2
mix

1-5.00

nformation and Management Sciences

(NASA-CR-129817) DATA STORAGE TECHNOLOGY: HARDWARE AND SOFTWARE, APPENDIX B J.D. Sable (Auerbach Associates, Inc., Arlington, Va.) 24 Aug. 1972 60 p CSCL 09B G3/08

N73-14190

Unclas 16410

AUERBACH ®

60 pp.

DATA STORAGE TECHNOLOGY — HARDWARE AND SOFTWARE

APPENDIX B TO

TECHNICAL REPORT
1958-100-TR-004

By:

JEROME D. SABLE

Submitted to:

NASA Headquarters
Under
Contract No. NASW-2285

August 24, 1972

AUERBACH

AUERBACH Associates, Inc.
1501 Wilson Boulevard
Arlington, Virginia
22209

TABLE OF CONTENTS

AUERBACH

SECTION 1. SUMMARY

1.1 PURPOSE OF THIS DOCUMENT

This document describes a project effort, and can be used by NASA to evaluate the worth and feasibility of launching that project effort. The subject project involves developing more economical ways of integrating and interfacing new storage devices and data processing programs into a computer system. It involves developing interface standards and a software/hardware architecture which will make it possible to develop machine independent devices and programs. These will interface with the machine dependent operating systems of particular computers. The development project will <u>not</u> be to develop the software which would ordinarily be the responsibility of the manufacturer to supply but to develop the standards with which that software is expected to conform in providing an interface with the user or storage system.

1.2 PURPOSE OF THE DEVELOPMENT PROJECT

The purpose of this project is to specify the components of a storage management system employing devices of advanced performance and massive capacity. The intent is to permit the development of a NASA standard system, which avoids the cost of developing individual mass data storage systems for each installation and eliminates differences in the procedures for automatic access to them. The system components, and the related development to be undertaken by the project are:

- Storage Devices. A hierarchy of storage equipment, ranging across the speed/capacity/cost spectrum, and consisting of items of standard manufacture will be used. Performance specifications for the devices required will be developed by this project.
- Storage Management Processes. This comprises the processes necessary to the logistics of moving data automatically between slower- and faster-access storage, which may be executed by hardware dedicated to the purpose or shared by other processes. The project will develop effective strategies, and a system architecture, for carrying out these storage management functions.
- Data Management Processes. These consist of programs to provide data management services, such as establishing a file, reading into it, writing from it, etc. The project will specify a set of data types suitable for general NASA use and a Data Management System architecture which supplies a full range of necessary services for these types.
- User Language. The project will specify a language for invoking the services of the Data Management System.

The project will also develop a storage system simulator, which will be used to determine the performance of a given mix of devices in a storage hierarchy and/or a given logistics strategy employed by the storage manager.

1.3 PRINCIPLE BENEFITS AND BENEFICIARIES OF THE RESULTS

The obvious areas to benefit from the use of a systematically specified set of storage devices are NASA's ground-based computing facilities, particularly those requiring access to large, on line data bases. These results of a uniform set of performance specifications are likely:

- A set of storage devices which more closely meets the needs of NASA is apt to become available.
- The Data Management System's paging function, which provides for automatic data movement across storage levels, will be more effectively designed, implemented and debugged.
- The way the Data Management System interfaces with the storage device controller will be designed once and will not have to be modified for each device or installation.

The use of the same data management system at several mass storage installations means that the stored data bases will be accessible in identical ways. Thus computer programs used to interact with data stored at any one of the installations will work equally well with all, aside from substantive differences in the data itself. This greatly simplifies program development for a single consumer of data which is stored at several of the sites, all using the standard Data Management System.

The availability of a standardized interface which provides data services at a number of levels, appropriate to a number of user types can have a profound effect on the cost of new program development for NASA. It is almost trite to say that the problem of data logistics is one of the most dominant problems in computing today. Every program, whether application program, compiler, or query interpreter, requires these services, hence every programmer must solve a data logistics problem. If these services are provided in a centralized, standardized, way in the computing facility's operating system then a large proportion of the cost of program design, implementation, debugging, and maintenance can be avoided.

There are certain difficulties inherent in undertaking this project and, in fact, procuring systems which satisfy these goals.

(1) NASA is a small part of the customer base of any one vendor and hence may not be able to sufficiently influence the design of the computer manufacturers' operating systems or storage devices.

(2) Manufacturers may be well into the next generation of operating systems design by the time these standards are completed, so that they may be committed to an incompatible course.

On the other hand, the arguments in favor of developing these standards at the present time are as follows:

(1) Computer manufacturers and independent software firms will be developing data management systems in the near future. There will be continued development of new computers and operating systems by manufacturers. Timely guidance by NASA may very well affect the course that this development may take.

(2) There are no existing interface standards in these areas and the need is recognized as critical.

(3) CODASYL's Data Base Task Group (DBTG) has proposed a standard Data Management System language, but many professionals and manufacturers feel that the DBTG specifications are not adequate for use as the common approach. They are not sufficiently complete, and what has been proposed has deficiencies. The shortcomings are with respect to data independence, data integrity, and compatibility. A fresh look by NASA, from a broader archetectural base, may contribute to an improved specification, satisfactory to CODASYL, manufacturers, and users.

1.4 SCOPE

The project comprises eight developments, divided equally between Storage Management System and Data Management System Tasks.

(1) Defining a Storage Management System Architecture. The functional specifications of a Storage Management System are

to be defined, along with at least one strategy for realization and use.

(2) Defining an Optimal Set of Storage Device Characteristics. The performance and control specifications for an optimal set of storage devices suitable for NASA-wide procurement are developed. In order to match these characteristics to NASA needs, statistics will be gathered and evaluated on representative data storage requirements and flows.

(3) Storage System Simulator. A system flow model and simulation program will be developed to estimate the performance of a given Storage Management System under various problem mixes. The model will permit a hierarchy of devices optimal for typical job mixes.

(4) Feasibility Studies and Analysis. Studies to evaluate the technical feasibility and probable industry/user acceptance of proposed device and interface standards are conducted.

(5) Specification of a Set of Data Types. A set of data types required by the range of system users and their applications is developed. These types range from the machine-oriented structures used by systems programmers to the application-oriented structures used by analysts.

(6) Development of a DMS Language. The data services, user languages, and the DMS/program interface concept are developed in this task. The language must have comprehensive data description and manipulation capabilities in order to facilitate program transferability.

(7) Development of a DMS Architecture. The architecture of the DMS is viewed as a set of virtual machine types which provide standard data management services to the program. Each of these types makes data available to the user at a logical and control level appropriate to his needs. In the larger computers, microprogramming can be used to implement the set of virtual machines.

(8) Feasibility Studies and Analysis. This task evaluates the realizability and performance of DMS designs, the effect of a standard design on DMS technology, its relationship with other proposed standards, etc.

1.5 ORGANIZATION OF THIS APPENDIX

1.5.1 Section 2. Background

In the background section, the major considerations arguing for the development of NASA standard mass storage hardware and software are discussed.

1.5.2 Section 3. Technical Dissertation on the Management of Storage and Data

This section discusses technical subjects germane to this project: types of users and levels of data control; storage management concepts; advanced storage device capabilities; and data management system architecture.

1.5.3 Section 4. Statement of the Problem and Requirements

In Section 4 the tasks of the development are discussed.

1.5.4 Section 5. Discussion of the Project

Section 5 sets forth possible approaches to the storage management and data management tasks.

SECTION 2. BACKGROUND

2.1 THE PROBLEM OF UTILIZING MASS STORAGE AND MANAGING DATA

The optimal design of a mass storage system, its efficient utilization during computation, the providing of useful data services to the programmer, and the effective handling of data logistics during program operations are some of the most important, and least satisfactorily answered, questions facing large-scale computer users at this time. These are the issues of data management, using the phrase in its broadest sense, and because they are related they deserve comprehensive analysis as a set of related issues. A concerted attack on these problems may yield the most powerful solutions and such an attack, in fact, is planned in the study proposed in this Appendix. For convenience, the study is divided into two facets, storage management, and data management. The problems of the design and selection of a storage system and the handling of physical space allocation and physical data storage logistics are treated within the first task (storage management), while the problems of the programmer interface and logical data structures are treated within the second task (data management).

The terms physical data and logical data are sometimes used to distinguish between two levels of the data management function. The first, physical data, is intended to imply concern with data for purposes of storage and movement, with no concern for its internal structure, meaning, or ultimate use. The second, logical data, is intended to imply concern with the internal structure (names, formats, relationships, etc.), meaning, and use of the data. Actually, this distinction turns out to be less useful than might appear on the surface because whether a given data entity can be considered physical data or logical data depends on the particular process it is undergoing at the moment and may, in fact, be treated as a structureless commodity at one moment and undergo an internal interpretation and decomposition at the next. Thus, although the distinction between physical data and logical data as such is apt to be equivocal, there may be some argument for the concepts of physical data management or logical data management as descriptive of a function being carried out at the moment.

Actually, as will be made evident below, the subdivision of data management is multi-level, not dichotomous, with the identity of the stratum dependent upon the kind of structures being considered at the moment, so that the physical/logical distinction will not be generally made hereafter. The storage management/data management distinction will be made, however, and this will depend upon whether the basic concern is with the storage devices and their efficient use, or with the management, interpretation, and use of the data which is stored in the system.

2.2 THE STORAGE MANAGEMENT FUNCTION

The storage management function comprises the specification and selection of a set of storage devices with adequate capacity for the installation, and a strategy (possibly a hardware storage processor) for allocating and managing storage space. The storage management function becomes critical when the total capacity required exceeds the capacity of fast-access disk systems and the responsiveness required can neither be compromised nor can it be satisfied by traditional manual reel-handling approaches. There are several instances in NASA where fast (real-time) response is required in installations which manage data volumes in the trillion bit range. To meet these demands requires the utilization of the latest terabit (10^{12} bit) capacity mass storage devices and advanced level-changing strategies in multi-level (hierarchic) storage systems. By level-changing is meant the transfer of data from slow-access to faster-access storage. A subsidiary need is to take full advantage of recent advances in storage devices and storage management algorithms, and perhaps to specify needs not satisfied at the moment in order to channel further research in storage devices and strategies into the most productive areas.

Research in information storage has spawned two new technology areas which promise devices with characteristics distinctly differenct from those available at the present time. The first area, which can be called bit transfer devices, promises to fill the space-time performance gap between magnetic core and drum/disk memories. The second area is optical storage of information in the form of a hologram or interference pattern. This second area presents the unique capability of providing parallel access to a page of data in microseconds. The emergence of new storage device technologies such as bit transfer devices (which include magnetic domain devices, charge-coupled devices, and others) and holographic stores significantly alters constraints on access time and capacity so that storage hierarchies of much more flexible characteristics can become available.

