# ASSOCIATIVE/PARALLEL PROCESSORS FOR SEARCHING VERY LARGE TEXTUAL DATA BASES

R. M. Bird
J. C. Tu
R. M. Worthy

Operating Systems, Inc.
Woodland Hills, CA

## ABSTRACT

This paper describes an approach to solving a major problem in the information processing sciences--that of searching very large (5-50 billion characters) data bases of unstructured free-text for random queries within a reasonable time and at an affordable price.

The need by information specialists and knowledge workers for large, fast low-cost text and document retrieval systems is growing rapidly. Conventional approaches to the problem have usually depended upon expensive, general purpose computers, upon special pre-preprocessing of the textual data (e.g. file inverting, indexing, abstracting, etc.), and upon elaborate, costly software. The resulting retrieval systems often cost hundreds of dollars per query and the full scanning of an uninverted, unstructured billion byte textual data base could take hours of computer services. However, in spite of these restrictions, such full text search systems have proved useful and even indispensible for many applications.

Computer technology of the late 1960's and the 1970's, in both hardware and software (e.g., minicomputers, low-cost, high density disk storage, "chip" electronics, natural language query systems, etc.), have made it practical to build special purpose, low-cost text retrieval systems. Such a system has been built, tested, and is now in a production stage. The system called the Associative File Processor (AFP), utilizes a conventional minicomputer (DEC's PDP-11/45) for control, off-the-shelf high density disks for storage, a special purpose parallel search module as a text term detector, and query and retrieval software. The AFP is currently being field tested at two sites. Full text, parallel searches on un-preprocessed textual data bases are being performed at the effective matching rates of 4 billion bytes per second (8K byte key memory times 500 Kbyte/second data stream). Estimated costs are 10 to 25 cents per query for a one billion byte data base. The costs per query and the time for searching increase in a linear fashion as data base increases. A basic architecture for the AFP is described and an implemented version is discussed. A more powerful term detector module is also under development. This system is designed around a finite state automaton algorithm.

## INTRODUCTION

The utility of free-text search and retrieval has long been recognized. The accumulation of textual records, documents, reports, contracts, briefs, studies, etc., is vast and ubiquitous throughout government business, industry, science, engineering, educational institutions, and the military. It has been estimated that the U.S. Federal Government alone maintains over 20,000 machine searchable data bases. Many of these data bases are composed totally of textual information; almost all have some textual files. New requirements, new applications, and new text retrieval systems appear daily. It is not unreasonable to assume that within the next decade language processing, text searching, and fact retrieval will become major disciplines in the information computer sciences. Today, however, in spite of the increasing need for, and the use of, text retrieval systems, the cost of creating, processing, and maintaining large textual data bases is growing. In other computer applications (business, engineering, process control) the cost per transaction, or for almost any computational work unit, has been continously decreasing during the last decade, even in the face of general economic inflation.

This cost anomaly between language processing and other computer processing applications is due in part to the reliance by designers on conventional, general purpose hardware and software systems. These "EDP" components and approaches are simply not cost-effective when applied to large language and textual data bases. Other special information processing disciplines have solved similar problems by building special purpose hardware and

special purpose programming languages, e.g., FFT and array processors, hybrid computers; text and string programming languages. This paper suggests that language processing and text retrieval systems will require special hardware architectures as well as innovations in computational linguistics, user interfaces, and in software/firmware techniques. A first step towards a family of special purpose processors for searching very large (5-50 billion bytes) textual data bases is described.

## CURRENT TEXT RETRIEVAL APPROACHES

### A Textual Data Base Problem

A small law library might consist of 1000 volumes of legal reports, briefs, cases, decisions, and legislative acts. Assuming that each volume contained 1,000,000 characters, then if the 1000 volumes were digitized, a raw textual file of a billion ($10^9$) characters would be available for machine processing. Unless special indicators had been inserted into the text to delimite document boundaries or other textual divisions, the raw data base could be viewed as a string of alphanumeric characters a billion bytes long. Except for statistical or linguistic processing, a billion byte string is an impossible structure for retrieval purposes.

