND M8     (forming R2)
COMBINE R1 OR R2
```

is more obvious in its effect than the single:

```
COMBINE M1 AND M4 OR M7 AND M8
```

SEARCH followed by an R term causes the system to search for citations satisfying the criteria of the term. Citations found are printed out on the terminal in sufficient detail to enable a user to locate them, and with their associated index terms and sub-headings. An asterisk against an index term means that it is a "print term", and appears against the reference to the citation in the indexing journal Index Medicus. If index terms are not desired, as with a profile of established reliability, adding the letter S to the search statement R number will suppress them giving faster printing.

SAMPLE is available only in Retrospective Medusa. It is similar to SEARCH in use but only searches the most recent month of citations.

Certain commands are available to remind you of details of previously used terms you may have forgotten about. This is particularly necessary in Current Awareness Medusa where you may

be coming back to a search profile after some time.

SHORT followed by a term will print a summary of the information about the term.

FULL is like SHORT but also gives the category information as displayed when a term is first found in the dictionary.

DISPLAY followed by an R term will give a complete breakdown of the structure of the search statement, including any R terms included in it.

Five commands are available to allow a user to exercise control over the session.

DESIST causes the system to abbreviate messages, but not term information.

HELP undoes the effect of DESIST should a user get into difficulties and need more explicit prompts from Medusa.

RESTART causes the current search profile to be suspended. In Retrospective Medusa it will be lost, and a new search started. In Current Awareness Medusa, the search is recallable at a later time, and the system will prompt the user for recall of an old search or start of a new one.

SIGNOFF will terminate both the search and the Medusa session. Wait until the terminal has stopped printing before turning it off.

DESTROY is only relevant to Current Awareness Medusa. As there is a limit of four on the number of searches held, it will eventually be necessary for a user to destroy an old one in order to start a new. DESTROY followed by the search number will do this.

One important point to note is the effect of the break or attention key on the terminal.

(a) during DOWN, XREF, QUAL, ACROSS and LIST hitting the break key will curtail printing, but any new terms produced by the command will still be generated.

(b) during execution of a SEARCH or SAMPLE command, searching will stop at the end of the current reference being printed.

(c) induce the keyboard to unlock at the start of a session, should any trouble occur.

(d) hitting the break key several times in succession can have a nasty effect on the system. Please wait for 10 seconds before hitting the key a second time, should the first have no effect.

Medusa Service User Manual

ADDITIONAL INFORMATION

The following procedure is necessary to start a Medusa session.

(1) Turn on the power to the terminal.

(2) Dial in to the computer on 0632-611301. A whistling tone should be heard when the computer answers. If the ringing tone persists the computer is not operational; confirm this by dialling Newcastle University Computer room. An engaged tone means that all the lines into the computer are tied up; try again later.

(3) Place receiver in acoustic coupler, or if you have a data telephone switch to DATA. Ensure that your terminal and coupler are both set to full duplex mode of operation.

(4) The computer should respond with:

PLEASE ENTER TERMINAL TYPE ..

Reply to this as instructed by your librarian. If this prompt is not printed, try hitting the break key once.

(5) When the system prints out a line starting MTS and then unlocks type:

SIGNON user-identifier PW=password

Entry to Medusa is then automatic.

News of changes in the Medusa service, either alterations in hours or in the data available are broadcast to the user as Medusa starts up. Should you sign on during normal Medusa hours and find that the system is not available, please inform Medusa staff who will endeavour to fix the trouble.

Occasionally, we hope very rarely, Medusa itself may develop a fault. It is, however, clever enough to diagnose itself and will print out a message starting:

PLEASE REPORT SYSTEM FAILURE ....

Please telephone Medusa staff if this happens.

Sometimes the computer itself will fail ("go down") during a session. Usually it will be back on the air within ten minutes but a telephone call will confirm the fault. Try redialling the computer first, as the fault may be with the telephone line only.

Medusa staff can be reached on 0632-29233 ext 270.

The computer room number is 0632-21586.

APPENDIX

AN EXAMPLE OF A MEDUSA RUN AT A TELETYPE

In the run of Retrospective Medusa which follows, all comments, i.e. lines not printed by the terminal, are shown in lower case. Otherwise this run is typical of a Medusa session.

```
PLEASE ENTER TERMINAL TYPE ..   WEST770
MTS(PDA1TT06-0031)
#SIGNON CLA1 PW=MEDUSER
# CHARGING RATE=UNIVERSITY,TERMINAL
#**LAST SIGNON WAS: 18:46.47   06-14-74
# USER "CLA1" SIGNED ON AT  14:08.52 ON 06-17-74


    MEDUSA INFORMATION RETRIEVAL SERVICE

 PLEASE INDICATE WHETHER YOU WISH TO USE CURRENT AWARENESS
 OR RETROSPECTIVE MEDUSA BY TYPING "C" OR "R".
?R


 RETROSPECTIVE MEDUSA


 THE RETROSPECTIVE SERVICE IS AVAILABLE FROM 12.00 - 14.00 AND
 FROM 16.00 - 19.00 EACH WEEKDAY. SHOULD YOU SIGN ON DURING THESE
 PERIODS AND FIND THE SYSTEM NOT AVAILABLE, PLEASE RING MEDUSA
 STAFF ON 0632-28511 EXT. 3828.
 SYSTEM HOLDS CITATIONS FOR APRIL AND MAY 1974, OCTOBER 1973 TO MARCH
 1974, APRIL TO SEPTEMBER 1973


*  SIGN INDICATES THAT SYSTEM IS READY FOR A REPLY
 DO YOU NEED HELP? Y/N.
*Y

 SEARCH NUMBER   1 : USERCODE, NAME, TITLE ?
*0109, J.A.HUNTER, DEMONSTRATION SEARCH
 ENTER TERMS. START BY TYPING IN A MEDICAL TERM RELATED TO AN ASPECT
 OF YOUR SEARCH
*STUDENTS
 M1=STUDENTS                    T285      1962              X
          DN 6          A EDUCATION                  (ANTHROPOLOGY,
 EDUCATION,)

          DN 0          B EDUCATION, NONPROFESSION(ANTHROPOLOGY,
 EDUCATION,)

          DN 1.5        C NAMED GROUPS (NON MESH)


          DN 0          D OCCUPATIONS                (SOCIOECONOMIC
 FACTORS    (POPULATION CHARACTERISTI))
```

The term "students" has been assigned the code M1. T285 gives the number of citations indexed under this heading, and 1962

gives the date of introduction of the term. X indicates the presence of one or more cross references. This term is in four categories A,B,C and D. In the C category, the code DN 1.5 means that there is one term in the next lower level of the category, and five in the level below that. In the A category, "education" is the broader term, and the information which follows it gives the category structure above it.

```
*DOWN M1A
 C1=STUDENTS                        TT858
 M2=STUDENTS, DENTAL                T93        1962                A,B,C
 M3=STUDENTS, HEALTH OCCUPAT        T21             NEW TERM       A,B,C
 M4=STUDENTS, MEDICAL               T267       1962                A,B,C
 M5=STUDENTS, NURSING               T174       1962                A,B,C
 M6=STUDENTS, PHARMACY              T9              NEW TERM       A,B,C
 M7=STUDENTS, PREMEDICAL            T9         1962                A,B,C
```

Here "down" has generated for the A category the category term C1, which encompasses all terms below and including "students" in that category. TT858 gives the total tally - the number of times the terms in the group have been used in indexing references.

```
*FULL M4
 M4=STUDENTS, MEDICAL               T267       1962
          DN 0          A STUDENTS                        (EDUCATION
      (ANTHROPOLOGY, EDUCATION,))

          DN 0          B STUDENTS, HEALTH OCCUPAT(STUDENTS
      (NAMED GROUPS (NON MESH) ))

          DN 0          C STUDENTS, HEALTH OCCUPAT(HEALTH MANPOWER
      (FACILITIES MANPOWER SERV))

*XREF M1
 M8=STUDENT DROPOUTS                T48        1962              X A,B,C
*STUDENT HEALTH SERVICES
 M9=STUDENT HEALTH SERVICES         T114       1962
          DN 0          A HEALTH SERVICES                 (FACILITIES
 MANPOWER SERV)

*COLLEGES
 M10=UNIVERSITIES                   T435       1962
          DN 0          A SCHOOLS                         (EDUCATION
      (ANTHROPOLOGY, EDUCATION,))
```

Note the action here in the case of a synonym being entered. "colleges" is held in the dictionary as a pointer to "universities", for which it generates a code.

```
*COMBINE M1 OR M8 OR M9 OR M10

 R1= M1 OR M8 OR M9 OR M10
 EXPECTED RETURN : LARGE
```

Here large means 25 or more citations would be retrieved.

```
*SAMPLE R1

 FIRST CITATION FOUND IN  21 SECS
```

```
 CIT NUM 00290919
 HOWELL R   CROWN S    HOWELL RW
 PERSONALITY AND PSYCHOSOCIAL INTERACTIONS IN AN UNDERGRADUATE SAMPLE.
 BR J PSYCHIATRY VOL123  699-701  DEC 73
 *PERSONALITY ASSESSMENT                 *STUDENTS
  ADOLESCENCE                             ADULT
  AFFECTIVE DISTURBANCES(DIAGNOSIS)       FACULTY
  FEMALE                                  HUMAN
  JUVENILE DELINQUENCY                    MALE
  PSYCHOMETRICS                           SCHOOLS
  SEX FACTORS                             SOCIAL CLASS
```

In this print out of a reference, the terms marked with an asterisk are "print terms" which would appear in Index Medicus. Terms in brackets are qualifiers, e.g. (diagnosis).

```
 CIT NUM 00290921
 DUDDLE M
 AN INCREASE OF ANOREXIA NERVOSA IN A UNIVERSITY POPULATION.
 BR J PSYCHIATRY VOL123  711-2  DEC 73
 *ANOREXIA NERVOSA(OCCURRENCE)           *STUDENTS
  ADOLESCENCE                             ADULT
  EDUCATIONAL STATUS                      ENGLAND
  FEMALE                                  HUMAN
  INFANT NUTRITION                        MALE
  OBESITY                                 UNIVERSITIES

 CIT NUM 00291303
 CAMPBELL LP
 MODIFYING ATTITUDES OF UPPER ELEMENTARY STUDENTS TOWARD SMOKING.
 J SCH HEALTH VOL44  97-8  FEB 74
 *HEALTH EDUCATION                       *SMOKING(PREV  CONTRL)
  ADOLESCENCE                             ATTITUDE TO !!

 ATTENTION INTERRUPT
  MALE                                    STUDENTS
```

Here the break key has been used to stop a search. Note that printing of the current citation is completed before control is returned to the user.

```
*DRUG ADDICTION
 M11=DRUG ADDICTION              T816      1962              X
          DN 3          A DRUG ABUSE                (PSYCHIATRY
     )

          DN 0          B SOCIOPATHIC PERSONALITY (PERSONALITY
 DISORDERS   (PSYCHIATRY              ))

*ACROSS M11A
 M12=GLUE SNIFFING               T18             NEW TERM    A,B,C
 M11=DRUG ADDICTION              T816      1962            X A,B
*UP M11A
 M13=DRUG ABUSE                  T708      1962            X A,B,C
*COMBINE R1 AND M13

 R2= R1 AND M13
 EXPECTED RETURN : SMALL
```

Here small means 10 or less retrievals.

```
*SAMPLE R2

 FIRST CITATION FOUND IN  18 SECS

 CIT NUM 00304555
 BIENER K
 <DIFFERENT PROBLEMS OF DRUGS IN TRADE SCHOOL AND HIGH SCHOOL
 STUDENTS (AUTHOR'S TRANSL)>
 PRAXIS VOL62  1612-5  26 DEC 73
 *DRUG ABUSE(OCCURRENCE)              *STUDENTS
  ADOLESCENCE                          AGE FACTORS
  CANNABIS                             COCAINE
  COMPARATIVE STUDY                    ECONOMICS
  EXPLORATORY BEHAVIOR                 FEMALE
  HUMAN                                LYSERGIC ACID DIETHYLAMI
  MALE                                 MESCALINE
  SOCIOECONOMIC FACTORS                STATISTICS
  SWITZERLAND
*QUAL M13
 Q1=BLOOD
 Q2=CEREBR.FLUID
 Q3=CHEM.INDUCED
 Q4=CLASSIFICAT.
 Q5=COMPLICATIONS
 Q6=DIAGNOSIS
 Q7=DRUG THERAPY
 Q8=EDUCATICN
 Q9=ENZYMOLOGY
 Q10=ETIOLOGY
 Q11=FAMIL&GENET.
 Q12=HISTORY
 Q13=IMMUNOLOGY
 Q14=INSTRUMENTATION
 Q15=MANPOWER
 Q16=METABOLISM
 Q17=MORTALITY
 Q18=NURSING
 Q19=OCCURRENCE
 Q20=PATHOLOGY
 Q21=PHYSIOPATH.
 Q22=PREV.&CONTRL
 Q23=RADIOGRAPHY
 Q24=RADIOTHRPY
 Q25=REHABILITAT.
 Q26=STANDARDS
 Q27=SURGERY
 Q28=THERAPY
 Q29=URINE
```

The following three combine commands show how to build up a search statement which will find references if they are indexed under "drug abuse" linked to either of two qualifiers. Note that a match will not occur unless one of these qualifiers is actually specified for the main heading "drug abuse", and that terms may appear several times in one citation linked to different qualifiers.