Several questions arise as a result of these developments, and form the subject of this study.

(1) Can a set of standard storage system specifications be developed which will guide the development and procurement of storage devices so that the following conditions are satisfied:

(a) the needs of specific installations, applications, and programs are satisfied,

(b) an optimal set of devices for NASA-wide procurement are provided, and

(c) the gamut of storage device technologies are exploited without undue constraint?

(2) What device characteristics should be available in the storage hierarchy of a given facility to maximize the effectiveness of the system?

(3) What storage management strategies should be developed to fully utilize a multi-level storage hierarchy?

Bit transfer devices and holographic storage systems can be engineered with a wide range of characteristics such as capacity, block size, and mean access time. There are a number of reasons why NASA should remain abreast of developments in these technologies and seek to guide the characteristics of devices which are developed, and control or standardize the characteristics of the devices it procures. Without the guidance of large-scale users such as NASA, it is likely that a set of devices with non-systematic characteristics will become available as each manufacturer seeks out a particular, and what appears to him as a unique, market segment. However, from the standpoint of efficient software development, it is important that the systems programmer be presented with a consistent and uniform set of hardware environments so that storage allocation and strategies can be effectively evolved. The results of this project will be an important step in insuring that this does indeed take place.

The obvious areas to benefit from the utilization of a systematically specified set of storage devices are NASA's general purpose ground-based computing facilities and the general software development activities associated with them. (Specialized spaceborne computers may also benefit as bit transfer devices are developed and achieve wider use.) These results of a uniform set of performance specifications are likely:

(1) A set of storage devices, which more closely meet the needs of NASA, is apt to become available.

(2) The Data Management System's paging function, which provides for automatic data movement across storage levels, will be more effectively designed, implemented, and debugged.

(3) The way the Data Management System interfaces with the storage device controller will be designed once, and will not have to be modified for each device or installation.

2.3 THE DATA MANAGEMENT FUNCTION

The cost of problem analysis and programming dominates by far the cost of computer utilization. This dominance will increase in the future as hardware elements become more powerful and economical and problems become more complex and demanding of highly skilled analysis. As this trend continues, users are seeking ways of permitting economic transfer of programs and data from one installation or hardware type to another. At the same time, new storage technologies which both extend and interpolate the range of performance characteristics available (in both capacity and access time) compound the complexity of data management strategies, such as automatic level changing, and make it mandatory that an organized and systematic attack on these data management problems be launched.

A proposed approach to these problems is a multi-level hierarchy of data management services integrated into the hardware and software of a program's environment.

If this strategy is adopted for a number of NASA installations, it will provide a powerful tool for increasing the effectiveness of program development activities and also serve to achieve standardized environments for program execution, hence as a powerful vehicle for achieving program and data transferability across installations and machine types. Although the organization of a Data Management System component of an operating system which meets these objectives can be outlined, there are central unsolved problems which still exist for machines which are to be used to manage large collections of shared, inter-related data for users at different levels (system programmers, application programmers, analysts, managers, and planners). However, these are some of the central and fundamental problems in computing today and an organized assault on these problems promises a high payoff in terms of potential efficiencies and inroads into the solution of long-standing problems.

The traditional approaches to data and program transferability have been through (a) the use of compatible hardware types which have presented "equivalent" hardware interpreters for data and program, and (b) the use of standard higher level procedure-oriented languages and compilers to translate programs into a particular machine type. The first approach guarantees complete interchangeability only as long as the program's support software is duplicated but removes the possiblity of matching special hardware types to problem areas for which they may be particularly appropriate. This restriction is unnecessarily severe in many cases. The second approach avoids this restriction but there are many problem areas for which standard procedure-oriented languages have not been adopted. Indeed, many systems will continue to be implemented in assembler and macro level languages.

It is believed that the problem of data and program transferability can be approached most generally by extending approach "(a)" to include software as well as hardware interpreters. Software interpreters, or simulators, have been used in the past to transfer a machine language program from machine A to machine B. However, these have not been entirely successful or widely used. The success of this approach hinges on the ability to write programs for one of several standard environments and to describe in a standard, yet general, way the data structures which are to be transmitted and interpreted. To permit general applicability, from machine level programs and data through to higher level language programs and other character stream messages, requires that a wide range of data structures and languages be describable in a standard way. The language description standards must include the lexicographic, syntactic, and semantic levels.

SECTION 3. TECHNICAL DISSERTATION ON MANAGEMENT OF STORAGE AND DATA

3.1 INTRODUCTION

The problems of the proper management and control of data and the effective utilization of mass storage resources remain as key technical issues in the utilization of computers for data processing. These issues are critical for NASA because of the large amount of data collected in aerospace research and the need for its timely analysis and widespread use. The NASA environment - widespread distribution of resources, continuous and intense computer program development, a large variety of data, data users, and user needs - compounds data control and storage problems.

3.2 DATA NEEDS IN LARGE MULTI-USER SYSTEMS

A typical general purpose computing facility in NASA is used by a large number of specialists (and non-specialists) with different kinds of skills and different kinds of needs for languages, services, and data. One can, for convenience, identify the following types of user and his needs:

(a) Systems Programmer - the "toolmaker" who devises software systems for other programmers to use. He is the skilled programmer who develops compilers, data management systems, display systems, simulators, etc. He requires access to the basic hardware, may program in assembly language, and utilizes the more primitive data structures such as pages and segments.

(b) Applications Programmer - develops an accounting system, an inventory system, a real-time control system, the engineering analysis of a structure, etc., using more basic tools such as compilers, assemblers, debug systems, etc. His "products" may be used parametrically by non-programmers. He deals with a wide variety of data structures, from pages and segments to user-oriented files.

(c) The Analyst - not a programmer, but someone who may use application programs to solve a detailed technical problem. He understands a technical discipline and may be using the computer and programmed systems as tools to solve his problem. He is the statistician analyzing experimental data, the engineer developing a structure or analyzing a network, or the accountant preparing a specialized report. He understands the use and limitations of his software tools, the parameters required, etc. He is never concerned with machine-oriented data structures but deals with scalars, vectors, and arrays. He may wish to create a network-like structure of data elements.

(d) The Data Administrator - performs a service function in assisting other users in utilizing a large common data base to solve problems. He helps establish the design of the data base, to optimize its effectiveness over many applications and users, establishes access rights, priorities, and is responsible for the integrity of the data base. He helps to train other users in the effective use of data resources, and the data management system.

(e) The Manager - uses packaged applications in a prescribed way. He is not expected to be expert in either the structure of the data base or the use of programmed systems. He customarily interfaces the computer through the analyst, or data administrator, but may enter data or queries within a prescribed interface and range of capabilities. He deals only with application-oriented data entities and is not concerned with their coding or representation in the system.

3.3 BASIC CONCEPTS OF STORAGE AND DATA STRUCTURE

3.3.1 Storage and Data

A clear distinction between storage structures and data structures should be drawn at the outset of any discussion of the storage and data management aspects of computer systems. A terminology which will permit accurate discussion of storage structures and can serve as a foundation for the discussion of data structures will be developed. By a store is meant a device which contains addressable space in which digitally coded information, or data, may be recorded (written), and then retrieved (read). Devices whose storage space may be reused (that is, in which data may be erased and rewritten) form the most important class of stores.

From an orientation of the data system architect, storage devices can be examined with respect to their logical characteristics, rather than physical characteristics. Thus work-addressable stores such as magnetic-core, twistor, plated wire, and thin-film memories all fall into the same class, while magnetic drum and head-per-track disks are logically similar. The larger capacity memories with random-access plus sequential access aspects are moving-head drums and disks, and magnetic card and strip. Magnetic tapes can also be thought of as having random-access plus sequential access components, if retrieval and mounting of the reel is considered.

The principal characteristics of a storage device, from the standpoint of the data system architect, are the capacity (size) of the addressable storage element, data access time, and data transmission rate associated with it. A computer system will invariably contain stores of differing capacity and access time. These will range from low-capacity/fast access (primary) stores to high-capacity slower access (secondary, tertiary, etc.) stores. Typically, these real stores (devices) will be separately addressable, yet it will be convenient to think in terms of an address space which covers all accessible storage in a system. Intervals in this virtual address space, or virtual store, can then be mapped to areas of real storage. We will talk of allocating intervals in the address space to data entities, and will use the words cell, block, and track to designate units of addressable and allocatable storage space in which data may be recorded. In the discussion of data structures which will take place later, the terms word, page, and train will designate data entities which may be recorded in the previously mentioned storage elements.

The design of effective data management systems must be based on consistent, precise, coherent, and adequate notions about storage, data and associated mapping and transformation operations. Unfortunately, there is no commonly agreed upon theory of data management and therefore we must face up to the problem of terminology. That is precisely why it is important to define these terms and others as they arise. The particular terms used are not as important as the concepts which they represent.

The cell is the basic addressable unit of primary storage. A block is a fixed length sequence of cells, and a track is a variable length sequence of cells or blocks.

3.3.2 Storage Device Features

Storage devices are available in a variety of media, physical characteristics, and logical characteristics, and as we already mentioned, a given data processing system configuration will usually employ a variety of device types with different characteristics. This is due to two factors, (1) the existence of data entities with differing accessibility requirements, and (2) the economic tradeoffs of capacity and access time in storage devices.

Taking a broad view for the moment and considering data as anything which must be stored, it is clear that system data (programs and the entities they operate on) take many forms. Some must always be resident in primary storage (magnetic core memory), but others may be transferred from secondary storage (drum or disc) to be operated on.

Data processing system performance is usually measured in terms of the complementary aspects of throughput and responsiveness. Throughput refers to the speed at which operands are converted into results, and can be measured in bits per second (although throughput has also been measured in instructions per second). Responsiveness, sometimes called turn-around time, refers to the time delay between submitting a job and receiving the results. Throughput is maximized in the processing mode known as batch processing, in which operands for each task are batched or queued so that set-up time is small compared to productive running time. Responsiveness, on the other hand, is maximized by responding to each operand as it arrives. This processing mode, called on-line or real-time processing, is required when there are short real-time deadlines. In this mode, overhead factors such as set-up time and process switching may become large compared to productive processing and, as a result, throughput will suffer. It is clear that real-time processing may require fast access storage devices.

Throughput is limited by internal processing speeds, primary memory access times, and data transfer rates from storage and input/output devices. Responsiveness, however, is primarily limited by secondary storage access time, and external devices.

