1963

Associative memory techniques for large data processors

William Rouse Smith
Iowa State University

ASSOCIATIVE MEMORY TECHNIQUES FOR LARGE DATA PROCESSORS

by

William Rouse Smith

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major Subject: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Head of Major Department

Signature was redacted for privacy.

Dean of Graduate College

Iowa State University
Of Science and Technology
Ames, Iowa

1963

TABLE OF CONTENTS

INTRODUCTION

Associative memory techniques using conventional, word-organized, destructive readout memories have been investigated. Organization based on long word lengths, hierarchal data structures, and variable length records allows for data storage and retrieval based on content rather than specific location. Special transformations are employed to allow conventional hardware to be used. Applications are surveyed and classified according to data structure, data volume, and speed requirements. Specific systems are proposed corresponding to two and multi-level data structures.

Problem Existing

It is generally accepted that conventional, directly-addressable memories are limited in speed and organization. The advantages of an associative memory become apparent when considering such applications as business data handling, language translation, list processing, matrix and numerical problems, sorting, and large memory bookkeeping. Considerable memory space is used in conventional memories for address bits and bookkeeping which becomes unnecessary in an associative memory.

Because many of the problems of associative and conventional storage are similar, most of the standard storage techniques are being explored in associative memory research. In addition, several new techniques are under consideration. Typical areas are cryogenics, magnetic films, ferrite cores, transistors, and software methods. Many of the techniques require nondestructive comparison of search criteria with stored words, identification of the matching word, and access to the word for readout. Although

this is easily achieved on a small scale, practical sized memories become very complex making economical manufacture difficult. Because of the difficulties no technique appears to dominate the field.

Accompanying the need for associative memories is the need to be able to store very large volumes of information. The range of memory sizes of current interest varies from 10^6 to 10^9 bits. For a memory as large as 10^9 bits associative techniques are essential.

Nomenclature

The term associative memory will be used in a general sense to describe any memory in which a data record is retrieved or identified by specifying the information content of an arbitrary portion of its structure. This definition includes memory systems in which data structure is utilized in storage and retrieval. Terms currently used with similar meaning are content-addressed, data-addressed, search, tag, and catalog memories.

The smallest complete unit of information is the record. The record is broken into groups of binary bits which are called catenae. A word is the smallest directly or indirectly addressable unit within the memory. Each word contains k catenae and each catena contains L bits. A catena is normally larger than a character. Catenae range from 10 to 30 bits long.

Each record is divided into description and data. The selection of a particular record in response to a retrieval request will be based on the description. The proportion of description to data varies with application. For example, the description may be a unique name of the

record, or, at the other extreme, the complete record may serve as the description in which any arbitrary bit pattern may be used as the basis for selection.

A list is an ordered group of catenae or records that share a common description. A list will normally possess a unique description which is called a name. A list may be a record, a group of records, a list of lists, etc.

Scope of Investigation

This is a report of a preliminary investigation into the uses of a conventional, word-organized, destructive readout memory as an associative memory. Several basic techniques including data transformations, word packing, and indirect word selection were investigated. Data structures for numerous applications were studied and classified. Several subsystems are specified and their operation compared to existing techniques. The associative memory subsystems serve to suggest a class of memory systems that could lead to practical manufacture and improved organization. The primitive operations of input, retrieval, and deletion are discussed. Due to the general nature of this study, the large number of variations, and the dependence on application a detailed evaluation was not practical. It is hoped that this investigation will lead to a more extensive study and eventually to a class of practical memory systems.

BACKGROUND

A brief survey of present methods of file addressing is given along with a discussion of data structures commonly encountered. A detailed example of a business problem is given for illustration. Several specialized associative memory systems have been proposed in the recent literature. These systems are described in as much as they apply to this study. Since large word-organized memories are necessary to implement the proposed scheme, the present state of the art is discussed.

Data Structures

Natural hierarchy exists in most data structures. This classification can be utilized when organizing a memory system (1, 2, 3, 4). For example, in a military personnel file the levels might be associated with assignment such as regiment, company, squadron, etc. The same file could also be organized according to rank or job classification. In a data processing program the levels of main program, subroutines, subroutines within subroutines, macroscopic operations, and primitive operations exist. In a theorem-proving problem Newell and Shaw (5) indicate that 12 levels of hierarchy are typical.