The first step then is to edit or preprocess the textual data to make it more accessible for search and retrieval functions. The amount of editing and preprocessing will turn out to be a major factor in determining the search and retrieval effectiveness and the total costs of the entire process. Major editing and restructuring of the basic text can help to speed up search and retrieval times, but often at the cost of losing content. For example, many retrieval pre-editing systems will remove all high frequency words such as pronouns, prepositions, and common adverbs. This will greatly compress the text but it also removes most syntactic relationships among the semantic bearing words.

In our digitized billion character legal file, if preprocessing consists only of inserting special characters at the end of each article or document, then programs could be written that would search each discrete piece of text for a query (say, key words and phrases), and that would retrieve the document if the query were satisfied. Without any further preprocessing of the text, the search would have to be carried out in a serial fashion. That is, each word in every document in the data base would be examined in turn to see if it met the search criteria. This technique would require a very long search time, but it would be the least destructive to the original structure of the text. This technique is called full-text scanning of unstructured text. Full text search of an unpreprocessed file may take several hours of CPU time and cost several hundred dollars per query for a multi-billion character file. Some applications require such text searching and many users are willing to pay the relatively high price. This basic approach is illustrated by Figure 1.

The system shown in Figure 1 minimizes preprocessing and prestructuring and maximizes context availability, at the expense of long search times. The system is also relatively simple and inexpensive to implement.

In most commercially available text retrieval systems, such as ORBIT[1], DIALOG[2], and Data Central[3], the emphasis is on search and retrieval time. Consequently, greater resources are devoted to "indexing" or preprocessing the textual data in order to produce a more efficient search mechanism for multi-users for on-line retrieval. The original text is seldom searched and some content and most structure is lost. In these systems, a representation of the documents are searched, not the basic text itself. This is illustrated in Figure 2.

In the structured text searching approach, greater resources are placed on preprocessing the text and creating an efficient search subsystem. The search surrogate of the text is related to the original text (and the retrieval file) by means of a pointer or mapping device. Whether the search surrogate is manually produced (human generated indexes) or

[1]ORBIT is a proprietary system of System Development Corp.

[2]DIALOG is a proprietary system developed by the Lockheed Palo Alto Research Laboratory.

[3]Data Central is a commercial system developed by Mead Data Central, Inc.