```
*COMBINE M13 LINK Q19
```

```
 R3= M13 LINK Q19
 EXPECTED RETURN : SMALL

*COMBINE M13 LINK Q22

 R4= M13 LINK Q22
 EXPECTED RETURN : SMALL

*COMBINE R3 OR R4

 R5= R3 OR R4
 EXPECTED RETURN : SMALL

*SEARCH R5

 APRIL AND MAY 1974 CITATIONS

 FIRST CITATION FOUND IN  14 SECS

 CIT NUM 00297727
 LOWINGER P
 HOW THE PEOPLE'S REPUBLIC OF CHINA SOLVED THE DRUG ABUSE PROBLEM.
 AM J CHIN MED VOL1  275-82  JUL 73
 *DRUG ADDICTION(PREV  CONTRL)        ATTITUDE TO HEALTH
  CHINA                               DRUG ABUSE(PREV  CONTRL)
  DRUG ADDICTION(DRUG THERAPY)        DRUG ADDICTION(REHABILITAT )
  DRUG AND NARCOTIC CONTRO            HISTORICAL ARTICLE
  HISTORY OF MEDICINE, 19T            HISTORY OF MEDICINE, 20T
  HISTORY OF MEDICINE, MED            HONG KONG
  HUMAN                               MORALS
  OPIUM(HISTORY)

 CIT NUM 00305850
 BRIGGS AH
 CAN WE PREVENT DRUG ABUSE IN INDUSTRY?
 TEX MED VOL70  49-54  JAN 74
 *DRUG ADDICTION(PREV  CONTRL)       *INDUSTRIAL MEDICINE
  DRUG ABUSE(PREV  CONTRL)            HEALTH EDUCATION
  HUMAN                               UNITED STATES

 CIT NUM 00287976
 EINSTEIN S
 DRUG ABUSE TRAINING AND EDUCATION: THE COMMUNITY ROLE.
 AM J PUBLIC HEALTH VOL64  99-106  FEB 74
 *DRUG ABUSE                         *HEALTH EDUCATION
  ATTITUDE OF HEALTH PERSO            CRIME
  CURRICULUM                          DECISION MAKING
  DRUG ABUSE(PREV  CONTRL)            DRUGS
  HUMAN                               JURISPRUDENCE
  METHODS                             RELIGION
  SCIENCE                             SOCIAL VALUES
  UNITED STATES
```

```
CIT NUM 00234363
DISTASIO C NAWROT M
METHAQUALONE.
AM J NURS VOL73  1922-5  NOV 73
*DRUG ABUSE(PREV  CONTRL)              *METHAQUALONE
 ADULT                                  DRUG AND NARCOTIC CONTRO
 DRUG WITHDRAWAL SYMPTOMS(DRUG THE)     HUMAN
 JAPAN                                  PENTOBARBITAL(THERAP  USE)
 UNITED STATES


OCTOBER 1973 TO MARCH 1974 CITATIONS

CIT NUM 00244640
GREENE MH  DUPONT RL  RUBENSTEIN RM
AMPHETAMINES IN THE DISTRICT OF COLUMBIA. II. PATTERNS OF ABUSE IN
AN ARRESTEE POPULATION.
ARCH GEN PSYCHIATRY VOL29  773-6  DEC 73
*AMPHETAMINE                           *CRIMINAL PSYCHOLOGY
*DRUG ABUSE(OCCURRENCE)                *SOCIAL CONTROL, FORMAL
 ADULT                                  DISTRICT OF COLUMBIA
 DRUG ABUSE(PREV  CONTRL)               DRUG ADDICTION(OCCURRENCE)
 FEMALE                                 HEROIN ADDICTION(OCCURRENCE)
 HUMAN                                  MALE
 METHADONE                              VIOLENCE

CIT NUM 00260832
REDFIELD JT
DRUGS IN THE WORKPLACE--SUBSTITUTING SENSE FOR SENSATIONALISM.
AM J PUBLIC HEALTH VOL63  1064-70  DEC 73
*DRUG ABUSE                            *INDUSTRIAL MEDICINE
 ADOLESCENCE                            ADULT!!

ATTENTION INTERRUPT
 CANNABIS                               DRUG ABUSE(DIAGNOSIS)
 DRUG ABUSE(OCCURRENCE)                 DRUG ABUSE(PREV  CONTRL)
 DRUG ABUSE(URINE)                      DRUG ADDICTION
 HALLUCINOGENS                          HEALTH EDUCATION
 HUMAN                                  HYPNOTICS AND SEDATIVES
 LYSERGIC ACID DIETHYLAMI               OCCUPATIONAL HEALTH SERV
 OPIUM                                  OREGON
 SMOKING(OCCURRENCE)                    STUDENTS
*SIGNOFF