The most economical storage, in terms of capacity (bits) per unit cost (dollars) is magnetic tape reels. Magnetic tape also provides high physical density of storage (bits per cubic centimeter). The highest performance storage (from an access time and throughput standpoint) is magnetic core (or more exotic and costly media such as semiconductor registers, and thin film memories). Between these extremes lie a number of device types and media such as (in order of increasing performance) tape strips, disks, and drums.

3.3.3 Multi-level Storage Management

What the programmer would like is to be able to take advantage of the economies of multi-level storage hierarchy, yet have intervening hardware/software elements which permit him to view storage as a large single-level addressable store. A powerful method for accomplishing this is to give the programmer a large address space, called a vritual store or virtual memory, while delegating the responsibility for transporting data between levels ("level-changing") to a combination of special purpose hardware and software elements. When the transfer of data across levels takes place in fixed-size elements called pages, we call this an automatic paging system.

A virtual address (the address of a word in virtual memory) is made up of two fields, a page address and a word address within the page. Only a limited number of a program's pages will be stored in primary storage at any time. Hence, a page-mapping table is required to locate pages wherever they may be stored in the storage hierarchy. The page table, usually implemented in a hardware associative memory, transforms the page address into a block address in real storage. If it falls in the block address part of primary storage or "memory," access is made immediately. If the block address is not in memory, a page fault occurs and paging must take place. That is, a reference is made to the page on secondary storage, it is brought into memory, and the page table is corrected accordingly. During the paging time the task has been interrupted and the central processor has been executing another task.

The effectiveness of this process depends on a careful engineering of many factors that can influence the performance of the system. Some of these are as follows:

- the size of primary storage
- page size
- the speed of access to secondary storage
- the allocation strategies at each level of storage
- the number of competing processes.

3.4 STORAGE SYSTEMS

3.4.1 Technological Problems in Storage Management

Current and future aerospace experiments generate and utilize vast amounts of data. The total amount of data needed in a large scale computer installation on a regular basis is in the terabit (10^{12}-bit) range, exceeding by a factor of 100 to 1000 the amount which can be feasibly accommodated by expensive fast-access disks and drums. Consequently, a tiny fraction of the required, active, data can be entitled to on-line residence, accessible within about 150 milliseconds of the time its need is apparent, while the vast majority of data remains beyond a large gap in access time, accessible only after tapes or disk packs are found and fetched from a library and physically mounted by an operator.

A large gap in access time actually occurs at the other extreme as well, between the sub-microsecond access time of word addressable (e.g., magnetic core) primary memory and the multi-millisecond access time of drums and disks. This problem is often approached, and sometimes solved, by data access strategies called paging systems, based on multi-channel servicing of queued requests from independent programs. Not all problems and processing situations lend themselves to this approach.

In the past, there has been no good universal solution to the above access time gaps. One used paging systems for the secondary/primary storage data movement requests, and for the tertiary/secondary storage data movement, one used tape and disk-pack libraries, mounting and demounting of volumes by operators. With both of these approaches one suffered the attendant delays in responsiveness or job turn-around time, and comparatively inefficient utilization of the other resources committed to the job.

There was no acceptable middleground in these gaps in the past. However, current and emerging advances in storage technology, promise to fill the core/disk gap in one instance, and provide for terabit levels of on-line storage in another instance (at one hundredth the cost per bit os disk storage).

We will address the emerging technologies in each of these areas, and examine the problems facing NASA in effectively exploiting these developments.

3.4.2 Storage Hierarchies

All large-scale computer systems utilize a hierarchy of stores whose capacity, access time, and (usually) density increase as one moves "outward" from the central processor. This architecture is generally found in large machines because of two economic factors. The first is the cost and throughput of a central processor, which vary with the word length (or width of the memory bus) of the machine; since an instruction word must contain space for each cell address in primary storage, the word length and hence cost must vary as the base-two logarithm of the capacity of the word-addressable storage system. The second factor is that word-addressable storage has a higher cost per bit and faster access time than block-addressable storage.

Storage systems are organized into levels of addressability and accessibility such as cell, block, track, etc. in order to maximize the performance/cost ratio of a given computer facility.

These factors are reinforced by the fact that the access pattern to cells in main memory is not random, but tends to be made up of linear sequences from a small number of (program or data) arrays at any one time.[1] Stating this from a slightly different viewpoint, historically (and currently) hardware and software strategies are such as to require execution of linear arrays of instructions and/or data. Current machine architectures, as well as compilers and data management systems, are designed to exploit the fact that access to contiguous strings of cells and blocks in storage is more efficient than random access to cells of storage.

The memory hierarchy of a computer system usually encompasses three or more levels of storage and the largest systems can conceivably utilize five or more. At the present time these levels in the storage hierarchy usually embrace different device technologies, each providing an optimal capacity/cost tradeoff for the desired access characteristics. The technologies involved may include integrated circuits, magnetic core, magnetic drum, disk, strips, and tape. To these are being added the emerging technologies of bit transfer devices such as magnetic domain and charge coupled devices and optical storage devices, both bit-by-bit and holographic. Bit transfer devices are particularly attractive for NASA applications because they do not involve moving parts and hence can be made highly rugged and reliable. Also, they can be designed to fill the performance gap between core and drum. Several papers relevant to the role of bit transfer devices in storage hierarchies have recently appeared.[2, 3]

The technology of digital storage covers a vast domain from the nanosecond access of bipolar integrated circuit technology to kilosecond access of tape archives. A large-scale computer installation may actually involve this range of data access in its storage "hierarchy." Figure 3-1 shows that as we move across the technologies of integrated circuits, magnetic core, "bit transfer" devices such as charge-coupled and magnetic-domain devices, rotating drums and disk, strips, and tapes, we find that the range of twelve orders of magnitude of access time is accompanied by a range of ten orders of magnitude of capacity and seven orders of magnitude of cost per bit.

We also see, in the same chart, the progress made in the technology over the last four years, and projections for the next four derived from current projects in development.

There are two areas of particular interest to us with respect to this study:

(1) the performance gap between core and drums

(2) the upper range of capacity of a trillion bits or more.

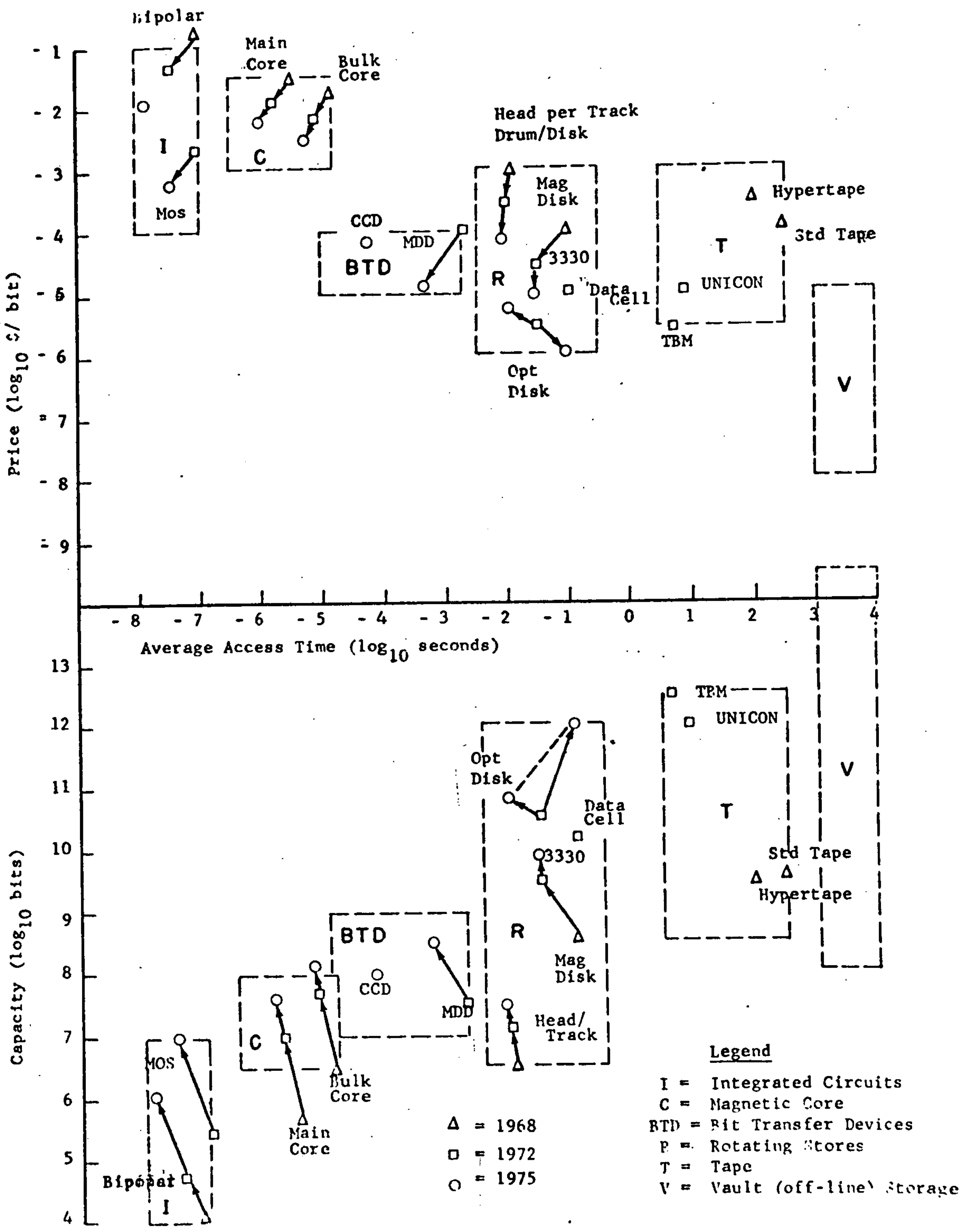


Figure 3-1. Storage Characteristics

The performance gap between core and drum can be filled possibly by magnetic domain-wall movement devices such as the "bubble store" being developed by Bell Telephone Laboratory and others. They will have an upper limit of capacity of about a billion bits.

Storage systems which have the potential capacity of a trillion bits on-line can be achieved with either extended performance magnetic head recording on wide tape, or optical recording on wide strips, tape, disks, or planes. Optical recording can be accomplished in either a bit-by-bit or holographic mode. Holographic mode optical recording will have a higher potential impact on computer system architecture than other forms of mass storage.

3.4.3 Bit Transfer Devices - The Billion Bit Store

The block-addressable random access storage devices in the billion bit range available at present or the near future are based on movable head disk technology. These devices require serial transfer of pages to primary memory before data is accessible to the central processor. Access time is on the order of tens to hundreds of milliseconds due to lateral head motion and rotational latency. However, there is an emerging class of devices, currently in the research stage, which promises to achieve block addressability in the billion-bit range with no mechanical motion and with an improvement in access time of at least two decimal orders of magnitude.