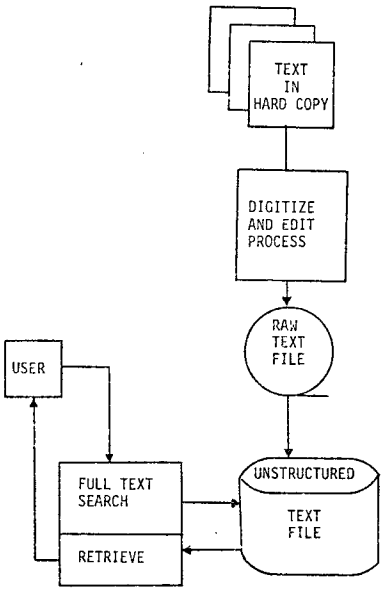


Figure 1. Full Text, Unstructured File Sea

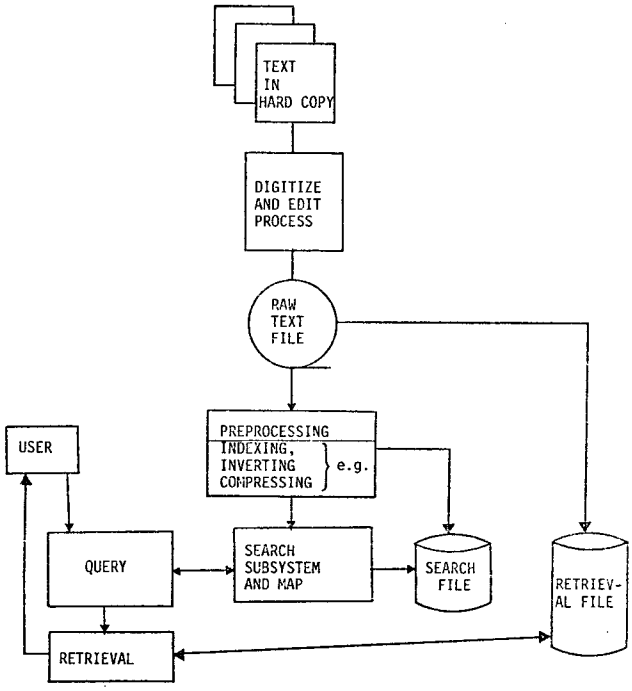


Figure 2. Structured Text Search

machine created (such as file inversion), a large and continuous cost is associated with creating the search surrogate subsystem. In rapidly changing and/or rapidly expanding textual data bases, the cost to prepare the search surrogate will eventually render the system virtually unaffordable.

For example, in utilizing the inverted file technique for producing the search surrogate, the indexes and their overhead can grow to many times larger than the original text file. In our example of the billion character legal data base, an inverted file approach that preserved most of the text could produce an index file of 2-3 billion characters in addition to the original text file.

Naturally the cost of computer services and programming increases as the data bases become larger. Currently the costs indexed and inverted systems are tolerated because the data bases are relatively small and because the alternative, full-text, unstructured approach is even more time consuming and costly. All approaches to text search and retrieval can be characterized by one or a combination of these two techniques. For textual data bases in the order of 5-50 billion characters, no current conventional approach seems reasonable.

## A Special Purpose Solution

It has become obvious to many system designers working in the language processing and text retrieval disciplines that new hardware devices are needed; continued reliance on software ingenuity will probably not produce a spectacular breakthrough.

Dramatic reductions in storage costs and increases in transfer speeds for magnetic disk systems have occurred in the last five years. Optical and other inexpensive bulk storage devices are on the horizon. Advanced in minicomputers, microprocessors, and electronics hardware in general have changed the information processing industry. However, the basic problem for text processing remains the same; no general purpose, serial CPU is fast enough to handle the processing required for searching billions of characters of text.

The solution that is proposed here is to build a special parallel processor, especially designed to handle textual data. This device will be relatively simple because one major text processing function has been identified as character matching. If a special, parallel character matching device is used in conjunction with conventional bulk storage and conventional CPU's, then the first step toward a complete text processor will have been taken. A somewhat similar system was developed by General Electric and called GESCAN or the Rapid Search Machine (RSM). RSM utilizes magnetic tape and has limited query capability.

The architecture for, and the implementation of, a special purpose text processor in its initial field tests indicate that hardware for text processing is cost-effective, extendable, and flexible. In this system, a billion byte, full test, data base with no preprocessing (no inverting or indexing) can be completely searched for 30-50 simultaneous natural English-like queries in 4-5 minutes, at a cost of a few cents per query.

## SPECIAL PURPOSE HARDWARE SOLUTIONS

### A General Associative File Processor Architecture

There are many types of computer architectures that could be configured to solve the problem of searching very large textual data bases without requiring elaborate preprocessing and resorting to file inversion.

A general Associative File Processor architecture is herein defined which meets specifications given below:

- File size up to $10^{9-12}$ bytes.
- Implemented with low cost conventional bulk memory.
- Hardware logic utilizes available off-the-shelf, conventional components.
- Utilizing standard minicomputers for control.
- Allows multiple users and simultaneous queries.
- English-like analyst-oriented query language.
- Unrestricted, variable length query terms with Boolean and proximity logic.
- Unstructured textual data into physical storage.
- Costs of systems are increased linearly with data base size.

The fundamental characteristics of this family of architecture are: (1) the use of conventional bulk memories to hold the data base; (2) batched queries allowing multiple users simultaneous access; and (3) the use of parallel and pipeline processing to accept data from the bulk memories and process it on the fly simultaneously against multiple queries.