 END SEARCH NUMBER 1
 ELAPSED TIME WAS  38 MINS  23 SECS
 THE COSTS FOLLOWING ARE IN POUNDS STERLING
#OFF AT 14:48.22    06-17-74
#ELAPSED TIME       39.5   MIN.        $.38
#CPU TIME USED      37.454 SEC.        $.62
#CPU STOR VMI       28.5   PAGE-MIN.   $.23
#WAIT STOR VMI      30.023 PAGE-HR.    $2.92
#DRUM READS        6395
#MAX VM PAGES       56
#APPROX. COST OF THIS RUN IS    $4.15
#DISK STORAGE      413     PAGE-HR.    $.03
#APPROX. REMAINING BALANCE:  $174.15
```

MEDUSA

—:o:—

ENTRY

TELE-TYPE-LINK — Switch on teletype and acoustic coupler, dial Newcastle 610822 or 610823, wait for high-pitched whistle (see Note No. 1), place telephone receiver in handset on the acoustic coupler.

MTS-ENTRY (See note 2) — The computer will request: PLEASE ENTER TERMINAL TYPE... Reply with the code supplied by your library for your type of terminal and press the RETURN key.

Wait for the computer to prompt you with a ♯ sign and type: SIGNON id PW=password where id is the 4-digit user identifying No. and password is the password for your centre.

MEDUSA PROGRAM

(See notes Nos. 3, 4)

The computer will ask you next which service you want to use. Reply C for current-awareness Medusa, and R for retrospective Medusa, and the appropriate program will be loaded.

LETTERS USED AS CODES

M	Mesh term (i.e. medical subject heading)
R	Search statement
Q	Qualifier (i.e. subheading) term
L	Language term
C	Category term, covering a collection of Mesh terms more specific than named term

---

COMMANDS WITH CODES

(See note No. 5)

Formed by typing the name of the command followed by a space and the code No., e.g., DOWN M2A.

UP	Shows more general term
DOWN	Shows more specific terms; creates category term
ACROSS	Shows related terms in same class
FULL	Gives all information on term
DISPLAY	Gives all information on search statement
SHORT	Displays name of term
XREF	Shows related terms in different classes
QUAL	Shows permitted qualifiers for a term; produces Q's.

LIST — Displays terms in dictionary starting with given letters, e.g. LIST POL displays terms beginning with POL

COMBINE — Followed by a logical search statement produces R e.g. COMBINE R1 AND M17. Terms may be connected by 'AND', 'OR', 'LINK', or 'AND NOT' logic. For examples, see appendix to the search manual.

SEARCH — Searches citation files using designated search statement, e.g. SEARCH R1. All index terms in retrieved citations are printed, unless the command is terminated by S, e.g. SEARCH R3S.

SAMPLE (Retro only) — Similar to SEARCH, except that only the most recent month in the data base is searched.

DESTROY (Curr. Awar. only) — Destroys complete searches when no longer required, e.g. DESTROY 1, 2 destroys both searches No. 1 and No. 2.

COMMANDS WITHOUT CODES

HELP	Shows fuller messages.
DESIST	Suppresses fuller messages.
RESTART	Finishes with current search No., enabling a different search No. to be selected.

DISPLAYS

DN	No. of terms below in hierarchy.
X	Cross-references present.
T	Average No. of times term used over the period the data base covers.
A, B, C, D	Letters assigned to each class in which term appears in tree-structure.
NON-MESH	Not a Mesh term.
EXPECTED-RETURN	Predicted order of output size from using the statement to search the data available.

ATTENTION INTERRUPT

Hitting the BREAK key once generates an attention interrupt which will:

(a) curtail printing in 'DOWN', 'XREF', 'QUAL', 'ACROSS', 'LIST', 'DISPLAY'.

(b) stop 'SEARCH' or 'SAMPLE' at the end of the reference being printed.

If the system does not apparently respond to the BREAK, wait at least 10 sec. before trying again to prevent a possible system failure.

EXIT

Type:
SIGNOFF
Wait until printing stops.

TELE-TYPE — Replace receiver on telephone. Switch off teletype and acoustic coupler.

NOTES

1. Engaged tone means that the line dialled is unavailable. A prolonged ring with no answer probably means that the computer is down; this may be confirmed by dialling the computer room.

2. Entry: Check that terminal is not in local and that both it and the acoustic coupler are set on full-duplex. Failure to respond probably means the computer has gone down.

3. If a system failure occurs, the Medusa program terminates. Please inform the Computing Laboratory Medusa staff or your library.

4. If the system is reported unavailable during usual Medusa times, inform Computing Laboratory Medusa staff.

5. If an inappropriate error message is printed, the line entered may have been scrambled, particularly by a telephone link.

Enter the line again.

USEFUL TELEPHONE NUMBERS

Computer Link—Newcastle 610822, 610823

Medusa staff—Newcastle 28511 ext. 2761

Computer room—Newcastle 21586

STD Code for Newcastle upon Tyne is 0632

UNIVERSITY OF
NEWCASTLE UPON TYNE
COMPUTING LABORATORY

ON-LINE MEDUSA INSTRUCTION LEAFLET

JUNE 1974

Appendix 4.

Pre-Searching Results with Test Collections

The following tables present results of experiments carried out with three test collections (CRANFIELD 200, KEEN and CRANFIELD 1400) to examine the feasibility of the method for retrieving document references in ranked order and the effect of pre-searching with various threshold weights (see chapter 6). Conclusions which can be drawn from the data are presented in chapter 6.

Programs were written in the BCPL programming language and run on an ICL 1904S computer.

Key:

S = number of true searches.

P = number of pre-searches.

T = total (S+P).

(Some totals may be in error by a figure of one or two due to the use of integer arithmetic within the BCPL programs and the consequent effect of truncation).

Figures immediately below the query size in each table indicate the number of queries of that size in the particular collection.

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 0.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	1	2	1	1	1	1	1	1
1 P	-	0	5	8	14	12	20	19	37	24
T	-	1	6	10	15	14	21	20	38	25
S	-	5	6	10	15	9	86	16	21	110
5 P	-	0	5	9	19	25	49	45	65	60
T	-	5	11	19	34	34	135	61	86	170
S	-	5	7	14	21	21	102	27	83	158
10 P	-	0	5	9	19	28	49	48	97	77
T	-	5	12	23	40	49	151	75	180	235
S	-	5	9	17	27	27	112	38	134	199
15 P	-	0	5	9	19	28	50	50	114	82
T	-	5	14	26	46	55	162	88	248	281
S	-	5	9	18	31	33	122	43	169	232
20 P	-	0	5	9	19	28	50	51	119	88
T	-	5	14	27	50	61	172	94	288	320
S	-	5	10	20	33	38	124	46	197	241
25 P	-	0	5	9	19	28	50	51	125	89
T	-	5	15	29	52	66	174	97	322	330
S	-	5	10	21	35	40	128	48	217	274
30 P	-	0	5	9	19	28	50	51	126	90
T	-	5	15	30	55	68	178	99	343	364
S	-	7	12	24	42	56	142	67	284	343
ALL P	-	0	5	9	19	28	50	52	127	91
ALL T	-	7	17	34	61	84	192	119	411	434
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 100.

	QUERY SIZE (and population)									
No DOCS	2 0	3 1	4 4	5 6	6 10	7 8	8 3	9 3	10 5	11 2
1 S	-	1	1	2	1	1	1	1	1	1
1 P	-	0	4	8	14	12	20	19	37	24
1 T	-	1	5	9	15	13	21	20	38	25
5 S	-	5	6	10	15	9	86	16	21	110
5 P	-	0	4	9	19	25	49	45	65	60
5 T	-	5	10	19	34	34	135	61	86	170
10 S	-	5	7	14	21	21	102	27	83	158
10 P	-	0	4	9	19	28	49	48	97	77
10 T	-	5	11	23	40	49	151	75	180	235
15 S	-	5	9	17	27	27	112	38	134	199
15 P	-	0	4	9	19	28	50	50	114	82
15 T	-	5	13	26	46	55	162	88	248	281
20 S	-	5	9	18	31	33	122	43	169	232
20 P	-	0	4	9	19	28	50	51	119	88
20 T	-	5	13	27	50	61	172	94	288	320
25 S	-	5	10	20	33	38	124	46	197	241
25 P	-	0	4	9	19	28	50	51	125	89
25 T	-	5	14	29	52	66	174	97	322	330
30 S	-	5	10	21	35	40	128	48	217	274
30 P	-	0	4	9	19	28	50	51	126	90
30 T	-	5	14	30	54	68	178	99	343	364
ALL S	-	7	12	24	42	56	142	67	284	343
ALL P	-	0	4	9	19	28	50	52	127	91
ALL T	-	7	16	33	61	84	192	119	411	434
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 200.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	1	2	1	1	1	1	1	1
1 P	-	0	2	4	12	12	20	19	36	24
T	-	1	3	6	13	13	21	20	37	25
S	-	5	6	10	15	9	86	16	21	110
5 P	-	0	2	5	16	24	48	43	64	59
T	-	5	8	15	31	33	134	59	85	169
S	-	5	7	14	21	21	102	27	83	158
10 P	-	0	2	5	16	26	49	46	96	76
T	-	5	9	19	37	47	151	73	179	234
S	-	5	9	17	27	27	112	38	134	199
15 P	-	0	2	5	16	26	50	49	113	81
T	-	5	11	22	43	53	162	87	247	280
S	-	5	9	18	31	33	122	43	169	232
20 P	-	0	2	5	16	26	50	49	118	87
T	-	5	11	23	47	59	172	92	287	319
S	-	5	10	20	33	38	124	46	197	241
25 P	-	0	2	5	16	26	50	49	124	89
T	-	5	12	25	49	64	174	95	321	330
S	-	5	10	21	35	40	128	48	217	274
30 P	-	0	2	5	16	26	50	49	125	90
T	-	5	12	26	51	66	178	97	342	364
S	-	7	12	24	42	56	142	68	285	346
ALL P	-	0	2	5	16	26	50	49	126	91
ALL T	-	7	14	29	58	82	192	117	411	437
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 300.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	1	4	1	1	1	1	1	1
1 P	-	0	0	0	6	9	19	19	35	27
T	-	1	1	4	7	10	20	20	36	28
S	-	5	6	13	17	12	87	29	21	110
5 P	-	0	0	0	8	15	45	43	65	67
T	-	5	6	13	25	27	132	72	86	177
S	-	5	7	16	24	24	104	42	83	158
10 P	-	0	0	0	8	15	45	45	101	96
T	-	5	7	16	32	39	149	87	184	254
S	-	5	9	20	30	31	115	63	134	223
15 P	-	0	0	0	8	15	45	45	121	103
T	-	5	9	20	38	46	160	108	255	326
S	-	5	10	21	34	47	125	69	176	272
20 P	-	0	0	0	8	15	45	45	125	103
T	-	5	10	21	42	52	170	114	301	375
S	-	5	10	23	36	42	128	72	212	286
25 P	-	0	0	0	8	15	45	45	128	103
T	-	5	10	23	44	57	173	117	339	389
S	-	5	10	24	38	44	131	74	235	321
30 P	-	0	0	0	8	15	45	45	128	103
T	-	5	10	24	46	59	176	119	363	424
S	-	7	13	27	46	59	145	94	308	395
ALL P	-	0	0	0	8	15	45	45	128	103
ALL T	-	7	13	27	54	74	190	139	436	498
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 400.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	3	6	2	8	1	15	1	1
1 P	-	0	0	0	2	10	19	27	48	60
T	-	1	3	6	4	18	20	42	49	61
S	-	5	8	16	23	30	107	81	26	198
5 P	-	0	0	0	4	13	43	49	95	132
T	-	5	8	16	27	43	150	130	121	330
S	-	5	9	20	30	45	125	107	141	352
10 P	-	0	0	0	4	13	43	49	126	132
T	-	5	9	20	34	58	168	156	267	484
S	-	5	11	24	36	52	136	128	239	429
15 P	-	0	0	0	4	13	43	49	130	132
T	-	5	11	24	40	65	179	177	369	561
S	-	5	12	25	40	57	147	134	295	478
20 P	-	0	0	0	4	13	43	49	130	132
T	-	5	12	25	44	70	190	183	425	610
S	-	5	12	26	42	62	149	137	336	492
25 P	-	0	0	0	4	13	43	49	130	132
T	-	5	12	26	46	75	192	186	466	624
S	-	5	12	27	44	64	154	139	359	527
30 P	-	0	0	0	4	13	43	49	130	132
T	-	5	12	27	48	77	196	188	489	659
S	-	7	15	31	51	80	167	159	432	601
ALL P	-	0	0	0	4	13	43	49	130	132
ALL T	-	7	15	31	55	93	210	208	562	733
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 500.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	3	6	8	29	4	57	19	124
1 P	-	0	0	0	0	4	17	27	88	91
T	-	1	3	6	8	33	21	84	107	215
S	-	5	8	16	30	57	143	165	140	543
5 P	-	0	0	0	1	5	29	46	113	143
T	-	5	8	16	31	62	172	211	253	686
S	-	5	9	20	37	72	161	191	319	696
10 P	-	0	0	0	1	5	29	46	127	143
T	-	5	9	20	38	77	190	237	446	839
S	-	5	11	24	43	78	172	212	427	773
15 P	-	0	0	0	1	5	29	46	127	143
T	-	5	11	24	44	83	201	258	554	916
S	-	5	12	25	47	84	183	218	483	823
20 P	-	0	0	0	1	5	29	46	127	143
T	-	5	12	25	48	89	212	264	610	966
S	-	5	12	26	49	89	185	221	524	836
25 P	-	0	0	0	1	5	29	46	127	143
T	-	5	12	26	50	94	214	267	651	979
S	-	5	12	27	51	91	189	223	547	871
30 P	-	0	0	·0	1	5	29	46	127	143
T	-	5	12	27	52	96	218	269	674	1014
S	-	7	15	31	58	107	203	243	620	945
ALL P	-	0	0	0	1	5	29	46	127	143
ALL T	-	7	15	31	59	112	232	289	747	1088