This class of device, of which Bell Telephone Laboratories' (BTL) "magnetic bubble" device is the the prime example, can be called bit transfer devices because they depend on shifting or propagating a serial bit stream along patterns or channels lithographed onto some substrate. The bit stream is sensed and generated at stationary positions along the propagation channel.

Devices in this class, along with BTL's "magnetic bubble" cylindrical domain device which has been mentioned, include charge coupled devices, magnetic domain propagation along helical wire and zig-zag patterns of magnetic film, and acoustic stress wave induced readout along magnetic strips on a glass substrate used for propagation of the stress wave.

These devices can achieve a storage density of 10^6 bits/cm.^{3} The BTL device, achieves its shift energy from a rotating magnetic field, eliminating the need of conductors to carry the shift signal. However, due to the need for a lithographed permalloy pattern at each bit position, and the use of a relatively critical material, the cost per bit of these devices is likely to remain higher than disk storage.

3.4.4 Mass Storage - The Trillion Bit Store

Conventional magnetic recording on moving media such as disks and tape have reached a limit of 10^3 bit/mm.^{2} in devices like the IBM 3330 disk, and Ampex TBM. Random access time in the hundred millisecond region probably requires that we keep recorded area below 10 m^2, making the limit of capacity around 10^{10} bit for conventional disk technology.

However, optical recording on magnetic and amorphous semiconductor (Ovonic) media can increase the practical recording density to 5×10^4 bit/mm^2. Several systems based on optical bit-by-bit recording on disks are under development at a 10^4 bit/mm^2 density. Thus, laser optical recording will put the trillion bit random access store within reach of disk technology. These developments are under way at Ampex and OMI. It is probable that the large computer manufacturers have similar projects underway.

An even more impressive improvement in storage density, hence access time, is promised through the use of holographic, rather than bit-by-bit recording. A hologram is a recording of the interference pattern formed when the light from an object illuminated by the coherent light of a laser is mixed with unmodulated light from the same laser. In digital holographic recording the hologram of a mosiac of light valves is formed on an area of one square millimeter or less. Because surface imperfections affect only the signal-to-noise ratio of the entire image, and not specific bits, and because a stationary medium is ued, (eliminating errors due to vibration, eccentricity, etc.), the full theoretical resolution of the medium can be approached. Although theoretical densities of 10^9 bit/mm^3 have been quoted for thick (volumetric) holograms, practical results for digital storage systems have been demonstrated only for surface holograms at a density of 10^5 bit/mm^2.

Although this would require 10 m^2 for a 10^{12} bit system, it would be difficult for a practical holographic system to handle this area of storage medium due to alignment tolerances. However, RCA feels a 10^{10} bit system with 10^5 pages and 10^5 bit/page accessible in microseconds is possible. Such a system, using .1 m^2 of medium area is probably the upper limit in systems whose entire medium area is electronically accessible.

Such a system would provide parallel access to a page of 10^5 bits. Serious thought would have to be given to the question of how this can be used, and whether, in fact, a multi-channel system with smaller pages would be better. In any event it is clear that the holographic store with microsecond access time to a large fraction of main memory will have a major impact on computer architecture.

Assuming such a device will become available, other interesting conclusions can be drawn. The need for BTD's such as discussed above is obviated on a performance basis (but perhaps not on a performance/cost basis). However, the holographic store does not obviate the need for archival storage since the holographic medium is limited in capacity. Replaceable media introduce alignment errors which reduce net achievable density.

The holographic storage device is not nearly as imminent as bit-by-bit optical storage on disk or the magnetic bubble store. Serious research problems still exist with respect to media, page composers, and deflection systems for holographic systems. Because of this, one cannot make an accurate estimate of when holographic systems will be operational and commercially available. On the other hand, OMI is planning commercial availability of the 4440 optical disk system in early 1973, although a more conservative estimate for the optical disk system would be late 1973.

3.5 DATA MANAGEMENT SYSTEM ARCHITECTURE

In a common data base system, the data base is treated as a pool of data which is shared among several different users in a variety of user jobs. An environment for processing data in a common data base system is shown in Figure 3-2. This figure shows the relationship between the user and the Activity Management System which schedules user jobs and divides the processing time of the system among several users and several tasks. A typical user task is shown with its relationship to a data management system, which is providing data services to it from the common data base (or as named in this figure, the data pool). The data management system provides services to the user tasks which allow the programmer to read and write data entities at the appropriate data structural level. Depending on that level, these entities may range from pages to records and fields from an item structure.

3.5.1 Levels of Data Management Service

Data Management Systems should offer a range of data structure types and will allow the user to involve himself, at an appropriate level, with the logistics of data movement and the strategies of data structuring and transformation. In effect, the user and the user task (the program actively serving him) are provided with one or more virtual machines which comprise his hardware/software environment.

Figure 3-3 illustrates the relationship between the user, his task program, and the hardware/software environment. The levels of virtual machines are listed as M_0, M_1, M_2, etc. The dashed lines show the flow of control and illustrate the logical dependence among service levels. For example, M_0, or M_0 and M_1, may be implemented in hardware, or possibly in hardware and firmware, while others, for example M_2, may be provided

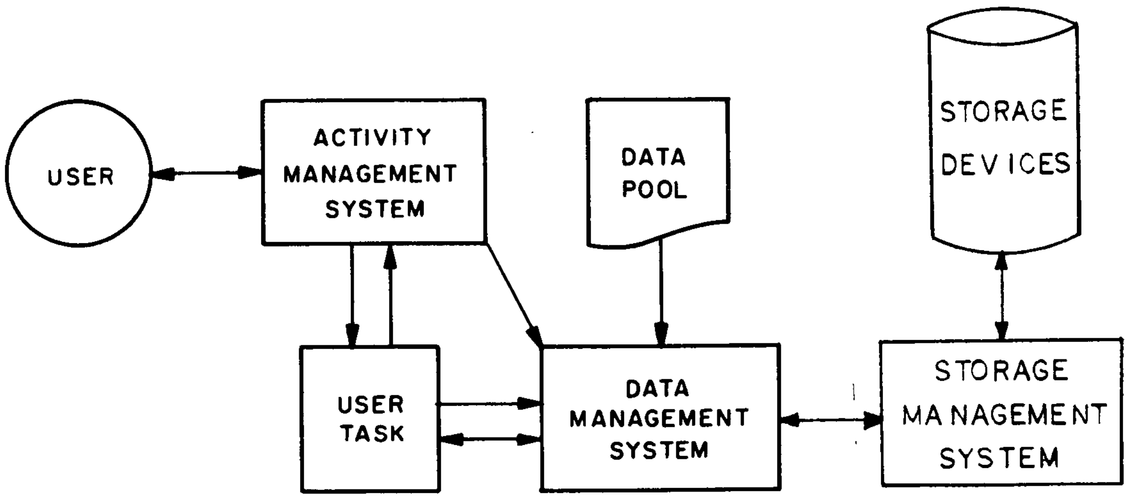


Figure 3-2. The User and User Task Environment

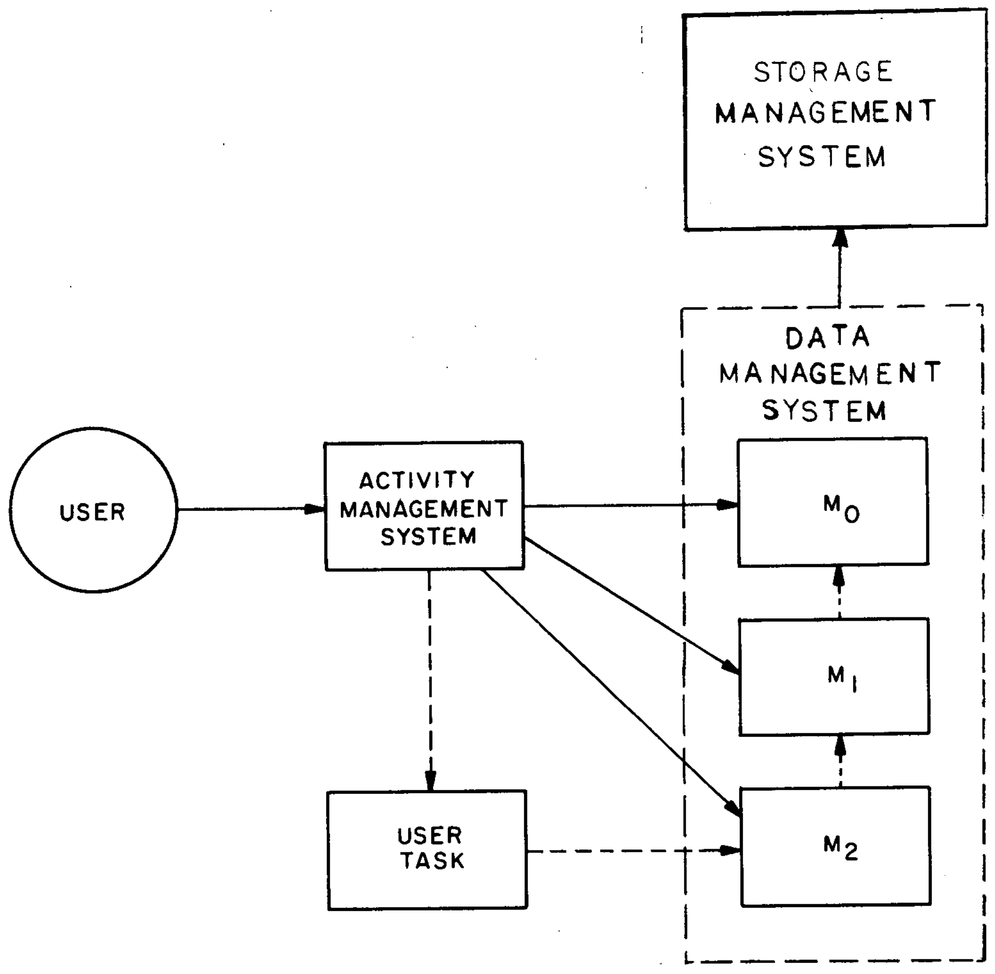


Figure 3-3. Service Levels

solely through software. It is useful to conjecture a system architecture in which the user task program is unaffected by how many service levels are hard or soft, and may be transferred to another environment or configuration in which the division between hard and soft levels is entirely different with no consequences to the program except possibly speed of execution.