The three principal functional elements of this architecture are shown in Figure 3. These are: (1) query translation — the translation of the multiple queries in a batch into (a) key words to be loaded into the term detectors; and (b) translation of the logical key word connectives and the appropriate user information to be loaded into the query resolver; (2) the associative searcher which accepts textual data from the bulk memory and processes it on the fly simultaneously against multiple key words, reporting hits and hit word position to the query resolver when they occur; and (3) the query resolver which performs the appropriate logical connectives between the detected key words and determines a matched query for a document when it occurs, recording the document identification with the appropriate user ID for the eventual document retrieval.

A more detailed functional diagram of the Associative File Processor architecture in which some of the hardware elements are shown is described in Figure 4. The users input/output devices are the CRT terminals as shown. In the configuration shown, the Associative File Processor system is totally self-contained with the host processor servicing the user terminals. In another configuration, which is not shown, the host processor could interface through a high speed communication line to a large main frame computer which would serve the users terminals. In this latter configuration, the Associative File Processor system is treated as a peripheral search subsystem to an already existing large main frame data processing system. Either configuration is practical.

In either of the above described configurations, the host processor shown in Figure 4 performs the query translation function, the query output response function, and the data base update function. In the

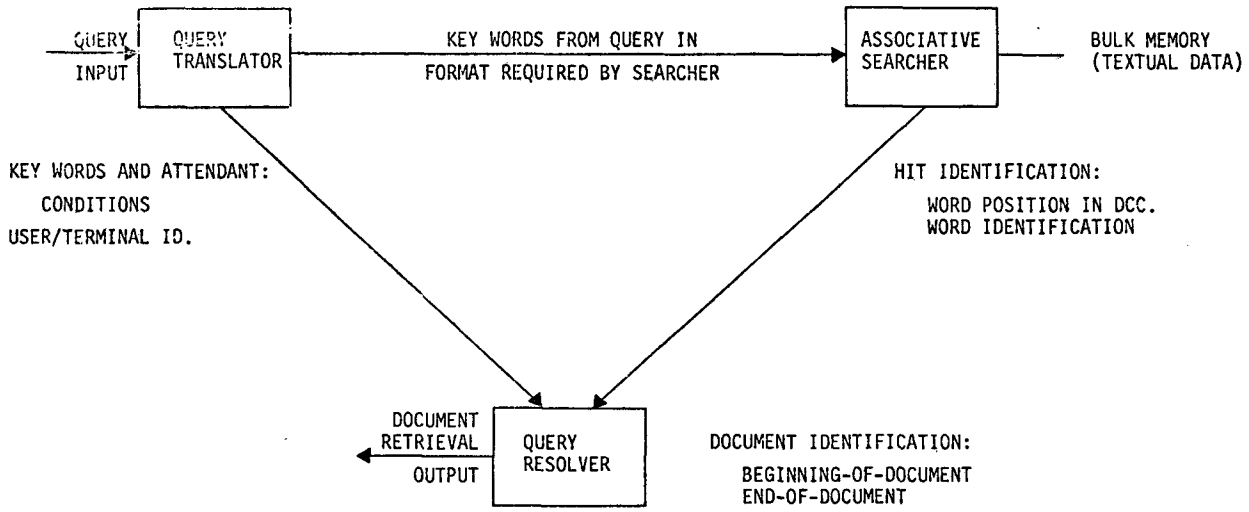


Figure 3. The Three Principal Functional Elements of The Associative File Processor