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 600.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	3	6	11	42	16	95	128	343
1 P	-	0	0	0	0	0	3	29	66	83
T	-	1	3	6	11	42	19	124	194	427
S	-	5	8	16	34	73	178	247	293	941
5 P	-	0	0	0	0	0	7	39	81	124
T	-	5	8	16	34	73	185	286	374	1065
S	-	5	9	20	41	87	197	273	494	1095
10 P	-	0	0	0	0	0	7	39	90	124
T	-	5	9	20	41	87	204	312	584	1219
S	-	5	11	24	47	94	208	295	602	1172
15 P	-	0	0	0	0	0	7	39	90	124
T	-	5	11	24	47	94	215	334	692	1296
S	-	5	12	25	51	99	218	300	658	1221
20 P	-	0	0	0	0	0	7	39	90	124
T	-	5	12	25	51	99	225	339	748	1345
S	-	5	12	26	53	105	221	303	699	1235
25 P	-	0	0	0	0	0	7	39	90	124
T	-	5	12	26	53	105	228	342	789	1359
S	-	5	12	27	55	107	224	306	722	1270
30 P	-	0	0	0	0	0	7	39	90	124
T	-	5	12	27	55	107	231	345	812	1394
S	-	7	15	31	63	127	238	325	795	1344
ALL P	-	0	0	0	0	0	7	39	90	124
ALL T	-	7	15	31	63	127	245	364	885	1468
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : 700.

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	3	6	11	47	19	136	210	520
1 P	-	0	0	0	0	0	0	30	45	72
T	-	1	3	6	11	47	19	166	255	592
S	-	5	8	16	34	77	194	312	387	1280
5 P	-	0	0	0	0	0	0	30	56	82
T	-	5	8	16	34	77	194	342	443	1362
S	-	5	9	20	41	92	213	338	609	1434
10 P	-	0	0	0	0	0	0	30	56	82
T	-	5	9	20	41	92	213	368	665	1516
S	-	5	11	24	47	98	224	359	717	1511
15 P	-	0	0	0	0	0	0	30	56	82
T	-	5	11	24	47	98	224	389	773	1593
S	-	5	12	25	51	104	235	365	773	1560
20 P	-	0	0	0	0	0	0	30	56	82
T	-	5	12	25	51	104	235	395	829	1642
S	-	5	12	26	53	109	237	368	814	1574
25 P	-	0	0	0	0	0	0	30	56	82
T	-	5	12	26	53	109	237	398	870	1656
S	-	5	12	27	55	111	240	370	837	1609
30 P	-	0	0	0	0	0	0	30	56	82
T	-	5	12	27	55	111	240	400	893	1691
S	-	7	15	31	63	127	255	390	910	1683
ALL P	-	0	0	0	0	0	0	30	56	82
T	-	7	15	31	63	127	255	420	966	1765
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: CRANFIELD 200. PRE-SEARCH THRESHOLD WEIGHT : α

	QUERY SIZE (and population)									
No	2	3	4	5	6	7	8	9	10	11
DOCS	0	1	4	6	10	8	3	3	5	2
S	-	1	3	6	11	47	19	257	298	854
1 P	-	0	0	0	0	0	0	0	0	0
T	-	1	3	6	11	47	19	257	298	854
S	-	5	8	16	34	77	194	433	499	1644
5 P	-	0	0	0	0	0	0	0	0	0
T	-	5	8	16	34	77	194	433	499	1644
S	-	5	9	20	41	92	213	459	721	1798
10 P	-	0	0	0	0	0	0	0	0	0
T	-	5	9	20	41	92	213	459	721	1798
S	-	5	11	24	47	98	224	480	829	1875
15 P	-	0	0	0	0	0	0	0	0	0
T	-	5	11	24	47	98	224	480	829	1875
S	-	5	12	25	51	104	235	486	885	1924
20 P	-	0	0	0	0	0	0	0	0	0
T	-	5	12	25	51	104	235	486	885	1924
S	-	5	12	26	53	109	237	489	926	1938
25 P	-	0	0	0	0	0	0	0	0	0
T	-	5	12	26	53	109	237	489	926	1938
S	-	5	12	27	55	111	240	491	949	1973
30 P	-	0	0	0	0	0	0	0	0	0
T	-	5	12	27	55	111	240	491	949	1973
S	-	7	15	31	63	127	255	511	1023	2047
ALL P	-	0	0	0	0	0	0	0	0	0
T	-	7	15	31	63	127	255	511	1023	2047
TTL DOCS	-	156	104	84	103	133	115	143	127	176

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 0.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	-	2
S	1	2	1	1	1	1	1	1	1	1	-	1
1 P	0	0	4	7	8	12	17	19	33	42	-	49
T	1	2	5	8	9	13	18	20	34	43	-	50
S	1	4	6	7	10	6	8	20	8	28	-	34
5 P	0	0	5	8	15	25	29	63	50	78	-	78
T	1	4	11	15	25	31	37	83	58	106	-	146
S	2	5	6	10	13	11	13	40	21	36	-	92
10 P	0	0	5	9	17	28	32	84	60	84	-	147
T	2	5	11	19	30	39	45	124	81	120	-	239
S	2	6	7	12	15	20	17	50	32	41	-	123
15 P	0	0	5	9	17	29	32	90	62	89	-	158
T	2	6	12	21	32	49	49	140	94	130	-	281
S	2	6	8	12	16	22	18	59	37	50	-	131
20 P	0	0	5	9	17	30	32	92	62	91	-	160
T	2	6	13	21	33	52	50	151	99	141	-	291
S	2	6	8	13	17	25	20	61	45	53	-	154
25 P	0	0	5	9	17	30	32	93	63	91	-	162
T	2	6	13	22	34	55	52	154	107	144	-	316
S	2	6	8	13	18	26	20	72	45	57	-	162
30 P	0	0	5	9	17	30	32	96	63	92	-	164
T	2	6	13	22	35	56	52	168	107	149	-	326
S	3	7	10	15	23	36	30	95	56	79	-	212
ALL P	0	0	5	9	17	30	33	96	63	93	-	166
ALL T	3	7	15	24	40	66	63	191	119	172	-	378
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 100.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	-	2
S	1	2	1	1	1	1	1	1	1	1	-	1
1 P	0	0	4	7	8	12	17	19	33	42	-	49
T	1	2	5	8	9	13	18	20	34	43	-	50
S	1	4	6	7	10	6	8	20	8	28	-	34
5 P	0	0	5	8	15	25	29	63	50	78	-	112
T	1	4	11	15	25	31	37	83	58	106	-	146
S	2	5	6	10	13	11	13	40	21	36	-	92
10 P	0	0	5	9	17	28	32	84	60	84	-	147
T	2	5	11	19	30	39	45	124	81	120	-	239
S	2	6	7	12	15	20	17	50	32	41	-	123
15 P	0	0	5	9	17	29	32	90	62	89	-	158
T	2	6	12	21	32	49	49	140	94	130	-	281
S	2	6	8	12	16	22	18	59	37	50	-	131
20 P	0	0	5	9	17	30	32	92	62	91	-	160
T	2	6	13	21	33	52	50	151	99	141	-	291
S	2	6	8	13	17	25	20	61	45	53	-	154
25 P	0	0	5	9	17	30	32	93	63	91	-	163
T	2	6	13	22	34	55	52	154	107	144	-	317
S	2	6	8	13	18	26	20	72	45	57	-	162
30 P	0	0	5	9	17	30	32	96	63	92	-	164
T	2	6	13	22	35	56	52	168	107	149	-	326
S	3	7	10	15	23	36	30	95	56	79	-	212
ALL P	0	0	5	9	17	30	33	96	63	93	-	167
ALL T	3	7	15	24	40	66	63	191	119	172	-	379
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 200.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
S	1	2	1	1	1	1	1	1	1	1	-	1
1 P	0	0	3	7	8	12	17	19	33	42	-	49
T	1	2	4	8	9	13	18	20	34	43	-	50
S	1	4	6	7	10	6	8	20	8	28	-	34
5 P	0	0	4	8	15	25	29	63	50	78	-	112
T	1	4	10	15	25	31	37	83	58	106	-	146
S	2	5	6	10	13	11	13	40	21	36	-	92
10 P	0	0	4	8	17	28	32	84	60	84	-	147
T	2	5	10	18	30	39	45	124	81	120	-	239
S	2	6	7	12	15	20	17	50	32	41	-	123
15 P	0	0	4	8	17	29	32	90	62	89	-	158
T	2	6	11	20	32	49	49	140	94	130	-	281
S	2	6	8	12	16	22	18	59	37	50	-	131
20 P	0	0	4	8	17	30	32	92	62	91	-	160
T	2	6	12	20	33	52	50	151	99	141	-	291
S	2	6	8	13	17	25	20	61	45	53	-	154
25 P	0	0	4	8	17	30	32	93	63	91	-	163
T	2	6	12	21	34	55	52	154	108	144	-	317
S	2	6	8	13	18	26	20	72	45	57	-	162
30 P	0	0	4	8	17	30	32	96	63	92	-	165
T	2	6	12	21	35	56	52	168	108	149	-	327
S	3	7	10	15	23	36	30	95	56	79	-	212
ALL P	0	0	4	8	17	30	33	96	63	93	-	167
ALL T	3	7	14	23	40	66	63	191	119	172	-	379
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 300.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
S	1	2	1	1	1	1	1	1	1	1	-	1
1 P	0	0	2	4	8	12	17	19	33	42	-	49
T	1	2	3	5	9	13	18	20	34	42	-	50
S	1	4	6	7	10	6	8	20	8	28	-	34
5 P	0	0	2	6	15	24	28	63	50	77	-	111
T	1	4	8	13	25	30	36	83	58	105	-	145
S	2	5	7	11	14	11	13	40	21	36	-	92
10 P	0	0	2	6	16	26	30	83	59	84	-	147
T	2	5	9	17	30	37	43	123	80	120	-	239
S	2	6	7	12	15	20	17	50	33	41	-	123
15 P	0	0	2	6	16	29	30	89	61	81	-	157
T	2	6	9	18	31	49	47	139	94	128	-	180
S	2	6	8	12	17	22	19	59	37	50	-	131
20 P	0	0	2	6	16	29	30	91	61	88	-	160
T	2	6	10	18	33	51	49	150	98	138	-	291
S	2	6	9	13	17	26	20	61	47	55	-	154
25 P	0	0	2	6	16	29	30	92	61	88	-	164
T	2	6	11	19	33	55	50	153	108	143	-	318
S	2	6	9	13	19	27	21	73	47	60	-	162
30 P	0	0	2	6	16	29	30	94	61	88	-	166
T	2	6	11	19	35	56	51	167	108	148	-	328
S	3	7	10	15	23	37	30	96	58	82	-	215
ALL P	0	0	2	6	16	29	30	94	61	88	-	167
T	3	7	12	21	39	66	60	190	119	170	-	381
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 400.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
S	1	2	4	3	1	1	1	1	1	1	-	1
1 P	0	0	0	2	7	13	18	17	38	52	-	56
T	1	2	4	5	8	14	19	18	39	53	-	57
S	1	4	10	11	12	11	14	20	10	32	-	34
5 P	0	0	0	2	12	22	28	63	57	115	-	135
T	1	4	10	13	24	33	42	83	67	147	-	169
S	2	5	10	14	17	18	22	45	26	56	-	92
10 P	0	0	0	2	12	23	28	82	69	115	-	191
T	2	5	10	16	29	41	50	127	95	171	-	283
S	2	6	11	16	18	29	26	60	45	65	-	126
15 P	0	0	0	2	12	25	28	82	69	115	-	209
T	2	6	11	18	30	54	54	142	114	180	-	335
S	2	6	12	16	20	33	28	72	51	80	-	135
20 P	0	0	0	2	12	25	28	82	69	115	-	212
T	2	6	12	18	32	58	56	154	120	195	-	347
S	2	6	12	17	21	36	29	74	60	85	-	176
25 P	0	0	0	2	12	25	28	82	69	115	-	215
T	2	6	12	19	33	61	57	156	129	200	-	391
S	2	6	13	17	22	37	30	87	60	90	-	187
30 P	0	0	0	2	12	25	28	82	69	115	-	215
T	2	6	13	19	34	62	58	169	129	205	-	402
S	3	7	15	19	26	48	39	110	72	112	-	247
ALL P	0	0	0	2	12	25	28	82	69	115	-	215
T	3	7	15	21	38	73	67	192	141	227	-	462
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	315

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 500.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
1 S	1	2	4	7	2	5	9	1	1	1	-	1
1 P	0	0	0	0	6	12	27	23	59	78	-	82
1 T	1	2	4	7	8	17	36	24	60	79	-	83
5 S	1	4	10	17	22	24	37	26	30	129	-	34
5 P	0	0	0	0	8	18	35	82	77	168	-	196
5 T	1	4	10	17	30	42	72	108	107	297	-	230
10 S	2	5	10	21	26	32	45	89	65	153	-	129
10 P	0	0	0	0	8	18	35	82	88	168	-	301
10 T	2	5	10	21	34	50	80	171	153	321	-	430
15 S	2	6	11	23	28	48	50	104	86	162	-	231
15 P	0	0	0	0	8	18	35	82	88	168	-	301
15 T	2	6	11	23	36	66	85	186	174	330	-	532
20 S	2	6	12	23	30	51	51	116	91	177	-	245
20 P	0	0	0	0	8	18	35	82	88	168	-	301
20 T	2	6	12	23	38	69	86	198	180	345	-	546
25 S	2	6	12	24	30	55	53	118	101	182	-	293
25 P	0	0	0	0	8	18	35	82	88	168	-	301
25 T	2	6	12	24	38	73	88	200	189	350	-	594
30 S	2	6	13	24	32	56	53	131	101	187	-	304
30 P	0	0	0	0	8	18	35	82	88	168	-	301
30 T	2	6	13	24	40	74	88	213	189	355	-	605
ALL S	3	7	15	31	36	66	63	154	112	209	-	363
ALL P	0	0	0	0	8	18	35	82	88	168	-	301
ALL T	3	7	15	31	44	84	98	236	200	377	-	664
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : 600.

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
S	1	2	4	11	8	18	29	1	21	1	-	1
1 P	0	0	0	0	3	8	31	27	79	125	-	131
T	1	2	4	11	11	26	60	28	100	126	-	132
S	1	4	10	22	32	42	67	87	79	268	-	34
5 P	0	0	0	0	4	13	35	73	104	228	-	460
T	1	4	10	22	36	55	102	160	183	496	-	494
S	2	5	11	26	37	51	75	150	133	292	-	386
10 P	0	0	0	0	4	13	35	73	104	228	-	534
T	2	5	11	26	41	64	110	223	237	520	-	920
S	2	6	12	27	39	68	79	165	154	301	-	488
15 P	0	0	0	0	4	13	35	73	104	228	-	484
T	2	6	12	27	43	81	114	238	258	529	-	972
S	2	6	12	28	40	71	81	177	160	316	-	502
20 P	0	0	0	0	4	13	35	73	104	228	-	484
T	2	6	12	28	44	84	116	250	264	544	-	986
S	2	6	13	28	41	74	83	179	169	321	-	550
25 P	0	0	0	0	4	13	35	73	104	228	-	484
T	2	6	13	28	45	87	118	252	273	549	-	1034
S	2	6	13	29	43	76	83	192	169	326	-	561
30 P	0	0	0	0	4	13	35	73	104	228	-	484
T	2	6	13	29	47	89	118	265	273	554	-	1045
S	3	7	15	31	47	86	93	215	181	348	-	620
ALL P	0	0	0	0	4	13	35	73	104	228	-	484
ALL T	3	7	15	31	51	99	128	288	285	576	-	1104
TTL DOCS	158	136	193	151	225	248	241	257	238	313	-	165

TEST COLLECTION: KEEN.

PRE-SEARCH THRESHOLD WEIGHT : α

	QUERY SIZE (and population)											
No	2	3	4	5	6	7	8	9	10	11	12	13
DOCS	6	11	8	14	8	4	6	1	2	1	0	2
S	1	2	4	11	21	47	173	188	797	1369	-	5061
1 P	0	0	0	0	0	0	0	0	0	0	-	0
T	1	2	4	11	21	47	173	188	797	1369	-	5061
S	1	4	10	22	48	83	229	383	921	1967	-	7446
5 P	0	0	0	0	0	0	0	0	0	0	-	0
T	1	4	10	22	48	83	229	383	921	1967	-	7446
S	2	5	11	26	53	92	237	446	975	1991	-	7906
10 P	0	0	0	0	0	0	0	0	0	0	-	0
T	2	5	11	26	53	92	237	446	975	1991	-	7906
S	2	6	12	27	54	108	241	461	996	2000	-	8059
15 P	0	0	0	0	0	0	0	0	0	0	-	0
T	2	6	12	27	54	108	241	461	996	2000	-	8059
S	2	6	12	28	56	112	243	473	1002	2015	-	8072
20 P	0	0	0	0	0	0	0	0	0	0	-	0
T	2	6	12	28	56	112	243	473	1002	2015	-	8072
S	2	6	13	28	57	115	245	475	1011	2020	-	8120
25 P	0	0	0	0	0	0	0	0	0	0	-	0
T	2	6	13	28	57	115	245	475	1011	2020	-	8120
S	2	6	13	29	58	116	245	488	1011	2025	-	8131
30 P	0	0	0	0	0	0	0	0	0	0	-	0
T	2	6	13	29	58	116	245	488	1011	2025	-	8131
S	3	7	15	31	63	127	255	511	1023	2047	-	8191
ALL P	0	0	0	0	0	0	0	0	0	0	-	0
ALL T	3	7	15	31	63	127	255	511	1023	2047	-	8191
TTL DOCS	158	136	193	151	225	248	241	357	238	313	-	165

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 0.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	1	2	1	1	1	1	1	1	1	1	1	1