This approach provides program transferability across configurations, or even hardware types, provided that the data structures and the service functions handled at each level have been standardized.

The Activity Management System is that part of the operating system which interfaces with the user on the one hand, and schedules the activities of program execution. It can provide for standard levels of human user interfaces just as it and the Data Management System provide for different levels of program interface.

The data structure types handled may range from the very machine oriented sequences of continuous words and pages to the management user oriented hierarchic files with variable length fields in a variety of alphabets and codes.

Large-scale computer users, with large data base and program investments need not be tolerant of a situation which prevents effective program and data transferability across different hardware types and even configurations. Standardization at the procedural language level (such as COBOL) is inadequate for the system programmer and for many common data base applications. What is required, and can be accomplished, is standardization at a number of operating system (or call it virtual machine) levels. CODASYL has developed a data description language at the application level but is far from a solution to the problem of standardizing program/machine interfaces. Research into this approach to the problem of program transferability is just beginning to get underway.

The problems of providing data management services within an installation, transferring data from one computer to another, and interpreting the data correctly, can be approached as a problem of constructing an adequate range of data structure types and devising a standard way of describing these types. No one level of data description is adequate. Rather, there must be a range of structures which go from the highly machine oriented cell structure to the user-information oriented structure. It is felt that a hierarchy of data structure types which is adequate for this task can be devised.

The problem of data description, however, is but a special case of language specification and special attention must be paid to the lexicographic, syntactic, and semantic aspects of specification. Some recently developed language specification languages and generalized language processors can be brought to bear on this problem. These processors can be employed as a standard interface in a network of different computer types. When furnished with the description of the structure (languages) to be accepted, they can carry out the appropriate translation and interpretation.

The stratification of data management services into a number of standard levels would make it appear to the programmer that at any one moment, he is interfacing with one of a number of virtual machines which form an upward compatible hierarchy. These services may be, in fact, provided by a mix of hardware and software modules which depends on the particular system implementation, and the hardware types being used in a given instance. Thus, each program has, as its interpreter, a virtual machine whose interface with the program is known but whose composition may vary and is irrelevant except for timing considerations. A given installation, because of hardware and software modules used, may provide interpretive virtual machines only up to a given level, requiring programs written for a higher level virtual machine to undergo a translation process down to the appropriate level before interpretation can take place.

That the goals outlined above may be difficult to achieve in today's technology is well recognized. However, it may be that at this point in time the technology may be mature enough, and the payoff to NASA great enough that NASA should embark on a comprehensive project to ensure comprehensive data management services for computer users, and software transferability across a wide range of problem areas and hardware types without limiting the development of new languages and machines.

SECTION 4. STATEMENT OF THE PROBLEM AND REQUIREMENTS

This Section defines the problems to be addressed in the project, the basic purpose of which is to develop more economical ways of integrating new storage devices and data processing programs into a computer system which has a large and growing data base to manage. This is to be developed in two major tasks:

(1) the development of methods for specifying hierarchies of storage devices and storage management strategies

(2) the development of data management services for various types of users.

These are recognized as complementary facets of the problem of processing the vast amounts of data to be spawned by future NASA experiments. The approach described here is one which improves the programability of data storage systems which must be developed, as well as the transferability of programs and data across installations.

4.1 THE STORAGE DEVICE AND THE STORAGE MANAGEMENT SYSTEM

The objective of this task is to derive a storage system architecture and a set of storage device specifications which will meet the long range requirements of NASA for massive (trillion bit) storage. This system should accommodate new storage devices as they are developed with a minimum of disruptive impact on mission-oriented programs and programmers. The task involves defining a storage management system architecture, an optimum set of device characteristics suitable for NASA-wide (or, preferably, industry-wide) procurement, and analyzing the realizability of those standards, and their potential acceptability by the manufacturer and users.

4.1.1 Defining a Storage Management System Architecture

It has been pointed out (Section 3.3) that storage devices which provide fast access and word addressability invariably have a high cost per bit while storage devices with low cost per bit and high capacity invariably have a relatively slow access time. However, within these constraints, and with one or two notable exceptions, storage devices with a wide range of performance characteristics are now available.

One of these exceptions is the performance gap between core and disk storage (Section 3.4) and the other is the promise of both high performance and high capacity from optical holographic storage technology. However, devices which will fill the access-time/capacity gap between core and disk are now under development and may become available within a few years. At the high capacity end of the scale, although holographic technology promises to achieve a break-through in performance, price, and capacity, these devices may not achieve a "product" status for another four to eight years.

As a result of this situation, the use of a hierarchy of storage devices utilizing several technologies will remain the only effective solution to providing optimal performance in those computer installations which must provide access to an extremely large volume of data. In order to realize the promise of high performance in a hierarchic storage configuration a complex logistical problem of storage allocation and data movement across levels of storage must be solved. This problem has indeed been solved in specific installations and it is conjectured that a general solution, insensitive to specific computer and storage device types is possible. When realized, this hardware/software processor, called the Storage Management System, will automatically allocate storage to data and change the highest storage level allocated to a specific data entity (page, or segment) either as a result of a specific program reference to a data entity, or in anticipation of such reference. This process of dynamic "level changing" will be a prime function of the Storage Management System.

As a result of this subtask, the functional specifications of a Storage Management System will be defined, along with at least one strategy for its realization and use. The specific design aspects which will be specified are:

(1) the classes of storage devices to interfaced

(2) the storage device control interface standards to be used

(3) the data and control interface with the computer

(4) the interface with the operating system

(5) the level-changing algorithms

(6) the net access time and throughput performance achievable with a specified problem mix.

4.1.2 Defining an Optimal Set of Storage Device Characteristics

Assuming the current and emerging state of digital storage technology is accurately portrayed in Section 3.4, this subtask will develop the performance and control specifications for an optimal set of storage devices suitable for NASA-wide (and perhaps industry-wide) procurement. The parameters to be specified, for each storage device, are as follows:

(1) unit of addressability (block size)

(2) address domain

(3) access time

(4) transmission rate

(5) channel width

(6) control commands.

In addition to these performance parameters, the maximum capacity, and the cost per bit as a function of capacity, should be estimated for various storage technologies.

In order that the storage needs of NASA installations are satisfied with a high degree of confidence, the volume of data stored and the storage needs in a representative selection (possibly all) of NASA's general purpose data processing installations will be evaluated. Statistics gathered should include estimates of data volume vs. access time required for all applications with significant storage needs. The aggregate flow of data across installations should also be estimated. Instances of data stored redundantly (at more than one installation) should be carefully noted.

4.1.3 Storage System Simulator

In order to estimate the performance yielded by a given Storage Management System and device set, a system flow model and simulation program will be developed. The model will be capable of experimentally determining the performance achieved by a given mix of devices (with the above parameters) in a storage hierarchy and/or a given level changing or control strategy in the Storage Manager.

The parameters of the flow model and simulation program will be the problem mix, channel characteristics, device characteristics, device mix, and storage allocation and level-changing algorithm. The simulation program should be designed to change these parameters conveniently and record the resulting performance changes.

The model, when implemented and tested, will be used to estimate the effectiveness of various mass storage device hierarchies at representative installations, for a number of typical job mixes.

4.1.4 Feasibility Studies and Analysis

As a final subtask of the Storage Management System task, a series of studies and analyses concerning the feasibility of realizing the Storage Processor and the storage devices will be carried out. The following aspects should be examined:

(1) the technical feasibility of developing industry-wide standards for given classes of system capability

(2) the extent of acceptability of these standards likely by the manufacturer and the user

(3) the likely effect of these performance and interface standards on the development of systems and the development and exploitation of storage technology

(4) estimates of the realizability of storage devices with the given characteristics and the likely dates of first commercial availability of devices with those characteristics.

4.2 THE DATA MANAGEMENT SYSTEM AND ITS INTERFACE WITH THE USER

The management of data goes hand in hand with the management of storage. The Storage Manager must interface with the Data Manager and it, in turn, must interface with the user. Both systems involve hardware and software aspects.

The objective of this task is to develop a data management system architecture and its interface with the user. This will involve specification of an appropriate set of data structure types, data management services, and user languages appropriate for each type of user of the data processing facility. The user may be a systems programmer, application programmer, data base administrator, engineer, analyst, or manager. These users, their roles, responsibilities, capabilities, and needs have been discussed previously (Section 3.2). An important subtask will be to develop and analyze alternate strategies for developing each level in the data management system and evaluating its feasibility of implementation and likelihood of its industry-wide acceptability.

4.2.1 Specification of a Set of Data Types

The management of large volumes of data is one of the key problems which faces almost all users of general-purpose digital computers in NASA. It has been recognized for some time that this problem can be approached through the use of generalized data management tools. However, the wide variety of users within NASA, and the disparity of their needs (see Section 3.2) raises the challenge (and the opportunity) of devising a comprehensive approach to the data management problem with a system whose overall architecture makes optimal use of its structural elements in meeting the needs of those users.

The Data Management System (DMS) should provide the capability to define, create, store, retrieve, and modify the types of data entities and data structures required by the range of users and applications which the system must serve. These must range from the machine-oriented data structures of the systems programmer, often tied to a logical or physical address space, to the application-oriented data structures of the analyst, application programmer, or manager, dealing with array, records, and fields, usually devoid of addressing considerations.

The following types of data structures should be considered, although the set is not unique, independent, or exhaustive:

(1) machine-oriented data structures, such as words, pages, and segments (word sequences), which can be allocated directly to storage structures, should be defined. A word would be allocated to a cell in primary storage, a page to an addressable block in secondary storage, a sequence of pages or a segment to a track, etc. A word and page (all cell and block) would be fixed, system-defined (rather than user-defined) entities.

(2) Fixed length elements whose size is user defined should be handled. Also sequences or networks of such elements should be definable. This would permit list structures to be handled.

(3) Variable length entities, and sequences of variable length entities should be handled.

(4) Fixed and variable-length records whose internal structure is defined to the system should be handled. The internal record structure may contain fields of various types (numeric, character strings, etc.), embedded records, and embedded files. Files (record sequences) should be handled at any level in the hierarchic structure.

A general capability which should be provided (as is evident in the above structural types) is the ability to create sequences and hierarchies of the definable elements. This is the basis of many processing strategies. There are several basic methods for providing access to entities in a sequence and these should be provided by the system. These are as follows:

(1) spatial access - the target entity is contiguous to the current entity in an address space

(2) tabular access - the location of the target entity is determined indirectly by reference to a contiguous (spatial access) list of links

(3) chained access - the location of the target entity is given by a link in the current entity

(4) calculated access - the location of the target entity is a computed function of a unique access key, or name, associated with the entity.