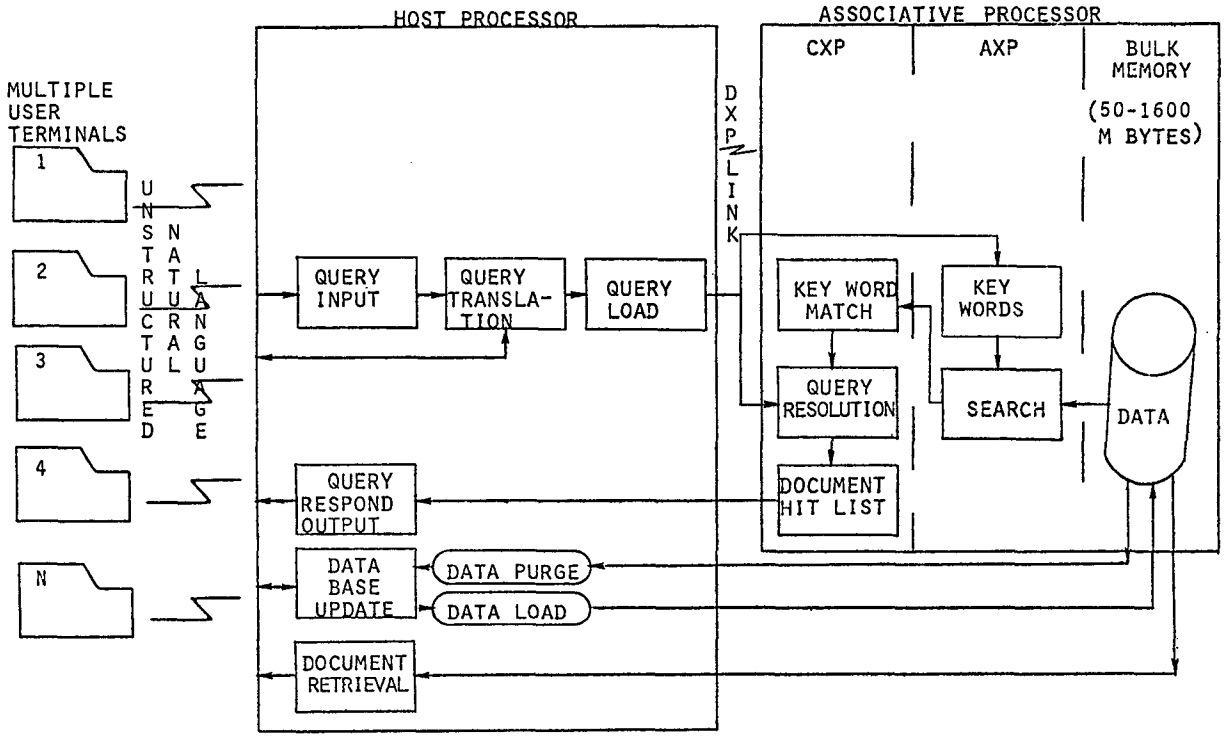


Figure 4. ASSOCIATIVE FILE PROCESSOR FUNCTIONAL BLOCK DIAGRAM

stand-alone configuration, the document retrieval function would also be performed by the host processor. In the large main frame configuration, the document retrieval function would be performed by the large computer.

The Associative Processor shown in Figure 4 is connected to the host processor through a data link (shown as a DXP link). A single associative processor is shown which requires one DXP data link to the host processor. It is possible to have many associative processors and many data link connections to the host processor.

Each associative processor contains a bulk memory device with a dedicated controller, a special purpose text matcher called an associative crosspoint processor (AXP), and a computer crosspoint processor (CXP). The key words are loaded into the AXP; the query logic and user IDs are loaded into the CXP. The data from the bulk memory channel are intercepted by the AXP and data words are matched against key words. As key word matches occur, this information is passed to the CXP where query resolution is performed. When a document matches a query requirement, this hit information is passed back to the host processor and then to the user. The user can request document retrieval at that time. If desired, the system will also display found documents while the search is being performed.

The host processor must also perform a data base update function to load new data into the bulk memory and to purge old data from the bulk memory. This function requires minimal software and data base management requirements, since new data may be loaded into any available space in the bulk memory without keeping a file of where it is loaded. Data may be purged in a similar manner. This is one of the significant advantages of this associative architecture; i.e., minimal data base management requirements, and no production of a surrogate search subsystem. The entire cost of text preprocessing and file inverting is eliminated. This approach is effectively a hardware version of the full-text, unstructured search strategy.

An Implementation of an Associative File Processor

Operating Systems, Inc. has been developing hardware for an Associative File Processor since the spring of 1974. Three prototype units has been built and tested over the last year and a half, demonstrating a completely stable hardware configuration and design. The AFP is now in production and is available.