1 P	-	0	5	8	15	19	23	27	24	31	36	39	45	58
T	-	1	6	10	16	20	24	28	25	32	37	40	46	59
S	-	3	3	6	7	11	19	24	38	64	185	355	146	74
5 P	-	0	5	9	19	27	39	50	54	72	91	149	175	164
T	-	3	8	15	26	38	58	74	92	136	276	504	321	238
S	-	3	5	11	12	20	32	46	69	128	278	471	213	224
10 P	-	0	5	9	20	31	45	59	69	96	121	208	217	211
T	-	3	10	20	32	51	77	105	138	224	399	679	430	435
S	-	3	6	12	16	26	41	61	94	204	342	554	259	328
15 P	-	0	5	9	20	31	48	62	75	112	138	239	245	250
T	-	3	11	21	36	57	89	123	169	316	480	793	504	578
S	-	4	6	14	19	31	49	75	113	238	405	611	289	?
20 P	-	0	5	9	20	32	49	65	79	118	152	255	262	?
T	-	4	11	23	39	63	98	140	192	356	557	866	551	?
S	-	4	7	15	22	36	55	86	128	265	449	681	327	?
25 P	-	0	5	9	20	32	50	67	83	121	160	277	276	?
T	-	4	12	24	42	68	105	153	211	386	609	958	603	?
S	-	5	7	16	23	39	59	93	142	300	500	731	361	?
30 P	-	0	5	9	20	32	51	68	85	125	169	292	287	?
T	-	5	12	25	43	71	110	161	227	425	669	1023	648	?
S	-	7	12	26	46	74	123	195	303	566	1042	1196	678	?
ALL P	-	0	5	9	20	33	51	70	92	137	192	343	321	?
ALL T	-	7	17	35	66	107	174	265	395	703	1234	1539	999	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	?

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 100.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	1	2	1	1	1	1	1	1	1	1	1	1
1 P	-	0	5	7	15	19	23	27	24	31	36	39	45	58
T	-	1	6	9	16	20	24	28	25	32	37	40	46	59
S	-	3	3	6	7	11	19	24	38	64	185	355	146	74
5 P	-	0	5	8	19	27	39	50	54	72	91	149	175	164
T	-	3	8	14	26	38	58	74	92	136	276	504	321	238
S	-	3	5	11	12	29	32	46	69	128	278	471	213	224
10 P	-	0	5	8	20	31	45	59	69	96	121	208	217	211
T	-	3	10	19	32	50	77	105	138	224	399	679	430	435
S	-	3	6	12	16	26	41	61	94	204	342	554	259	328
15 P	-	0	5	8	20	31	48	62	75	112	138	239	245	250
T	-	3	11	20	36	57	89	123	169	316	480	793	504	578
S	-	4	6	14	19	31	49	75	113	238	405	611	289	?
20 P	-	0	5	8	20	32	49	65	79	118	152	255	262	?
T	-	4	11	22	39	63	98	140	192	356	557	866	551	?
S	-	4	7	15	22	36	55	86	128	265	449	681	327	?
25 P	-	0	5	8	20	32	50	67	83	121	160	277	276	?
T	-	4	12	23	42	68	105	153	211	386	609	958	603	?
S	-	5	7	16	23	39	59	93	142	300	500	731	361	?
30 P	-	0	5	.8	20	32	51	68	85	125	169	292	287	?
T	-	5	12	24	43	71	110	161	227	425	669	1023	648	?
S	-	7	12	26	46	74	123	195	303	566	1042	1196	678	?
ALL P	-	0	5	8	20	33	51	70	92	137	192	343	321	?
ALL T	-	7	17	34	66	107	174	265	395	703	1234	1539	999	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	?

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 200.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	1	2	1	1	1	1	1	1	1	1	1	1
1 P	-	0	3	5	14	19	23	27	24	31	36	39	45	58
T	-	1	4	7	15	20	24	28	25	32	37	40	46	59
S	-	3	3	6	7	11	19	24	38	64	185	355	146	74
5 P	-	0	4	5	18	27	39	50	54	72	91	149	175	164
T	-	3	7	11	25	38	58	74	92	136	277	504	321	238
S	-	3	5	11	12	20	32	46	69	128	278	471	213	224
10 P	-	0	4	5	18	31	45	59	69	96	121	208	217	211
T	-	3	9	16	30	51	77	105	138	224	399	679	430	435
S	-	3	6	12	16	26	41	61	94	204	342	554	259	328
15 P	-	0	4	5	19	31	48	62	75	112	138	239	245	250
T	-	3	10	17	35	57	89	123	169	316	480	793	504	518
S	-	4	6	14	19	31	49	75	113	238	405	611	289	?
20 P	-	0	4	5	19	32	49	65	79	117	152	255	262	?
T	-	4	10	19	38	63	98	140	192	355	557	866	551	?
S	-	4	7	15	22	36	55	86	128	265	449	681	327	?
25 P	-	0	4	5	19	32	50	77	83	121	160	277	276	?
T	-	4	11	20	41	68	105	153	211	386	609	958	603	?
S	-	5	7	16	23	39	59	93	142	300	500	731	361	?
30 P	-	0	4	5	19	32	51	68	85	125	169	292	287	?
T	-	5	11	21	42	71	110	161	227	425	669	1023	648	?
S	-	7	12	26	46	74	123	195	303	566	1042	1196	678	?
ALL P	-	0	4	5	19	33	51	70	92	137	192	343	321	?
ALL T	-	7	16	31	65	107	174	265	395	703	1234	1539	999	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	?

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 300.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	1	2	1	1	1	1	1	1	1	1	1	1
1 P	-	0	0	2	9	17	22	27	24	30	35	39	44	55
T	-	1	1	4	10	18	23	28	25	31	36	40	45	56
S	-	3	4	7	7	11	19	24	38	64	185	355	146	74
5 P	-	0	1	2	12	25	37	49	52	70	90	148	172	161
T	-	3	5	9	19	36	56	73	90	134	275	503	318	235
S	-	3	5	11	12	20	32	46	69	128	278	471	213	224
10 P	-	0	1	2	12	27	43	58	67	95	119	206	213	207
T	-	3	6	13	24	47	75	104	136	223	397	677	426	431
S	-	3	6	13	16	26	41	61	94	204	342	554	259	328
15 P	-	0	1	2	12	29	47	62	74	111	136	237	240	246
T	-	3	7	15	28	55	88	123	168	315	478	791	499	574
S	-	4	6	14	19	32	49	75	113	238	405	611	289	360
20 P	-	0	1	2	12	29	47	64	78	117	150	253	258	253
T	-	4	7	16	31	61	96	139	191	355	555	864	547	613
S	-	4	7	15	22	36	55	86	129	265	449	681	327	539
25 P	-	0	1	2	12	29	48	66	81	120	158	276	272	289
T	-	4	8	17	34	65	103	154	210	385	607	957	599	828
S	-	5	8	16	23	39	60	94	142	300	500	731	361	658
30 P	-	0	1	2	12	29	49	67	81	124	168	290	283	311
T	-	5	9	18	35	68	109	161	223	424	668	1021	644	769
S	-	7	13	27	46	74	123	196	304	568	1043	1197	683	3621
ALL P	-	0	1	2	12	29	49	69	89	135	190	341	314	447
ALL T	-	7	14	29	58	103	172	264	393	703	1233	1538	997	4068
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	1059

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 400.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	1	3	2	1	1	1	1	1	1	1	1	1
1 P	-	0	0	0	2	11	19	26	25	31	36	41	46	63
T	-	1	1	3	4	12	20	27	26	32	37	42	47	64
S	-	3	4	9	10	14	20	24	38	64	185	355	146	74
5 P	-	0	0	0	3	16	34	50	56	74	96	154	193	193
T	-	3	4	9	13	30	54	74	94	138	281	509	339	267
S	-	3	5	14	17	25	35	49	72	128	278	471	213	224
10 P	-	0	0	0	3	18	40	59	73	105	131	220	246	260
T	-	3	5	14	20	43	75	108	145	232	409	691	459	484
S	-	3	6	15	21	32	46	68	100	207	342	554	259	328
15 P	-	0	0	0	3	18	41	62	80	124	153	256	283	311
T	-	3	6	15	24	50	87	130	180	331	495	810	542	639
S	-	4	7	17	25	38	56	84	121	244	408	611	289	360
20 P	-	0	0	0	3	18	41	63	86	133	169	275	310	321
T	-	4	7	17	28	56	97	147	207	377	577	886	599	681
S	-	4	8	18	28	43	62	97	139	278	453	681	334	539
25 P	-	0	0	0	3	18	41	64	88	146	181	306	333	373
T	-	4	8	18	31	61	103	161	227	414	634	987	667	912
S	-	5	8	19	29	46	67	105	154	320	506	733	373	658
30 P	-	0	0	0	3	18	41	66	90	141	188	322	350	408
T	-	5	8	19	32	64	108	171	244	461	694	1055	723	1066
S	-	7	15	31	52	82	132	210	332	610	1076	1252	788	3850
ALL P	-	0	0	0	3	18	41	65	93	147	214	371	359	711
ALL T	-	7	15	31	55	100	173	275	425	757	1290	1623	1147	4561
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	1059

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : 500.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
1 S	-	1	2	4	7	6	2	1	3	1	1	1	1	1
1 P	-	0	0	0	0	6	18	26	33	40	43	54	66	105
1 T	-	1	2	4	7	12	20	27	36	41	44	55	67	106
5 S	-	3	5	10	18	24	34	39	55	77	185	355	146	74
5 P	-	0	0	0	0	9	30	55	77	103	130	198	324	401
5 T	-	3	5	10	18	33	64	94	132	180	315	553	470	475
10 S	-	3	6	15	25	38	58	78	107	169	280	471	295	224
10 P	-	0	0	0	0	9	31	60	98	141	190	303	432	622
10 T	-	3	6	15	25	47	89	138	205	310	470	774	727	846
15 S	-	3	7	17	30	46	73	101	160	282	372	565	410	328
15 P	-	0	0	0	0	9	31	60	101	164	222	368	501	721
15 T	-	3	7	17	30	55	104	161	261	446	594	933	911	1249
20 S	-	4	8	18	33	52	83	121	193	341	454	651	521	?
20 P	-	0	0	0	0	9	31	61	103	167	249	390	512	?
20 T	-	4	8	18	33	61	114	182	296	508	703	1041	1033	?
25 S	-	4	9	19	36	57	90	135	218	384	510	767	647	?
25 P	-	0	0	0	0	9	31	61	103	170	261	426	514	?
25 T	-	4	9	19	36	66	121	196	321	554	771	1193	1161	?
30 S	-	5	9	20	37	60	95	144	237	445	586	838	745	?
30 P	-	0	0	0	0	9	31	62	104	172	274	445	514	?
30 T	-	5	9	20	37	69	126	206	341	617	860	1283	1259	?
ALL S	-	7	15	31	63	96	160	252	422	755	1237	1489	1185	?
ALL P	-	0	0	0	0	9	31	62	104	172	280	460	514	?
ALL T	-	7	15	31	63	105	191	314	526	927	1517	1949	1699	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	?

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT :600.

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
1 S	-	1	2	4	9	14	18	10	17	9	1	1	1	1
1 P	-	0	0	0	0	3	15	30	46	65	65	84	96	222
1 T	-	1	2	4	9	17	33	40	63	74	66	85	97	223
5 S	-	3	5	10	21	36	66	89	134	171	205	355	451	74
5 P	-	0	0	0	0	5	20	50	91	138	232	348	232	348
5 T	-	3	5	10	21	41	86	139	225	309	437	703	1077	1076
10 S	-	3	7	15	28	52	92	137	229	328	378	584	887	224
10 P	-	0	0	0	0	5	21	53	96	171	328	533	712	1897
10 T	-	3	7	15	28	57	113	190	325	499	706	1117	1599	2121
15 S	-	3	8	17	32	59	107	162	289	500	560	837	1204	328
15 P	-	0	0	0	0	5	21	53	96	179	341	573	712	2463
15 T	-	3	8	17	32	64	128	215	385	679	901	1410	1916	2791
20 S	-	4	8	18	36	65	117	183	328	568	719	969	1346	?
20 P	-	0	0	0	0	5	21	53	96	179	342	590	712	?
20 T	-	4	8	18	36	70	238	236	424	747	1061	1559	2058	?
25 S	-	4	9	19	39	71	124	197	354	620	812	1212	1475	?
25 P	-	0	0	0	0	5	21	53	96	179	342	591	712	?
25 T	-	4	9	19	39	76	145	250	450	799	1154	1803	2187	?
30 S	-	5	9	20	40	74	129	207	374	683	916	1331	1574	?
30 P	-	0	0	0	0	5	21	53	96	179	342	591	712	?
30 T	-	5	9	20	40	79	150	261	470	862	1258	1922	2286	?
ALL S	-	7	15	31	63	109	194	317	559	993	1582	2014	2014	?
ALL P	-	0	0	0	0	5	21	53	96	179	342	591	712	?
ALL T	-	7	15	31	63	114	215	370	655	1172	1924	2605	2726	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	1081	?

TEST COLLECTION: CRANFIELD 1400. PRE-SEARCH THRESHOLD WEIGHT : ∞

	QUERY SIZE (and population)													
No	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DOCS	0	10	10	31	22	30	38	20	25	14	8	10	5	1
S	-	1	2	4	9	26	62	127	304	509	1068	2306	?	?
1 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	1	2	4	9	26	62	127	304	509	1068	2306	?	?
S	-	3	5	10	21	53	125	275	583	1090	2425	5546	?	?
5 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	3	5	10	21	53	125	275	583	1090	2425	5546	?	?
S	-	3	7	15	28	69	152	331	692	1355	2858	6632	?	?
10 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	3	7	15	28	69	152	331	692	1355	2858	6632	?	?
S	-	3	8	17	32	77	167	356	753	1553	3069	6976	?	?
15 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	3	8	17	32	77	167	356	753	1553	3069	6976	?	?
S	-	4	8	18	36	83	177	377	791	1621	3231	7144	?	?
20 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	4	8	18	36	83	177	377	791	1621	3231	7144	?	?
S	-	4	9	19	39	88	184	391	818	1673	3324	7388	?	?
25 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	4	9	19	39	88	184	391	818	1673	3324	7388	?	?
S	-	5	9	20	40	91	190	401	838	1736	3429	7507	?	?
30 P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	5	9	20	40	91	190	401	838	1736	3429	7507	?	?
S	-	7	15	31	63	127	255	511	1023	2047	4095	8191	?	?
ALL P	-	0	0	0	0	0	0	0	0	0	0	0	?	?
T	-	7	15	31	63	127	255	511	1023	2047	4095	8191	?	?
TTL DOCS	-	343	335	603	642	735	773	786	964	946	891	931	-	-

References

The following abbreviations have been used within the list of references:-

ACM - Association for Computing Machinery.

AFIPS - American Federation of Information Processing Societies.

ARIST - Annual Review of Information Science and Technology.

ASIS - American Society for Information Science.

CACM - Communications of the Association for Computing Machinery.

Conf. - conference.

IEEE - Institute of Electrical and Electronics Engineers.

Intl. - international.

JACM - Journal of the Association for Computing Machinery.

JASIS - Journal of the American Society for Information Science.

J.Doc.- Journal of Documentation.

Procs.- proceedings.

Trans.- transactions.

Agrawal JP,Ninan J
Hardware Considerations in FFT Processors.