Additions or modifications to the data structures and entity types listed above may be proposed, and the more complex entities can often be defined and implemented as a sequence, tree, or network of a more primitive type or in terms of a mapping of its elements onto another type.

4.2.2 Development of a DMS Language

As part of the previous subtask, a comprehensive set of data structures will be defined as a basis for a set of standard data services to be provided to the programmer and other users. A feasible candidate set of data structures has been discussed above (Section 4.2.1). The data services, user languages, and the DMS/program interface concept will be developed in the current subtask.

The use of a "host-language" concept shall be investigated and adopted if feasible. In a host-language approach the data services are accessible through service calls that originate in standard programming languages, and the existing language processors need not be modified to accommodate the additional user capabilities.

The user language should have descriptive, conditional, and imperative components. The descriptive component should be a machine and host language independent data description language appropriate for each class of data structures. The conditional component should permit boolean logical conditions to be imposed which defines a subset of the data base or a unique data base element. The imperative component should define the operation, command, or procedure to be carried out by the DMS.

The success of the approach to data access and retrieval and data and program transferability being proposed here depends on the adoption of comprehensive data description and data manipulation languages (DDL and DML). These languages must meet a number of requirements:

(1) They must be able to handle at least the range of data types listed above.

(2) The requirement to handle item structures and source language messages means that they should be capable of expressing the symbols and syntax of context-free languages.

(3) They should be capable of expressing the semantics of translation or interpretation of the source language strings, including procedure calls to other processors.

(4) They should be representable in a graphical format that exhibits the structure of the language or data being described so that it represents an effective tool for human communication.

(5) They should also be representable by linear strings amendable to transmission and computer input, and it should be easy

to translate from the graphical to the linear version, and vice versa.

4.2.3 Development of a DMS Architecture

As a result of this subtask, the structure of a Data Management System will be defined.

A comprehensive set of data structures and a set of standard data services will be defined as a result of the previous two subtasks. Their significance is as follows:

(1) Each of these data structure types implies a set of data management services, such as "Read," "Write," and "Modify," which should interface with the program and user.

(2) Each set of data management services can be considered as an essential component of a virtual machine which forms a standard host environment for programs dealing with data at that level.

(3) The definition of standard program host environments and an adequate data definition language can be the key to transferability of programs and data from one environment to another.

(4) We are witnessing the emergence of new classes of machine types making use of large scale integration technology and, in several instances these machines are based on microprogrammed logic, making it possible to emulate in hardware or firmware the standard environment for which a program was developed.

Thus, the architecture of the DMS can be viewed as a set of virtual machine types which will provide standard data management services to the program. In each of these standard interfaces, the data should be viewed as a system resource which is made accessible to the user at an appropriate level of control, provided that access rights at that level have been established and verified.

In the largest computers, all or most of these virtual machines will be microprogrammed to be effectively provided as an emulated machine.

In the smallest computers, these virtual machines will be simulated by data management service programs which provide a software implementation of the virtual machines. Future generations of machine types may not be directly compatible with today's machines but may, through microprogramming, offer increased flexibility to adapt to special requirements or to emulate other machine types, existing or proposed.

The stratification of data management services into a number of standard levels would make it appear to the programmer that at any one moment, he is interfacing with one of a number of virtual machines which form an upward compatible hierarchy. These services may be, in fact, provided by a mix of hardware and software modules which depends on the particular system implementation, and the hardware types being used in a given instance. Thus, each program has, as its interpreter, a virtual machine whose interface with the program is known but whose composition may vary and is irrelevant except for timing considerations. A given installation, because of hardware and software modules used, may provide interpretive virtual machines only up to a given level, requiring programs written for a higher level virtual machine to undergo a translation process down to the appropriate level before interpretation can take place.

4.2.4 Feasibility Study and Analysis

As a final subtask of the DMS task, a series of studies and analyses concerning the feasibility of realizing the DMS will be carried out. The following aspects should be examined.

(1) the effect of the proposed DMS languages on the development and exploitation of DMS technology

(2) the relationship of other proposed systems, such as the CODASYL Data Base Task Group proposal, on the DMS

(3) the extent of acceptability of these standards likely by the manufacturer, software development companies, and the user

(4) estimates of the implementability of a system with the given characteristics, a likely date of availability of initial models, and a development plan for its realization

AUERBACH

(5) the level of performance likely to be achieved by such a system and an estimate of the degree to which performance is sacrificed (if at all) in achieving the other goals of the system.

SECTION 5. DISCUSSION OF THE PROJECT

Aspects of the Mass Storage project which will be discussed in this section are technical feasibility, project structure, possible approaches, and anticipated results. The two major tasks, storage management and data management will be discussed separately.

5.1 STORAGE MANAGEMENT

5.1.1 Technical Feasibility

The Storage Management task is concerned with answering several questions which arise in utilizing modern mass storage technology:

(1) Can a set of standard storage system specifications be developed which will guide the development and procurement of storage devices so that the following conditions are satisfied:

(a) the needs of specific installations, applications, and programs are satisfied,

(b) an optimal set of devices for NASA-wide procurement are provided, and

(c) the gamut of storage device technologies are exploited without undue constraint?

(2) What device characteristics should be available in the storage hierarchy of a given facility to maximize the effectiveness of the system?

(3) How should the set of mass storage devices, viewed as a heterogeneous resource, be allocated to best serve the synamic needs of the active (and soon to be active) jobs in an installation?

(4) Will the specification of devices with new performance characteristics influence the liklihood of commercial availability of such devices?

These questions will be addressed in specific subtasks of this task. The relationship between capacity, access time, and cost per bit appears to be subject to physical constraints characteristic of each storage technology and device configuration (See Fig. 3-1). The question of an optimal device mix for a given installation is therefore a question of performance/cost tradeoffs and can be approached through application of known economic modeling and linear programming techniques with a high degree of confidence.

The technology of magnetic storage on disk is fully matured. The technologies of bubble memories (and other bit transfer devices) and optical recording on disk are still in the laboratory stage but several "product" projections have already been made so that a reasonable performance/cost estimate can be made. Therefore, the feasibility of making an accurate estimate of overall system performance, and specifying an effective, realizable set of storage devices, depends on the ability to accurately model the performance of a hierarchic storage system. This is the problem addressed in the task discussed in Section 4.1.3, and the answer depends on our ability to parameterize the performance of devices utilizing different technologies and the skill with which the model is constructed. It is felt to be within the scope of today's technology.

The realizability of the system and the degree of commercial acceptance of the devices and control techniques will be the subject of the final subtask.

5.1.2 Possible Approach to Storage Management

Several terabit (trillion bit) storage devices (for example, Ampex TBM, Grumman Masstape, and General Precision Unicon) have already reached "product" status. However, their inherently slow access time requires that special approaches be taken to assure the overall responsiveness of the Storage Management System. This discussion is concerned with one particular problem in the utilization of a multi-level mass storage hierarchy--the automatic migration of data from slow access to fast access storage in response to, or in anticipation of, data requests.

Broadly speaking, two classes of service are meaningful to consider for a Storage Management System to render to one or more user computers. These are:

Solution 1 - At the job scheduling level, data sets or files to be used by a job must be made available before committing core memory and CPU resources to it. This may require human intervention to mount tapes and/or disk packs. Also, files which normally reside on expensive on-line storage are wasting valuable resources during periods of inactivity. A Storage System which could just support migration of these data sets at beginning and end of the jobs which use them has much to recommend it.

Solution 2 - It is conceivable to think of an elaborate, high performance Storage Management System as a direct replacement for secondary storage devices. Such an SMS would emulate disk storage in responding to each program request issued in the host. The SMS in this case must comprise a hierarchy of storage devices and contain sophisticated software for its management. The performance of such a subsystem would at times greatly surpass conventional secondary storage devices, as many transfer requests could be anticipated and responded to via core-to-core movement.

5.1.2.1 Solution 1 - File Access. The fundamental role of a Storage Management System is as a repository for files or data sets, where a data set is copied onto faster (e.g., disk) storage when needed by a job. No longer is there a need to consume expensive secondary storage space for data sets not in use. No longer is it necessary to keep less frequently used data sets off-line.

The great appeal of Solution 1 is the promise of the tremendous payoff achievable; yet relatively little software or hardware complication is entailed, viz:

- Instead of mounting and demounting volumes, user-programmed routines at job initiation and termination see to the migration of data sets or files.
- Manufacturers' design guidelines include minimizing or obviating operating system modification.
- Some of the recent terabit stores are meant for large I/O transfers as designed, fitting the requirement with neither gross deficiency nor surplus capability.

There are properties and areas of optimization to look for in an initial selection, implementation and in future development, which bear upon performance of a "Solution 1" Storage Management System:

(1) It would be desirable for the SMS to directly share disk facilities with user (or "host") computer systems, so that data set migration does not require traffic through host computer main memory. The sharing and allocation of disk storage must be completely controlled by software in the host in order to avoid problems of deadly embrace and to maintain integrity of directories and data. None of the products are designed for this today, but some are more amenable to such augmentation of capability than others.

(2) The ability of a product to accommodate various strategic measures in file allocation and request scheduling is worthy of attention as significant gains in performance are then possible.

(3) The storage device should be highly programmable by the user, and amenable to variations in configuration. It is unlikely that any standard device can be the most cost-effective solution of a given problem without some tailoring.

5.1.2.2 Solution 2 - Access At All Levels. The notion of centralizing all file storage and access into a single, responsive, on-line storage device is certainly an attractive one. A storage processing subsystem which could hold a trillion bits and perform like a drum for most requests is an ambitious idea, but plausible.

The high capacity is achievable with mass storage devices already available.* Such performance, however, requires that the SMS is a computer system in its own right, with drum storage and core storage, and sophisticated hierarchical file management strategies.

When a file is first requested, a copy is made from the terabit store to the drum, still within the SMS. During processing, fragments of the file pass between drum and SMS core memory in such a way as to maximize the probability that a given request from a host may be serviced by immediate core-to-core transfer. With this type of system, a heavily I/O bound application might run in as little as a tenth of the time it takes in a conventional environment using on-line disk files. Such a scheme applies most obviously to sequentially accessed files, and is also promising for indexed files. The actual performance gain will be a function of the core and drum capacity of the SMS, the investment made in SMS software, and the percentage of data traffic having predictable order. It is also possible, but not essential to the main theme, to assign spooling functions to this type of SMS. In that case, nearly all secondary storage could be removed from the host systems. Also, the sequential access typical in reading spooled input and producing spooled output would be serviced at core-to-core speed. Large print files could descend immediately to the terabit store with quite an economic advantage.