This product uses a PDP-11 series computer as the host processor with a single data channel from the bulk memory. A block diagram is shown in Figure 5. Because of the single data channel, the PDP-11 also serves as the crosspoint processor (CXP). The AXP unit has 8192 bytes for key word storage which will hold about 1200 English key words. This gives the AFP the capability to batch about 40 to 70 complex queries for the simultaneous search. The bulk memory is implemented by a 200 megabyte disk storage system which requires between four and five minutes for a complete search. The user is normally occupied during the search interim period since the first query matching document is displayed on the CRT terminals at the time the match occurs.

A more detailed version of the single channel AFP showing the bus structure is contained in Figure 6. A bus switch is placed between the CPU side of the bus and the disk controller side of the bus. When in normal mode without the AXP in operation, the two buses are connected and the DPU can communicate with the disk controller. Just before a search cycle, the CPU sets the controller up for data transfer and then commands the bus switch to the search mode. The AXP can then accept data from the controller with the CPU being disconnected and free from the bus activity on the controller side of the bus. The CPU can then, at the same time, act as the CXP performing the query resolution function.

A functional block diagram of the AXP unit is shown in Figure 7. There are four register interfaces within the AXP unit that are attached to the bus. These are: (1) the bit interface; (2) the bus switch interface; (3) the load interface and (4) the search interface. All communications and data transfer into and out of the AXP are through these interfaces.