Intl. Conference on Acoustics and Speech Signal Processing, USA 1976, IEEE, (1976), 618-621

Aho AV
Algorithms and Computational Complexity.
Acta Crystallographica (Section A), 33(1977), 5-12

Aho AV,Hopcroft JE,Ullman JD
The Design and Analysis of Computer Algorithms.
Addison-Wesley, New York., (1974),

Ahrens JH,Finke G
Merging and Sorting Applied to the Zero-One Knapsack Problem.
Operational Research, 23(1975), 1099-1109

Albano A,Orsini R
A Tree Search Approach to the M-Partition and Knapsack Problems.
Computer Journal, 23(1980), 256-261

Alsberg PA,Bailey JF
Intelligent Terminals as User Agents.
Symposium on Trends and Applications of Mini and Micro Systems, USA, May 1976, IEEE, (1976), 129-135

Angione PV
On the Equivalence of Boolean and Weighted Searching Based on the Convertibility of Query Forms.
JASIS, 26(1975), 112-124

Areu E,Brewer C,Schneiderman B
Experimental Comparison of Boolean Operators in Bibliographical Retrieval Systems.
Procs. of the ASIS Annual Meeting, 15(1978), 13-16

Artandi S
On-line Information Systems in Perspective.
Journal of Chemical Information and Computer Science, 16(1976), 80-81

Artandi S
Man, Information and Society: New Patterns of Interaction.
JASIS, 30(1979), 15-18

Artandi S,Wolf EH
The Effectiveness of Automatically Generated Weights and Links in Mechanical Indexing.
American Documentation, 20(1969), 198-202

Atherton P
Standards for a User-System Interface Language in On-Line Retrieval Systems.
Online Review, 2(1978), 57-61

Attar R,Fraenkel AS
Local Feedback in Full-Text Retrieval Systems.
JACM, 24(1977), 397-417

Babb E
Implementing a Relational Database by means of Specialised Hardware.
ACM Transactions on Database Systems, 4(1979), 1-29

Backler J
Raising CRT-Terminal IQs.
Digital Design, 6(Sep 1976), 39-56

Baer JL
Multiprocessing Systems.
IEEE Trans. on Computers, 25(1976), 1271-1277

Banerjee J,Hsiao DK,Kannan K
Concepts and Capabilities of a Data Base Computer.
ACM Transactions on Database Systems, 3(1978), 347-384

Banerjee J,Hsiao DK,Kannan K
DBC - A Database Computer for Very Large Databases.
IEEE Trans. on Computers, 28(1979), 414-429

Barber AS,Barraclough ED,Gray WA
Medlars On-line Search Formulation and Indexing.
The University of Newcastle upon Tyne, Computing Laboratory Technical Report No. 34(Jun 1972).

Barber AS,Barraclough ED,Gray WA
On-line Information Retrieval as a Scientists Tool.
Information Storage and Retrieval, 9(1973), 429-40

Barker FH,Veal D,Wyatt BK
Towards Automatic Profile Construction.
J.Doc., 28(1972), 44-55

Barraclough ED
On-Line Searching in Information Retrieval.
J.Doc., 33(1977), 220-238

Barraclough ED
Opportunities for Testing with On-Line Systems.
in Information Retrieval Experiment, Ed. K. Sparck-Jones, publ. Butterworths, (1981).

Barraclough ED,Hunter JA,Lovett AJ,Rossiter BN
The Medusa Current Awareness Experiment.
The University of Newcastle upon Tyne, Computing Laboratory Technical Report No. 40(Oct 1975).

Baum RI,Hsiao DK
Database Computers - A Step Towards Data Utilities.
IEEE Trans. on Computers, 25(1976), 1254-1258

Beaven PA,Lewin DW
An Associative Parallel Processing System for Non-numerical Computation.
Computer Journal, 15(1973), 343-349

Beckenbach EF
Applied Combinatorial Mathematics.
John Wiley & Sons, New York, USA, (1964).

Belkin N
Ineffable Concepts in Information Retrieval.
in Information Retrieval Experiment, Ed. K. Sparck-Jones, publ. Butterworths, (1981).

Bennett JL
The User Interface in Interactive Systems.
ARIST, 7(1972), 159-196

Bentley JL
An Introduction to Algorithm Design.
Computer, 12,2(1979), 66-78

Bergland GD
Fast Fourier Transform Hardware Implementations - An Overview.
IEEE Trans. on Audio and Electroacoustics, 17(1969), 162-165

Bernhard R
Bubbles Take on Disks.
IEEE Spectrum, 17,5(1980), 30-33

Berra PB
Associative Processors and Data Base Management.
Parallel Computers - Parallel Mathematics(Ed. M.Feilmeier), Intl. Assoc. for Maths and Computers in Simulation, (1977), 39-48

Berra PB,Oliver E
The Role of the Associative Array Processor in Data Base Machine Architecturte.
Computer, 12,3(1979), 53-61

Bhandarkar DP
The Impact of Semiconductor Technology on Computer Systems.
Computer, 12,9(1979), 92-98

Booth GM
Hierarchical Configuarations for Distributed Processing.
Procs. of the 15th IEEE Computer Society Intl. Conference.
IEEE, (1977), 32-33

Bourne CP
On-line Systems: History, Technology, And Economics.
JASIS, 31(1980), 155-160

Bradley GH
Transformation of Integer Programs to Knapsack Problems
Discrete Mathematics, 1(1971), 29-45

Brandhorst WT
Simulation of Boolean Logic Constraints Through the Use of Term Weights.
American Documentation, 17(1966), 145-146

British Library
BLAISE User Manual.
British Library, (1980),

Brown AR
Optimum Packing and Depletion.
MacDonald, (1971),

Bush V
As We May Think.
Atlantic Monthly, 176(1945), 101-108

Cabot AV
An Emuneration Algorithm for Knapsack Problems.
Operational Research, 18(1970), 306-311

Canaday RH,Harrison RD,Ivie EL,Ryder JL,Wehr LA
A Back-end Computer for Data Management.
CACM, 17(1974), 575-582

Capon PC,Morris D,Rohl JS,Wilson IR
The MU5 Compiler Target Language and Autocode
Computer Journal, 15(1972), 109-112

Carroll JM
Prospects for Parallelism and the Computer Crunch.
JASIS, 27(1976), 63-69

Carroll JM,Roelloffs R
Computer Selection of Keywords Using Word Frequency Analysis.
American Documentation, 20(1969), 227-233

Cattley JM,Moore JE,Banks DG,O'Leary PT
Retrieving Patents by Weighted Term Searching.
Chemical Engineering Progress, 62(1966), 91-96

Cawkwell AE
The New Technology
ASLIB Procs., 31(1979), 29-32

Chai DT
An Information Retrieval System Using Keyword Dialog.
Information Storage and Retrieval, 9(1973), 373-387

Champine GA
Back-End Technology Trends.
Computer, 13,2(1980), 50-58

Chen TC
Magnetic Bubble Memory and Logic.
Advances in Computers, 17(1978), 223-282

Chevance RJ
Design of High Level Language Oriented Processors.
Sigplan Notices, 12(1977), 40-51

Cleverdon C,Mills J,Keen M
Cranfield Research Project: Factors Determining the Performance of Indexing Systems.
Cranfield College of Technology, Cranfield, England., (1966),

Colthurst JP,Shilling ME
On-line Searching in a Research Environment.
On-line Review, 1(1977), 311-317

Coulouris GF,Evans JM,Mitchell RW
Towards Content addressing in Data Bases.
Computer Journal, 15(1972), 95-98

Croft WB
Clustering Large Files of Documents Using the Single-Link Method.
JASIS, 28(1977), 341-344

Cuadra CA
On-Line Systems: Promises and Pitfalls.
JASIS, 22(1971), 107-114

Cuadra CA
Commercially Funded On-line Retrieval Services - Past, Present and Future.
ASLIB Procs., 30(1978), 2-15

D'Alleyrand MR
Holography and Future Information Systems.
Procs. of the ASIS Annual Meeting, 15(1978), 93-94

Dattola RT
FIRST: Flexible Information Retieval System for Text.
JASIS, 30(1979), 9-14

Davis CH
A Simple Program for Weighted Term Searching.
Special Libraries, 63(1972), 381-384

DeWitt DJ
DIRECT - A Multiprocessor Organisation for Supporting Relational Database Management Systems.
IEEE Trans. on Computers, 28(1979), 395-406

Dillon M.Desper J
The Use of Automatic Relevance Feedback in Boolean Retrieval Systems.
J.Doc., 36(1980), 197-208

Dominick WD.Penniman WD
Automated Monitoring to Support the Analysis and Evaluation of Information Systems.
Procs. of the 2nd Intl. Conference on Information Storage and Retrieval, Dallas, USA, ACM, (1979), 2-9

Doszkocs TE
An Associative Interactive Dictionary for On-line Searching.
On-line Review, 2(1978a), 163-173

Doszkocs TE
An Associative Interactive Dictionary (AID) for Online Bibliographic Searching.
Procs. of the ASIS Annual Meeting, 15(1978b), 105-109

Doszkocs TE,Rapp BA
Searching Medline in English: A Prototype User Interface with Natural Language Query, Ranked Output, and Relevance Feedback.
Procs. of the ASIS Annual Meeting, 42(1979), 131-137

Doty KL,Greenblatt JD,Su SYW
Magnetic Bubble Memory Architectures for Supporting Associative Searching of Relational Databases.
IEEE Trans. on Computers, 29(1980), 957-970

Duff MJB,Watson DM
The Cellular Logic Array.
Computer Journal, 20(1977), 68-72

EEC
The European On-Line Information Network.
Commision of the European Communities, Kirchberg, Luxembourg., (1976),

Elsworth TE
Compilation via an Intermediate Language.
University of Aston in Birmingham Technical Report No. 7801(1978),

Enslow PH
Multiprocessor Organisation - A Survey.
Computing Surveys, 9(1977), 103-129

Fayard D,Plateau GL
Resolution of the 0-1 Knapsack Problem: Comparison of Methods
Mathematical Programming, 8(1975), 272-307

Fedida S
Viewdata: An Interactive Information Medium for the General Public.
The Post Office Electrical Engineers Journal, 71(1978), 21-27

Fedida S,Malik R
The Viewdata Revolution.
Associated Business Press, (1979),

Fitzgerald J,Easton S
Fundamentals of Data Communications
Wiley/Hamilton, New York, (1978),

Foster CC
Content Addressable Parallel Processors.
Van Nostrand, (1976),

Foster D,Monkhouse V,Stone MB
User Satisfaction in Accessing Multiple Bibliographic Databases.
Procs. of the 4th Open Conference on Information Science. Canadian Association for Inf. Sci., (1977).

Franklin INSTITUTE
OL'SAM - OnLine Search Assistance Machine.
Franklin Institute Research Laboratory, Inc., Philadelphia, USA., (1980).

Gagliardi UO
Trends in Computing System Architecture.
Procs. of the IEEE, 163(1975), 858-862

Garey MR,Johnson DS
"Strong" NP-Completeness Results: Motivation, Examples and Implications.
JACM, 25(1978), 499-508

Gebhardt F,Stellmacher I
Opinion Paper: Design Criteria for Documentation Retrieval Languages.
JASIS, 29(1978), 191-199

Gilmore PC,Gomory RE
The Theory and Computation of Knapsack Functions
Operational Research, 14(1966), 1045-1075

Goldstein CM,Ford WH
The User-cordial Interface.
Online Review, 2(1978), 269-275

Greenberg H,Hegerich RL
A Branch Search Algorithm for the Knapsack Problem.
Mangement Science, 16(1970), 327-332

Grover D
Visual Display Units and Their Applications
IPC Science and Technology Press Ltd., (1976).

Guaccimanni LA,Bucci G
The Influence of Computer Hardware and Communication Lines on the Cost of Distributed Computer Systems.
Computer Networks and Remote Data Processing(Germany), (Mar 1976), 61-73

Hall JL
On-line Information Retrieval Sourcebook.
ASLIB, (1977).

Hall JL,Negus AE,Dancy DJ
On-line Information Retrieval: A Method of Query Formulation Using a Video Terminal.
Program, 6(1972), 175-186

Hall PAV,Dowling GR
Approximate String Matching.
Computing Surveys, 12(1980), 381-402

Hankins HCA
Interactive Television.
Advance, 15(Oct 1974), 34-37

Hardy GH,Wright EM
An Introduction to the Theory of Numbers.
Clarendon Press,Oxford, England., (1959),

Harley AJ,LeMinor M,Weil A
Towards a Universal Search Formulation Language for Information Retrieval.
J.Doc., 28(1972), 202-213

Harper DJ,VanRijsbergen CJ
An Evaluation of Feedback in Document Retrieval Using Co-occurrence Data.
J.Doc., 34(1978), 189-216

Harris CL,Roberts SH
The Search for Tomorrow: Low Cost, Full Text Search with Minimum Front End Investment.
Procs. of the ASIS Annual Meeting, 15(1978), 368-372

Haskin R