Although the implementation of this type of SMS requires a significant investment in hardware and software on the SMS side, careful planning will minimize the need for modification on the host side. Any scheme will require further study and verification, but a strategy such as the following, or a variation on it is the key to a clean interface with the manufacturer's operating system.

* For example, Ampex TBM, Grumman MASSTAPE, and GPC Unicon.

A simple method is to deceive the operating system into "thinking" that each permanent data set has its own volume and competes with no other data set for that space. In reality, this virtual space will be managed by the SMS. On the host side each permanent data set is therefore assigned a serial number for its virtual volume which becomes the parameter for the command used to catalog the data set. If it proves desirable for an SMS to support temporary data sets, the same concept will apply with some additional considerations.

With the help of hardware emulation, no changes would be necessary to host I/O schedulers. The SMS will receive head and track specifications, translate them to its internally managed space, and set up responses identical to those of the simulated drum or disk facility. If emulation at the interface is not feasible, it would be necessary to develop software for the host to intercept supervisor calls to substitute the needed channel commands. Also, tailored channel-end software would be needed. While this is non-trivial, it is not impractical, and is minor compared to the SMS internal software system.

5.2 DATA MANAGEMENT SYSTEM

5.2.1 Technical Feasibility

The Data Management System task will be carried out in a number of phases. As specified in Section 4.2, it is basically a design problem, followed by an analysis and evaluation of that design. Implementation of the system will be undertaken only if indicated by the results of the analysis and evaluation phase, and may in fact be undertaken by the manufacturers and/or independent software developers if a system with a high degree of industry-wide user acceptability is specified.

A project with many of the elements of this task, was one just carried out by CODASYL's Data Base Task Group (DBTG). The DBTG developed what they intended to be accepted as an industry-wide standard specification for a management system. However, the DBTG did not view their task with the breadth of scope intended for the NASA DMS task. The DBTG was concerned with the data base management question at the item level (heirarchically related records and files) and with providing a COBOL-oriented capability.

A major portion of the industry has taken issue with the DBTG proposal. (In spite of this, some software developers are implementing subsets of the system). It is felt that the DBTG system fails because of its limited scopte and needless complexity due to lack of development of more primitive systems upon which it can be built. The recognized shortcomings of the DBTG system can be avoided with the more global approach to the problem, which is inherent in the system specified here.

The advantages of simplicity and structural modularity result because:

(1) The system can be designed in terms of interrelated modules which provide data services for a related system of data structures, and

(2) it should be possible to implement each system module in a way which takes full advantage of the capability of the computer hardware, the Storage Manager, more primitive modules and the standard language processors of the system.

5.2.2 Technical Approach

One approach to the design of a set of data structures which meets the requirements outlined in Section 4.2 is given below.

5.2.2.1 Page and Train. The most primitive data structures under consideration are the page and a sequence of pages, the train. Figure 5-1 shows a track in storage that is allocated in units of fixed-length entities called blocks. A train of pages is to be defined and stored on blocks in that track. This figure illustrates one way of accomplishing this without requiring the allocation of contiguous pages to contiguous blocks. In this example, a train index (an indexed list with pages as list entities) is used. The address of each page is the block number in the address space for the track. Unused space in the data train is indicated by a zero tag.

in the index entry. Periodically, the blocks on which unused pages are stored are collected and added to a list called a space train index. This is simply a list of unused pages, referenced through the index shown on the righthand side of the figure.

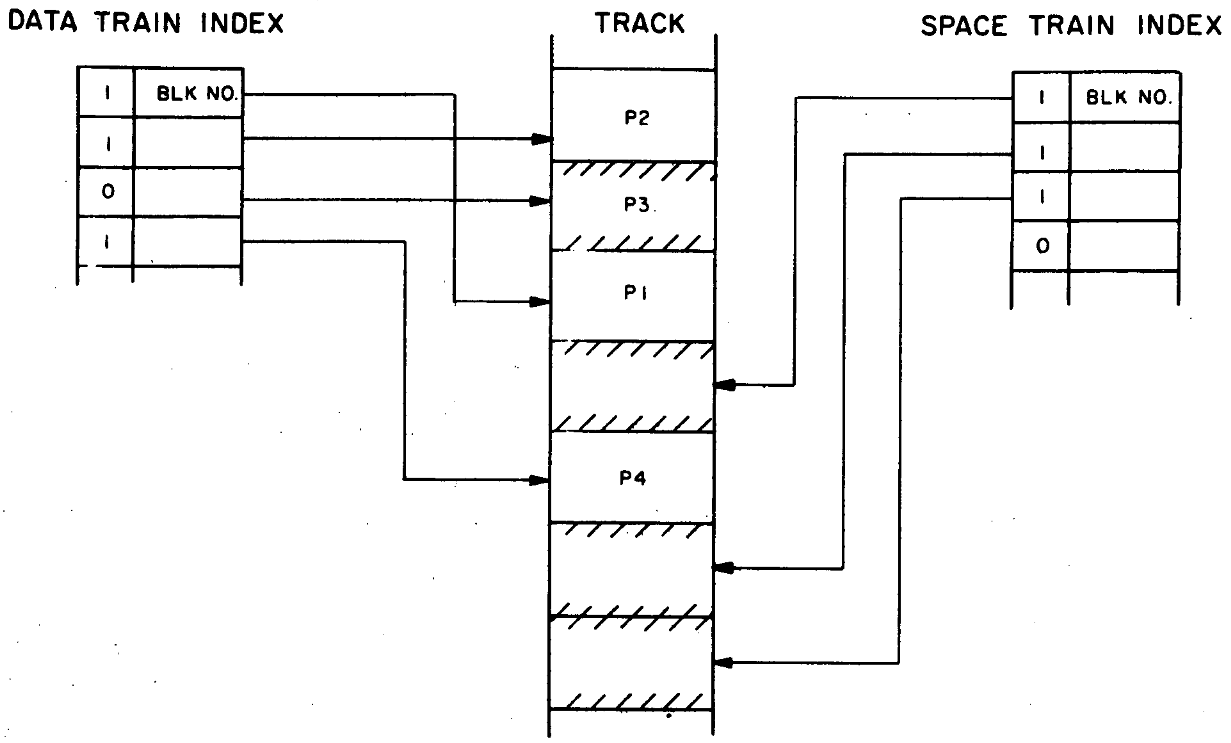


Figure 5-1. Page Addressing and Block Allocation

The smallest element of addressable data is a word; a page is a fixed-length sequence of words; and a sequence of words of arbitrary length is called a segment. These terms and their definitions are used by a large segment of the computing field.

The page is used as the unit of data movement between primary and secondary storage; its size is fixed, and it is equal to the unit of storage allocation, the block. With the page and the train defined, we can consider sequences of elements that possess user-defined size but that bear a fixed relationship to the unit of data movement. Such a structure is called a strip, and its element, a bead. There can be a constant ratio of beads per page throughout the strip, so that additional tables at the bead level are not necessary. The system must be aware of the size of the bead as well as the internal structure of fixed fields that may represent information elements (properties of some object) or links to other beads (relationships among objects).

Figure 5-2 shows a bead that may exist in a network (plex). In this case the bead has been divided into two areas: one area contains the properties of the object to which it corresponds; the other area contains links to other beads, representing a number of different types of relationships with other entities in the structure. Two links are illustrated; one points to the next bead of the same type; one points down to a bead of perhaps a different type at a different level of the hierarchy.

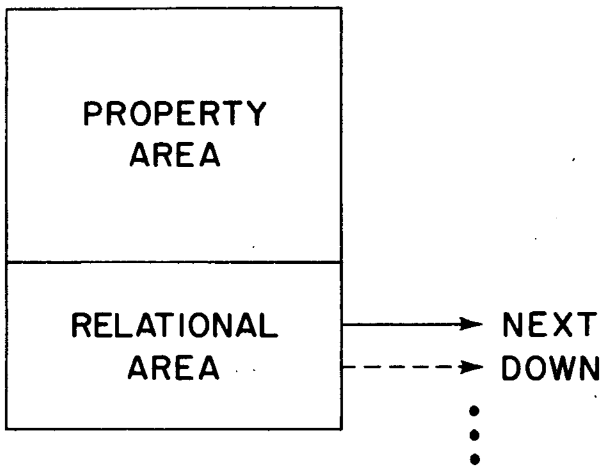


Figure 5-2. A Plex Bead

Figure 5-3 illustrates how beads of different types that exist in different strips may be linked together to form a plex. A <u>root strip</u> contains pointers to each strip in the plex. Pointers from one bead to another model the different types of relationships previously discussed.

5.2.2.2 Stream. At times it is inconvenient to consider sequences of entities of the same fixed length. A sequence of entities of variable lengths is called a stream; Figure 5-4 details how a stream may control the processing of data structures. In this figure it is assumed that each stream element is a segment. The elements are called R1, R2, R3, etc. They are stored on a train, with pages called P1, P2, P3, etc. This mapping is defined in the stream index; entries include the page number and the word with which each stream intity (segment) begins on the page. As before, the pages in the page train are mapped to blocks in a track by means of a train index. A stream is the structure that can appropriately be used to represent several types of entities, processed by hardware and software interpreters. For example, a task program is built as a stream in which each sub-program procedure is a sequence of machine instructions. The data structure, processed by the task, can be considered a stream of sub-program data areas.

5.2.2.3 The Item. The composite data structures which have been defined up to this point -- the train, the plex, and the stream -- and the virtual machines and services which have been implied to service them provide increasingly more general tools for the programmer. Nevertheless, these tools have been designed primarily for the systems programmer, or toolmaker, rather than for the applications programmer or even the non-programming consumer interested in commercial file-processing applications.

Appropriate tools for the applications programmer and non-programming consumer should provide the ability to define, create, and transform structures of named items without regard to the access strategies employed. Yet, because these structures may involve large quantities of information with varying access requirements and processing deadlines, it should be possible to avoid the wide-spread use of internal links (as in the plex structure) and to provide the user with the space-time trade off of creating and destroying indexes whenever appropriate. Furthermore, it should be possible to manage a large common data base of commercial file-oriented data with proper access and recovery safeguards and with a convenient language interface for the non-programming user or consumer. To meet the needs of this type of user, the item structure (and item machine) were devised.