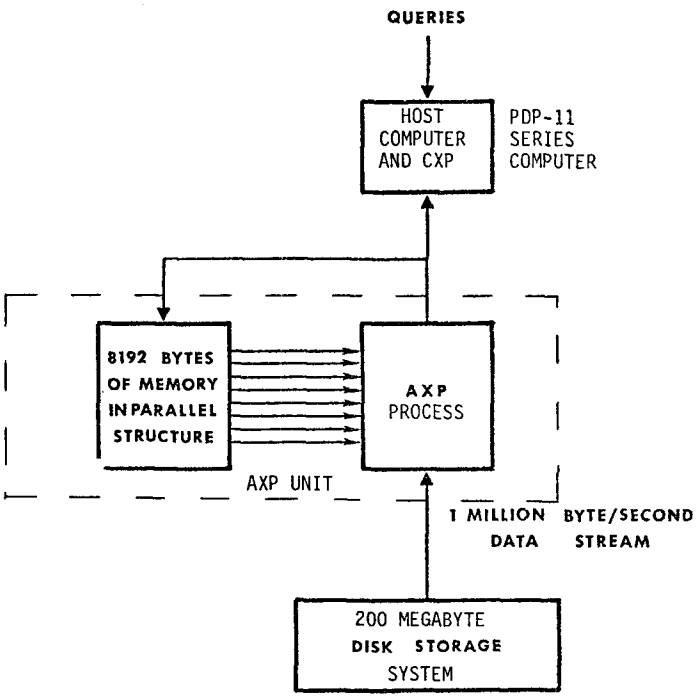


Figure 5. Single Data Channel AFP

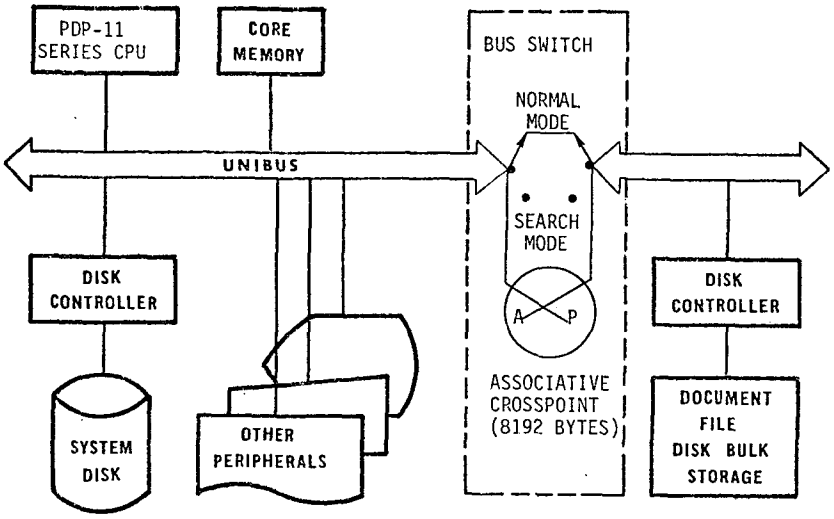


Figure 6. Single Channel AFP Bus Structure

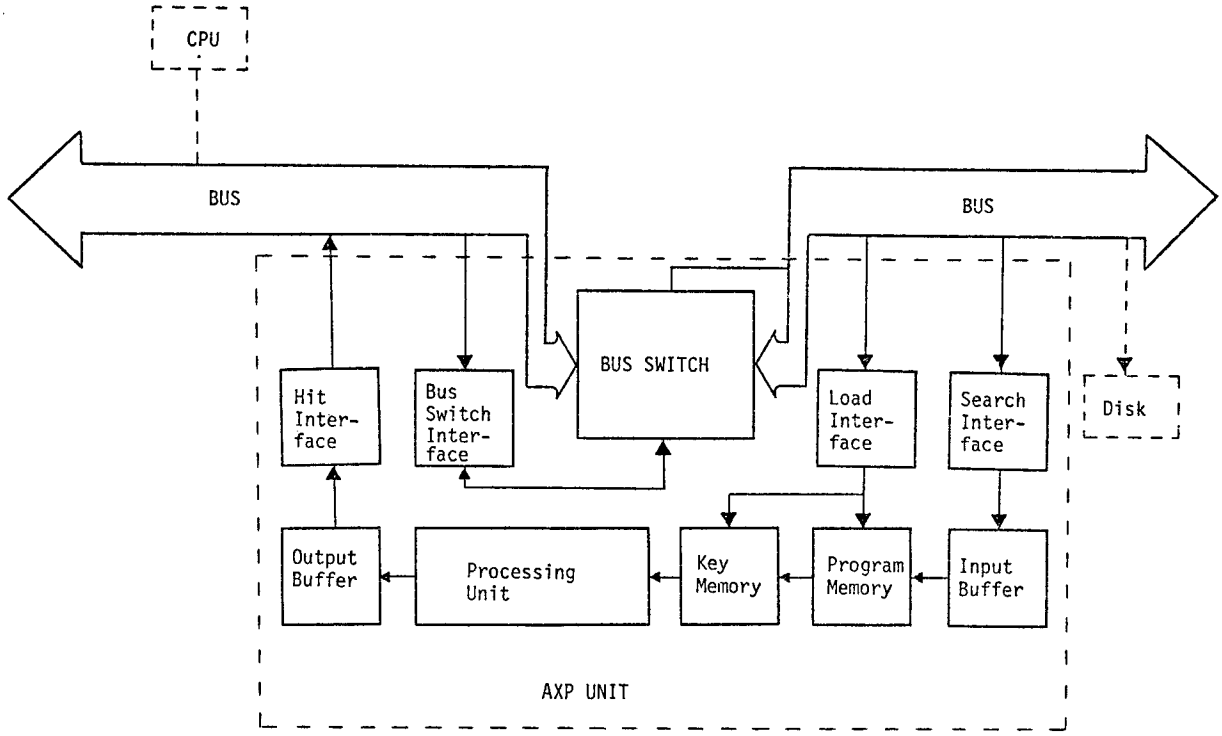


Figure 7. AXP Functional Block Diagram

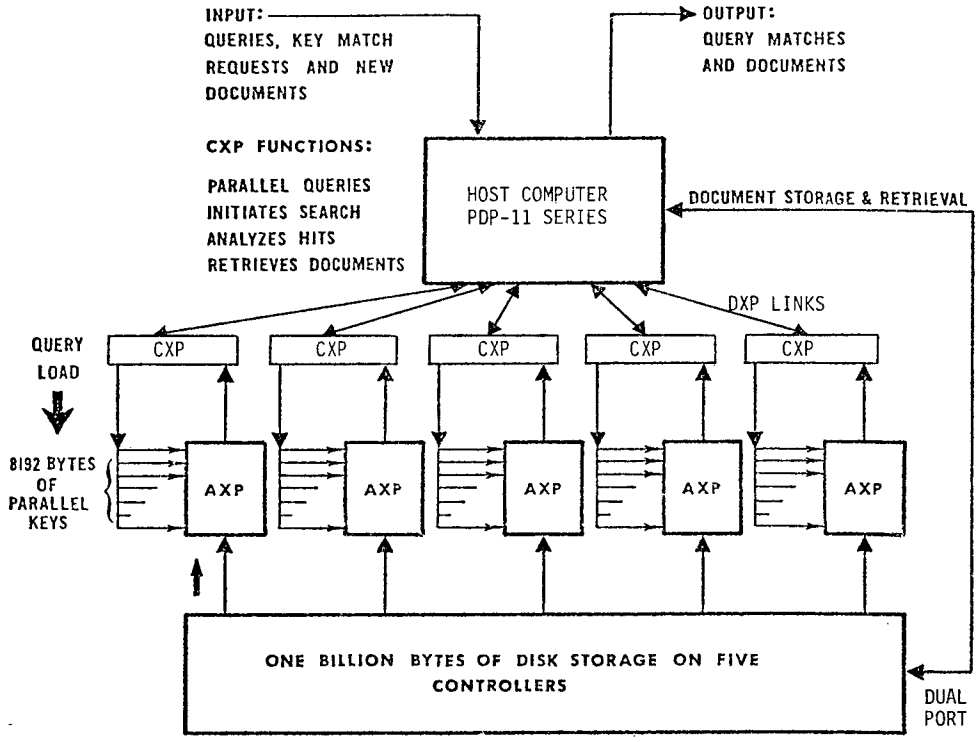


Figure 8. Parallel Channel AFP

There is an input and output buffer that function asynchronously to the internal processing within the AXP. There is a key memory which holds the key words and a micro program memory which holds a memory map for mapping search input words into the key memory. There is a processing unit which makes logical comparisons between the characters of a key memory word and the input search word, and determines a matched condition for outputting through the bit interface. These are the principal functions performed by the AXP unit.

The single channel AFP can be expanded to a parallel channel channel AFP in modular form by incorporating a CXP at each AXP data channel level. Such a configuration is shown in Figure 8 employing five AXP channels providing a one-million byte storage capability. This data base can still be associatively searched within the four to five minute period for the multiple queries, or the same as for the single data channel configuration. Thus, the system is modular and can be expanded to billions of bytes of data storage, where the search time remains constant.

The parallel data channel AFP is presently in development and is planned for completion before the end of 1977. The CXP is implemented by a PDP-11/04. A dual port feature for document storage and retrieval is optional. Data would be stored and retrieved through the CXP channels if the dual port feature is not made available.