Hardware for Searching Very Large Text Databases.
SIGIR Forum, 15,2(1980), 49-56

Hawkins DT
On-line Information Retrieval Bibliography, 1965-1976.
On-line Review(supplement), 1(1977a), 1-55

Hawkins DT
Unconventional Uses of On-line Information Retrieval Systems: On-line Bibliometric Studies.
JASIS, 28(1977b), 13-18

Hawkins DT
On-line Information Retrieval Bibliography (First Update).
On-line Review, 2(1978), 63-106

Hawkins DT
On-line Information Retrieval Bibliography: Second Update.
On-line Review, 3(Mar 1979), 37-73

Heaps HS
Information Retrieval: Computational and Theoretical Aspects.
Academic Press, London, (1978).

Hillman B,Dominick WD
Developing Monitoring Techniques for an On-Line Information Retrieval System.
Information Storage and Retrieval, 9(1973), 297-308

Hillman DJ
Negotiations of Inquiries in an On-Line Retrieval System.
Information Storage and Retrieval, 4(1968), 219-238

Hillman DJ
Customized Services via Interactions with Leadermart.
Information Storage and Retrieval, 9(1973), 587-596

Hillman DJ,Kasarda AJ
The LEADER Retrieval System.
Procs. of the Spring Joint Computer Conference, AFIPS., 34(1969), 447-455

Hirschberg DS,Wong CK
Polynomial-Time Algorithm for Knapsack Problem with Two Variables.
JACM, 23(1976), 147-154

Hobson GS
Charged Coupled Devices.
IEE Reviews, 124(1977), 925-945

Hollaar LA
A List Merging Processor for Inverted File Information Retrieval Systems.
Ph.D. Thesis, University of Illinois, USA., (1975).

Hollaar LA
Rotating Memory Processors for the Matching of Complex Textual Patterns.
Procs. of the 5th Annual Symposium on Computer Architecture, IEEE., (1978a), 39-43

Hollaar LA
Specialised Merge Processor Networks for Combining Sorted Lists.
ACM Transactions on Database Systems, 3(1978b), 272-284

Hollaar LA
Unconventional Computer Architectures for Information Retrieval.
ARIST, 14(1979a), 129-151

Hollaar LA
A Design for a List Merging Network.
IEEE Trans. on Computers, 28(1979b), 406-413

Hollaar LA
Text Retrieval Computers.
Computer, 12,3(1979c), 40-50

Hollaar LA,Stellhorn WH
A Specialized Architecture for Textual Information Retrieval.
Procs. of the National Computer Conference, AFIPS., 46(1977), 697-702

Holmes PL
On-line Information Retrieval: An Introduction and Guide to the British Library's Short-Term Experimental Information Network Project.
Volumes 1 and 2, British Library R & D Reports no. 5360HC and 5397, (1978).

Holmes PL
The Re-use of Machine Readable Data and Copyright - A Pragmatic Approach to the Problems
Procs. of the 4th Intl. Online Meeting, London, 1980, Learned Information., (1980), 1-13

Horowitz E,Sahni S
Computing Partitions with Application to the Knapsack Problem.
JACM, 21(1974), 277-292

Horowitz E,Sahni S
Fundamentals of Algorithms.
Pitman, London, (1979).

Hsiao DK
Database Machines are Coming, Database Machines are Coming!
Computer, 12,3(1979), 7-9

Hyde J
Multiprocessing
Microprocessors, 1(1977), 385-388

Hyman M
The Role of Mini-Computers in Libraries and Information Units.
ASLIB Procs., 30(1978), 373-382

Ibarra OH,Kim CE
Fast Approximation Algorithms for the Knapack and Sum of Subset Problems.
JACM, 22(1975), 463-468

Iizuka H,Ishii O,Fujii K,Ohomote R,Furuya T
ACE - A New Modular Computer Architecture.
Procs. of 2nd USA-JAPAN Computer Conference. AFIPS. (1975). 36-41

Iker HP
Solution of Boolean Equations Through the Use of Term Weights to the Base Two.
American Documentation. 18(1967). 47-47

Ingargio GP,Korsh JF
General Algorithm for One-Dimensional Knapsack Problems.
Operational Research. 25(1977). 752-759

Intel INC
Intel Component Data Catalogue 1979
Intel Corporation. California. USA. (1979). 911-918

Karp RM
Reducibility Among Combinatorial Problems.
In Complexity of Computer Computations by R.E.Miller & J.W.Thatcher(Eds). Plenum Press. (1972). 85-104

Kartashev SP,Kartashev SI
Supersystems for the 80's.
Computer. 13,11(1980). 11-14

Keenan S
Development of Mechanised Documentation.
J.Doc.. 34(1978). 333-341

Kellog CH
On-Line Translation of Natural Language Questions into Artificial Language Queries.
Information Storage and Retrieval. 4(1968). 287-307

Kelly PTF
The Development of Packet Switched Data Transmission Services in the United Kingdom
Procs. of the Intl. Conference on Communications Equipment and Systems. England. June 76. IEEE.. (1976). 194-198

Kent A
Textbook on Mechanised Information Retrieval.
Wiley. New York. (1966).

Kerr DS
Data Base Machines with Large Content Addressable Blocks and Structural Information Processors.
Computer. 12,3(1979). 64-79

King DW
Introduction to 'Key Papers in the Design and Evaluation of Information Systems'
D.W.King(Ed), Knowledge Industries, New York., (1978).

Kiseda JR,Peterson HE,Seelbach WC,Teig M
A Magnetic Associative Memory.
IBM Journal of Research and Development, 5(1961), 106-121

Knight GR
Holographic Memories.
Optical Engineering, 14(1975), 453-459

Knuth DE
Art of Computer Programming,Vol 2:Seminumerical Algorithms.
Addison Wesley, (1969).

Kober R
The Multiprocessor System SMS 201 - Combining 128 Microprocessors to a Powerful Computer.
Procs. of the 15th IEEE Computer Society Intl. Conference.
IEEE, (1977), 225-230

Kolesar PJ
A Branch and Bound Algorithm for the Knapsack Problem.
Management Science, 13(1967), 723-735

Korfhage RR
The Impact of Personal Computers on Library-Based Information Services.
SIGIR Forum,ACM, 12,4(1978), 10-13

Kraft DH,Lee T
Stopping Rules and Their Effect on Expected Search Length.
Information Processing and Management, 15(1979), 47-58

Kuck DJ
Parallel Processing of Ordinary Programs.
Advances in Computers, 15(1976), 119-179

Lancaster FW
Interaction Between Requesters and a Large Mechanized Retieval System.
Information Storage and Retrieval, 4(1968), 239-252

Lancaster FW
The Cost-Effectiveness Analysis of Information Retrieval and Dissemination Systems.
JASIS, 22(1971a), 12-27

Lancaster FW
Are We Ready For On-Line Information Retrieval?
Procs. of the ACM Annual Conference 1971, ACM, (1971b), 565-568

Lancaster FW,Fayen EG
Information Retrieval On-line.
Melville Publishing Co., (1973).

Langdon GG
A Note on Associative Processors for Data Management.
ACM Transactions on Database Systems, 3(1978), 148-158

Langdon GG
Database Machines: An Introduction.
IEEE Trans. on Computers, 28(1979), 381-383

Laver HC
Discussion on Ph.D. Thesis Proposals in Computing Science
Computer Journal, 18(1976), 279-281

Lea RM
Information Processing with an Associative Parallel Processor.
Computer, 8,11(1975), 25-32

Lea RM
Associative Processing of Non-numerical Information.
Computer Architecture(Ed. GG Boulaye and DW Lewin), Reidel, Netherlands, (1977), 171-215

Lea RM
Text Compression with an Associative Parallel Processor.
Computer Journal, 21(1978), 45-56

Ledgard H,Whiteside JA,Singer A,Seymour W
The Natural Language of Interactive Systems.
CACM, 23(1980), 556-563

Lee CY,Paull MC
A Content Addressable Distributed Logic Memory with Applications to Information Retrieval.
Procs. of the IEEE, 51(1963), 924-932

Lee SY,Chang H
Associative-Search Bubble Devices for Content-Addressable Memory and Array Logic.
IEEE Trans. on Computers, 28(1979), 627-636

Lewin MH
Retrieval of Ordered Lists from a Content Addressed Memory.
RCA Review, 23(1962), 215-229

Lewis HR,Papidimitriou CH
The Efficiency of Algorithms
Scientific American, 238(Jan 1978), 97-109

Licklider JCR
Libraries of the Future.
MIT Press., (1965).

Lin CS,Smith DCP,Smith JM
The Design of Rotating Associative Memory for Relational Database Applications.
ACM Transactions on Database Systems, 1(1976), 53-65

Liu JWS,Jino M
Intelligent Magnetic Bubble Memories and their Applications in Data Base Systems.
IEEE Trans. on Computers, 28(1979), 888-906

Livesey B,Kendall M
Teaching On-Line Information Retrieval and Automated Circulation Control.
Leeds Polytechnic School of Librarianship Report., (1978).

Lockheed INC
DIALOG Information Retrieval Service.
Lockheed Missiles & Space Company Inc.,Palo Alto, California, USA, (1980).

Long PL
Computer Technology - An Update.
ARIST, 11(1976), 211-22

Luhn HP
The Automatic Creation of Literature Abstracts.
IBM Journal of Research and Development, 2(1958), 159-165

McGill MJ,Smith LC,Davidson S,Noreault T
Syracuse Information Retrieval Experiment(SIRE). Design of an On-line Bibliographic Retrieval System.
SIGIR Forum, 11,4(1976), 37-44

McGregor DR,Thomson RG,Dawson WN
High Performance Hardware for Database Systems.
in Systems for Large Databases(Ed. PC Lockemann and EJ Newhold), North-Holland, (1976), 103-116

MacKenzie KF
Application of 64k CCDs.
Procs. of the 15th IEEE Computer Society Intl. Conference, IEEE, (1977), 386-389

MacKinnon RJ,Schuegraf EJ
A Minicomputer as a Front End Processor for an Automated Libray System.
Procs. of the 3rd Open Conference on Information Science, Canadian Association for Inf. Sci., (1975), 236-242

Malinovsky BN,Bojun BV,Kozlov LG
Real-Time Information Processing and Special Purpose Microprocessors.
Euromicro, 1977, North-Holland., (1977), 67-72

Maller VAJ
The Content Addressable File Store - CAFS.
ICL Technical Journal, 1(1979), 265-279

Marcus RS,Reintjes JF
A Translating Computer Interface for End-User Operation of Heterogeneous Retrieval Systems. I. Design.
JASIS, 32(1981), 287-303

Marcus RS,Reintjes JF
A Translating Computer Interface for End-User Operation of Heterogeneous Retrieval Systems. II. Evaluations.
JASIS, 32(1981), 304-317

Marron B,Fife D
On-Line Systems - Techniques and Services.
ARIST, 11(1976), 163-210

Martella G,Gobbi G
A Microprocessor Architecture for Bibliographic Retrieval System.
Information Processing and Management, 17(1981), 239-247

Maryanski FJ
Backend Database Systems.
Computing Surveys, 12(1980), 3-25

Mathews EW,Thompson L
Weighted Term Search: A Computer Program for an Inverted Coordiate Index on Magnetic Tape.
Journal of Chemical Documentation., 7(1967), 49-56

Meadow CT,Toliver DE,Edelmann JV
A Technique for Machine Assistance to Online Searchers.
Procs. of the ASIS Annual Meeting, 15(1978), 222-225

Meilander WC
The Evolution of Parallel Processor Architecture for Image Processing.
Procs. of the 15th IEEE Computer Society Intl. Conference.
IEEE., (1977), 52-57

Melen R,Buss D
Charged Coupled Devices: Technology and Applications.
IEEE Press., (1977),

Merckle RC,Hellman ME
Hiding Information and Signatures in Trapdoor Knapsacks.
IEEE Trans. on Information Theory., 24(1978), 525-530

Middelhoek S,George PK,Dekker P
Physics of Computer Memory Devices.
Academic Press., (1976).

Miller RF,Tighe DL
Library and Information Networks.
ARIST, 9(1974), 173-219

Miller WL
A Probalistic Search Strategy for Medlars.
J.Doc., 27(1971), 254-266

Minker J
An Overview of Associative or Content-Addressable Memory Systems and a KWIC Index to the Literature 1956-1970.
Computing Reviews., 12(1971), 453-504

Mittman B,Dominick WD
Developing Monitoring Techniques for an On-Line Information Retrieval System.
Information Storage and Retrieval, 9(1973), 297-307

Monsen GL
Computer Terminals and Minicomputers in Online Retrieval.
Online Review, 1(1977), 217-229

Morris D,Ibett RN