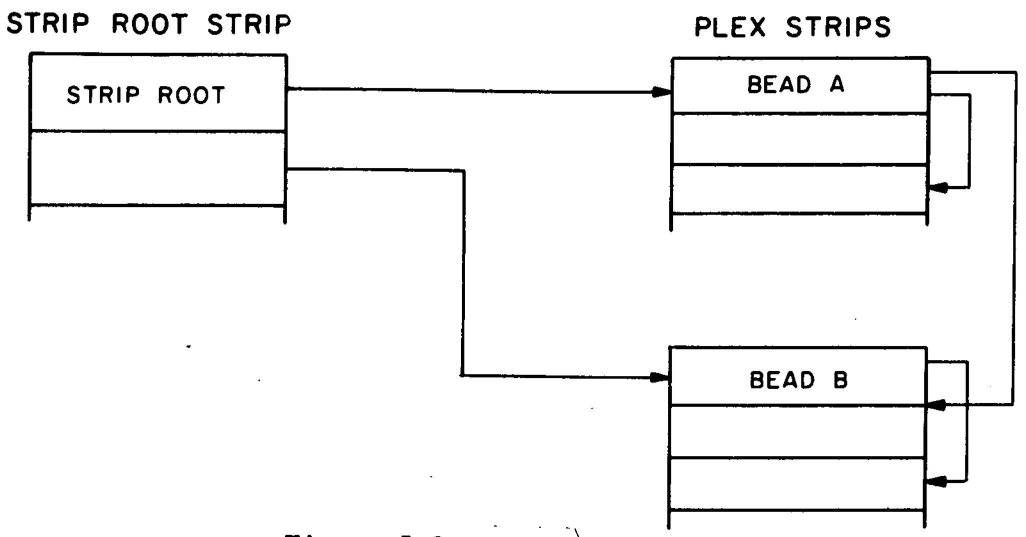


Figure 5-3. Plex

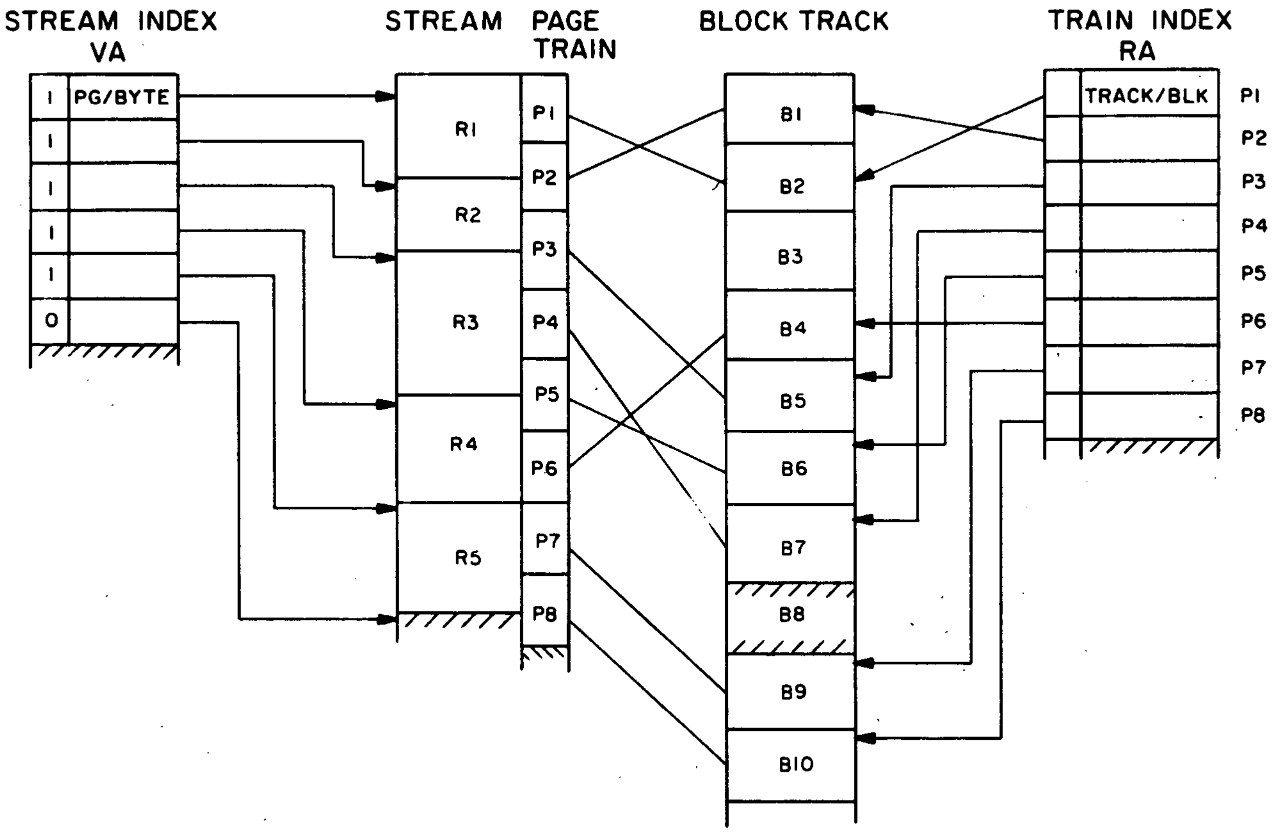


Figure 5-4. Stream Addressing

Items are named and defined structures, devoid of any reference to an address space. The item is both application and consumer oriented, rather than access or programmer oriented.

There are three types of items: the record, the file, and the field. The first two are composite items: They may contain sub-items. The field is a simple or terminal item: It contains no sub-items. Every item is defined by its name, type and size. The size of a composite item is the number of immediate sub-items it contains; the size of a field is the number of characters in the field. A file or a field may be variable in length, but a record must contain a fixed number of immediate sub-items. These item types are defined more precisely as follows:

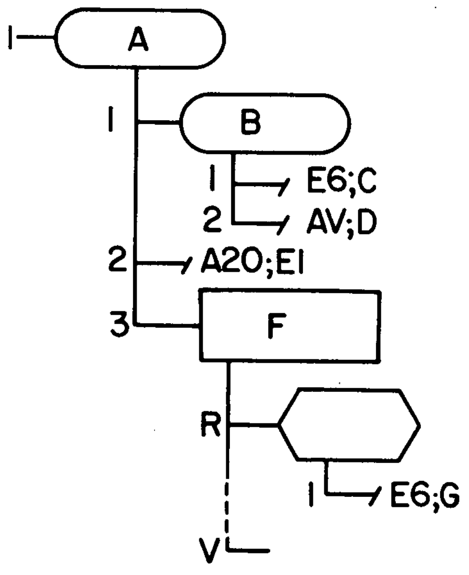
- A record is a composite item that contains some fixed sequence of sub-items of any type
- A file is a fixed or variable sequence of records with the same definition.
- A field is a fixed or variable sequence of characters of a given alphabet.

A record that is not part of a file (that is, it is not repeated) is called a statement. These concepts are sufficient to represent data structures that can be shown topologically as a tree. In a tree, each item can be represented as a node that belongs to one, and only one, parent item (node). Tree structures can be mapped into pages and trains so that the fields encountered in a regular tree-coursing path (which exhausts all branches systematically, left-to-right) produce contiguous fields in the train-address space. For example, it is possible to have a sequential organization in which adjacent fields of each record are contiguously addressable in the train. This would apply to each record. If the record contains other embedded items, these too can be mapped contiguously in the address space.

Item structures can be defined and represented in several ways. Figure 5-5 shows four methods of representing item structures. A structure of four levels is defined in this example, which corresponds topologically to a rooted tree.

The <u>indented tree</u> has the most intuitive appeal and would be used in the initial design of the structure. In the rooted tree, each increase in depth level is indicated by indenting to the right. Items connected to the same veritcal line are at the same level. Structures are indicated by boxes of three different shapes: rectangular for files; hexagonal for records; and oblate for statements. The name of each structure appears in the box, and the names of simple items (scalars, fields, or leaves of the tree) appear after a slash joined to the vertical line representing its parent structure. A letter and number in front of a field name indicate the character type and size of field (V signifies variable length).

(a) INDENTED TREE



(b) INDENTED OUTLINE

```
S;A

   S;B

      E6;C
      AV;D
   A20;E

   FV;F


      R

         E6;G
```

(c) ITEM LIST/TERM LIST

S3	A
S2	B
E6	C
AV	D
A20	E
FV	F
R1	
E6	G

Figure 5-5. Item Structure Representations

The indented outline closely follows the indented tree form, except that the box shapes are indicated by a character. This form can be keyed from the tree as input to the item definition job. The item list term list combination may be similar to the internal representation of the structure in the system directory. In this form, indentation is not used; instead, the size of statements and records is presented in terms of the number of sub-items.

Because the logical structure of all data items is defined in this structure definition (stored internally as a system table and described with the same convention), considerable leeway is afforded in the format of input data. In agreement with the structure definition, data may be keyed in strict field sequence with empty fields indicated by a slash, or data may be keyed without regard to field sequence if each out-of-sequence field is tagged with the appropriate name or identifying number. Blanks or leading zeros need not be keyed; the system can automatically justify data and can supply blanks or leading zeros. The system can also rearrange the data in proper sequence, filling missing fields with blanks where necessary.

5.2.2.4 Conclusion. When structure definitions are collected into one comprehensive directory for all items, a data pool or common data-base concept can be realized. An alphabetical listing, by item name, can direct the system to all occurrences of the item as structure definitions and as data. Thus information (both structure definitions and data) is available to users without prior knowledge of the existence of the data or its detailed structure. The user needs to know only the generic names of the data to be investigated. There is no need to approach the data on a file-by-file basis, not is the user limited in any way by the original structure definitions. Rather, a structure definition in a dialog mode is available so that the user may take full advantage of the system and its data. The lack of constraints at both ends of the process - data definition and data retrieval - provides a new measure of flexibility in data-management processes.

5.2.3 Anticipated Results

It is anticipated that the data management task of the data storage project will result in a set of data structure types and data services which will form the basis for a standard interface between users (and user programs) and a data management system for satisfying user needs over a wide range of user types. It is expected that the language and functional specifications of the DMS would be suitable for implementation either by NASA, manufacturers, or commercial software developers. The specification should be suitable for presentation to CODASYL, USASI, user groups, and others concerned with simplifying the effective use of computers, especially when a large common data base is to be provided for a heterogeneous mix of users. The common programming interface for data services which this provides should be an important factor in insuring the transferability of programs and data from one installation to another.

REFERENCES

Section 3

1. E. Shapiro, "Technologies for Storage Hierarchies," Preprints, International Federation of Information Processing Congress 71, Invited Papers, p. 46-51.

2. W. Anacker, "Possible Uses of Charge Transfer Devices and Magnetic Domain Devices in Memory Hierarchies," Intermag Conference, Denver, April 1971.

3. C.V. Ramamoorthy, "Architectural Considerations of Memory Hierarchies," NEREM 71 Record, Part 1, p. 98-101.