The cost of the single channel AFP as an add-on to an existing PDP-11 series computer is about $100,000. The cost of the one billion byte parallel channel AFP including the host PDP-11 computer and bulk memory is about $600,000. With a reasonable utilization, the cost per query for these AFP systems is in the order of a few cents per query. We conclude that these AFP configurations are a cost-effective solution to the free text search problem for many applications.

## FUTURE SYSTEMS AND CONCLUSIONS

### Future Systems

Operating Systems, Inc. has under development an extended AFP system in the same family are previously described, but with several additional features. This AFP system will be available for field testing in about two years. It is being designed to handle as many as 200 users simultaneously with a four-to-five minute query batch processing time. It will have additional hardware features in the term detection unit, including a variable length don't-care key word matching capability, a contiguous word phase capability, and special document boundary processing capability for delineating document zones. Sentences and paragraph boundaries as part of the query formulation. It also will have the query resolution requirement built into special microprocessor logic for high speed processing. The new AFP will produce key word statistics as a result of a search to assist the user in making a better query for a possible next batch run. Finally, the system will allow document lists to be specified as a part of the query so that the search will be limited to only that document list.

We believe that this general AFP architecture has a prominent place now and in the future for querying very large textual data bases. Furthermore, we believe that this architecture will enhance and make feasible a next higher level processing capability for relational data bases and for fact retrieval systems.