The MU5 Computer System.
MacMillan Press Ltd., (1979).

Motorola INC
M6800 Application Manual
Motorola Semiconductor Products Inc., Switzerland., (1975a).

Motorola INC
M6800 Programming Manual
Motorola Semiconductor Products Inc., Switzerland, (1975b).

Motorola INC
M6800 Systems Reference and Data Sheets
Motorola Semiconductor Products Inc., Arizona, USA, (1975c).

Motorola INC
MC6809 Preliminary Programming Manual
Motorola Semiconductor Products Inc., Switzerland, (1979a).

Motorola INC
MC6809 Advance Information
Motorola Semiconductor Products Inc., Switzerland, (1979b).

Mueller GE
Data Processing for Truckers - The Real Era Lies Ahead.
System Development Corporation, USA., (1977).

Muhlenfeld E
A Holographic Associative Memory for Information Retrieval.
Intl. Optical Computing Conference, Digest of Papers, Italy 1976, IEEE., (1976), 111-111

Mulrooney T
Microprogramming A Mini-Computer for Fast Signal Processing.
Electronics, 51,6(1978), 136-141

Mulvihill J,Brenner EH
Ranking Boolean Search Output.
American Documentation, 19(1968), 204-205

Myers W
Key Developments in Computer Technology.
Computer, 9,11(1976), 48-75

Myers W
Current Developments in Magnetic Bubble Technology.
Computer, 10,8(1977), 73-82

Nauss RM
An Efficient Algorithm for the 0-1 Knapsack Problem.
Management Science, 23(1976), 27-31

Negus AE
EURONET Guideline: Standard Commands for Retrieval Systems.
INSPEC, London, (1977).

Nemhauser GL,Garfinkel R
Integer Programming.
J.Wiley & Sons Ltd., New York., (1972).

Noreault T,Koll M,McGill MJ
Automatic Ranked Output from Boolean Searches in Sire.
JASIS, 28(1977), 333-339

Oddy RN
Reference Retrieval Based on User Induced Dynamic Clustering.
Ph.D. Thesis, University of Newcastle upon Tyne., (1974).

Oddy RN
Information Retrieval Through Man-Machine Dialogue.
J.Doc., 33(1977), 1-14

Overhage CFJ,Reintjes JF
Project Intrex: a General Review.
Information Storage and Retrieval, 10(1974), 157-188

Paice CD
Information Retrieval and the Computer.
MacDonald and Janes, (1977).

Parhami B
A Highly Parallel Computing System for Information Retrieval.
Procs. of the Fall Joint Computer Conference, AFIPS., 41(1972), 681-690

Parhami B
Associative Memories and Processors - An Overview and Selected Bibliography.
Procs. of the IEEE, 61(1973), 722-730

Pratt AD
The Use of Microcomputers in Libraries.
Journal of Library Automation, 13(1980), 7-17

Preece SE
Clustering as an Output Option.
Procs. of the ASIS Annual Meeting, 10(1973), 189-190

Radhakrishnan T,Rajaraman V
Parallel Processing Computer Architecture for Information Retrieval.
Procs. of the 3rd Open Conference on Information Science, Canadian Assoc. for Inf. Sci., (1975), 56-64

Ramamoorthy CV,Turner JL,Wah BW
Design of a Fast Cellular Associative Memory for Ordered Retrieval.
IEEE Trans. on Computers, 27(1978), 800-815

Richards M
The Portability of the BCPL Compiler.
Software Practice and Experience, 1(1971), 135-146

Richards M
The BCPL Programming Manual.
Computing Laboratory, University of Cambridge, (1974).

Robertson SE
Term Specificity.
(Letter) J.Doc., 28(1972), 164-165

Robertson SE
Specificity and Weighted Retrieval.
J.Doc., 30(1974), 41-46

Robertson SE
A Theoretical Model of the Retrieval Characteristics of Information Retrieval Systems.
Ph.D. Thesis, University of London, (1976).

Robertson SE
Progress in Documentation: Theories and Models in Information Retrieval.
J.Doc., 33(1977), 126-148

Robertson SE
The Methodology of Information Retrieval Experiment.
in Information Retrieval Experiment, Ed. K. Sparck-Jones, publ. Butterworths, (1981).

Robertson SE,Sparck-Jones K
Relevance Weighting of Search Terms.
JASIS, 26(1976), 129-146

Robinson WL
Britain's Packet Switching Experiment.
Computer Decisions, 8,2(1976), 26-28

Rouse WB,Rouse SH
Assessing the Impact of Computer Technology on the Performance of Interlibrary Loan Networks.
JASIS, 27(1977), 79-88

Rush JE
Development and Operation of a Database Machine for On-line Access and Update of a Large Database.
On-line Review, 4(1980), 237-261

Sahni S
On the Knapsack and other computationally Related Problems.
Ph.D. Dissertation, Cornell University, N.Y.,USA., (1973).

Sahni S
Approximate Algorithms for the 0/1 Knapsack Problem
JACM, 22(1975), 115-124

Sahni S,Horowitz E
Combinatorial Problems - Reducibility and Approximation
Operational Research, 26(1978), 718-759

Salton G
Automatic Information Organisation and Retrieval.
Mc.Graw Hill, NY, USA., (1968).

Salton G
Evaluation Problems in Interactive Information Retrieval.
Information Storage and Retrieval, 6(1970), 29-44

Salton G
The SMART Retrieval System - Experiments in Automatic Document Processing
Prentice Hall, Englewood Cliffs NJ,USA., (1971).

Salton G
Dynamic Document Processing
CACM, 15(1972), 658-668

Salton G
Dynamic Information and Library Processing
Prentice Hall, Englewood Cliffs, New Jersey, USA, (1975).

Salton G
Automatic Information Retrieval:
Computer, 13,9(1980), 41-55

Salton G,Waldstein RK
Term Relevance Weights in On-Line Information Retrieval.
Information Processing and Management, 14(1978), 29-35

Salton G,Wong A,Yu CT
Automatic Indexing Using Term Discrimination and Term Precision Measurements.
Information Processing and Management, 12(1976), 43-51

Salton G,Yang CS
On the Specification of Term Values in Automatic Indexing
J.Doc., 29(1973), 351-372

Schultz L
Breaking the Communications Barrier between Searcher and Literature File: An Interactive Guide.
JASIS, 25(1974), 3-9

Schuster SA,Nguyen HB,Ozkarahan EA,Smith KC
RAP.2 - An Associative Processor for Databases and its Applications.
IEEE Trans. on Computers, 28(1979), 446-458

Seeber RR,Lindquist AB
Associative Memory with Ordered Retrieval.
IBM Journal of Research and Development, 6(1962), 126-136

Shapiro SD
Performance of Heuristic Bin Packing Algorithms with Segments of Random Length.
Information and Control, 35(1977), 146-158

Sharpless GT,Penna DE,Turner SR
An Advanced Home Terminal for Interactive Data Communication.
Procs. of the Intl. Conference on Communications Part 2.
IEEE., (1977), 47-50

Slotnick DL
Logic Per Track Devices.
Advances in Computers, 10(1970), 291-296

Smith DCP,Smith JM
Relational Data Base Machines.
Computer, 12.3(1979), 28-38

Souster J
Data Communications, Intelligent Terminals and Word Processors.
M.Sc. Dissertation, Sheffield Postgraduate School of Library and Inf Sci, (1977).

Sparck-Jones K
A Statistical Interpretation of Term Specificity and its Application in Retrieval.
J.Doc., 28(1972), 11-21

Sparck-Jones K
Index Term Weighting.
Information Storage and Retrieval, 9(1973a), 619-633

Sparck-Jones K
Index Term Frequency and Quartile Deletion.
JASIS, 24(1973b), 166-167

Sparck-Jones K
Experiments in Relevance Weighting of Search Terms.
Information Processing and Management, 15(1979), 133-144

Sparck-Jones K
Search Term Relevance Weighting Given Little Relevance Information.
J.Doc., 35(1979), 30-48

Sparck-Jones K
Search Term Relevance Weighting - Some Recent Results.
Journal of Information Science, 1(1980), 325-332

Sparck-Jones K,Webster CA
Research on Relevance Weighting 1976-1979.
British Library Research and Development Report No. 5553, (1980).

Stellhorn WH
A Specialised Computer for Information Retrieval.
Ph.D. Thesis, University of Illinois, USA., (1974).

Stellhorn WH
An Inverted File Processor for Information Retrieval.
IEEE Trans. on Computers, 26(1977), 1258-1267

Stiefel ML
What is an Intelligent Terminal?
Mini-Micro Systems, (Mar 1977), 50-59

Su SYW
Cellular Logic Devices: Concepts and Applications.
Computer, 12,3(1979), 11-25

Su SYW,Nguyen LH,Emam A,Lipovski GJ
The Architectural Features and Implementation Techniques of the Multicell CASSM.
IEEE Trans. on Computers, 28(1979), 430-445

Svenonius E
An Experiment in Index Term Weighting.
JASIS, 23(1972), 109-121

Tarjan RE
Complexity of Combinatorial Algorithms
SIAM Review, 20(1978), 457-491

Teskey FN
STATUS and Integrated Information Systems.
J.Doc, 36(1980), 33-39

Thompson DA
Interface Design for an Interactive Information Retrieval System: A Literature survey and a Research System Description.
JASIS, 22(1971), 361-373

Thornton JE
Back-End Network Approaches.
Computer, 13,2(1980), 10-17

Thurber KJ,Berge RO
Applications of Associative Processors.
Computer Design, 10(Nov 1971), 103-110

Thurber KJ,Wald LD
Associative and Parallel Processors.
Computing Surveys, 7(1975), 215-255

Ungerer H
EURONET: the EEC On-Line Information Network.
UNESCO Bulletin for Libraries, 31(1977), 128-133

VanDam A,Rice DE
Computers and Publishing, Writing, Editing and Printing.
Advances in Computers, 10(1970), 145-174

VanRijsbergen CJ
A Theoretical Basis for the use of Co-occurrence Data in Information Retrieval.
J.Doc., 33(1977), 106-119

VanRijsbergen CJ
Information Retrieval (Second Edition)
Butterworths, London, (1979).

VanRijsbergen CJ,Robertson SE,Porter MF
New Models in Probalistic Information Retrieval.
British Library Research and Development Report No.5587, (1980).

Vemuri V,Liuzzi RA,Cavano JP,Berra PB
Evaluation of Alternate Data Base Machine Designs.
SIGIR Forum, 15,2(1980), 29-35

Verhoeff J,Goffman W,Belzer J
Inefficiency of the Use of Boolean Functions for Information Retrieval Systems.
CACM, 4(1961), 557-558, 594

Vernimb C
Automatic Query Adjustment in Document Retrieval.
Information Processing and Management, 13(1977), 339-353

Vick CR,Kartashev SP,Kartashev SI
Adaptable Architectures for Supersystems.
Computer, 13,11(1980), 17-35

Volk JL
Interactive Television Experiment in Reston, Virginia.
Journal of Educational Technology Systems, 3(1974), 73-84

Wagner RA,Fischer MJ
The String-to-String Correction Problem.
JACM, 21(1974), 168-173

Ware WH
The Ultimate Computer.
IEEE Spectrum, 9,3(1972), 84-91

Watkins DCJ
Distributed Processing - A Logical Approach to Real Time, On-line Systems.
Intl. Conf. on Distributed Computer Control, Birmingham, England., (1977), 13-20

Watson RW
Network Architecture Design for Back-End Storage Networks.
Computer, 13,2(1980), 32-48

Weide B
A Survey of Analysis Techniques for Discrete Algorithms.
Computing Surveys, 9(1977), 291-313

Wesley MA
Associative Parallel Processing for the Fast Fourier Transform.
IEEE Trans. on Audio and Electroacoustics, 17(1969), 162-165

Williams JH
Functions of a Man-Machine Interactive Information Retrieval System.
JASIS, 22(1971), 311-317

Williams ME
Data Bases - A History of Developments and Trends from 1966 through 1975.
JASIS, 28(1977), 71-78

Williams ME,Preece SE
Database Selector for Network Use: a Feasibility Study.
Procs. ASIS Annual Meeting, 14(1977), 34-35

Williams PW
Intelligent Access to Online Systems.
Procs. of the 4th International Online Meeting, London, 1980, Learned Information., (1980), 397-407

Wilner WT
Design of the Borroughs B1700.
Procs. of the Fall Joint Computer Conference, AFIPS., 41(1972), 489-497

Yu CT,Salton G
Precision Weighting - An Effective Automatic Indexing Method.
JACM, 23(1976), 76-88

Zilog INC
Z80-CPU Product Specification
Zilog Inc., California, USA, (Mar 1978),

Zoltners AA
Direct Descent Binary Knapsack Algorithm.
JACM, 25(1978), 304-311