Artificial hierarchy can be used to break down a file. A dictionary can be organized according to the first letter, second letter, etc. (6). A document file can be preprocessed to allow block rejection when handling a retrieval request (3). When a data file is broken down into hierarchal levels, the result is normally some form of a tree. Tree selection techniques are often utilized either directly or indirectly.

Description coding techniques are important from the standpoint of controlling redundancy and increasing system efficiency. Consider briefly the problem of coding descriptors in a document retrieval application. Each document has an arbitrary number of descriptors which are normally of the same descriptor class. If the English language terms are stored sequentially into memory for each document, the task of retrieval can become impractical. Goldberg and Green suggest using a code book to reduce the redundancy in storing each descriptor and a superposition technique to combine the descriptors into a single field (7, 8). The type of retrieval must also be considered when coding a description. Prywes discusses a system in which there are several unique descriptor classes per item (9). A retrieval request can also be in the form of a logical statement (10).

Unique and nonunique retrieval responses are encountered. In a list processor only one list is associated with a particular name. In a business problem a possible request is to locate the records of all people living in city A, within income bracket B, and with occupation C. There may be several, one, or no responses to a request of this type.

Conventional Storage and Retrieval Methods

Description

Serial scanning of a file of information is perhaps the most common searching method for conventional memories. Serial searching involves examining sequentially each record in a file until the desired record is found. It is useful for small, short lived, or serial access files. It

can be used for both numeric and symbolic data. The records are normally unordered so that if there are N records the average number of accesses required for retrieval is N/2. This can be improved substantially if the information is arranged according to frequency of usage. The usefulness of this method for secondary searching must not be overlooked when considering more sophisticated methods. King (11) describes a method whereby different levels of serial scanning are used in a dictionary.

The retrieval time can be reduced considerably if the file is ordered. The logarithmic or binary search (12) reduces the maximum number of accesses to $\log_2 N$. With the binary search the file is first divided into two parts. By interrogating the record that separates the two parts it is determined whether the desired record lies in the first or the second part. The selected section is again divided and the part containing the record selected. This continues until the record is found. If this technique is used to locate a word in Webster's Unabridged Dictionary, a maximum of about $\log_2 10^6$ or 20 words will be interrogated. The average retrieval time is normally very close to the maximum. The primary disadvantage of this technique is that the file must be ordered and additions and deletions are awkward.

Tree selection schemes for symbolic data are useful for rapid access to a file (6, 13, 14). Each node in the tree may have two or more branches. Each branch may reference another node or a data record. The referencing is accomplished with chaining addresses. The number of accesses necessary to retrieve a particular item is equal to the number of levels in the route taken through the tree. If there are more than two

branches per node, a special purpose computer is usually necessary (23). The efficiency of a tree is dependent on the number of branches per node and the balance of the tree. If the tree is expandable, effort may be required to keep the tree balanced. The memory space utilization for this method is diminished by the space required to store the chaining addresses.

Another method closely related to tree selection is the organization of the file into lists. This technique along with several tree selection schemes is based on the work of Newell and Shaw (5). With proper organization of the data, list processing can serve as an information storage and retrieval tool (9).

Open or randomized addressing methods are useful for rapid retrieval. A detailed analysis of this method is given by Peterson (15). The particular record address is obtained by a randomizing process from the record name with subsequent testing to determine if it is the desired address. This is an efficient method for memories that are less than about 90% full.

Example

Consider a personnel file for a large industrial company. Assume that the company has 10,000 employees. A file is composed of one record for each employee. Each record consists of 500 to 1000 bits organized into fields. Each field contains a descriptor belonging to a descriptor class. A common set of descriptor classes is name, address, payroll number, job classification, department, salary rate, length of service, sex, marital status, and number of dependents. A typical record corresponding to the above classification is John W. Jones, 2510 Ontario Road, Ames, Iowa, AB23542, Electrical Engineering, Research, $8500.00, 5, Male, Married, 7.

John W. Jones and AB23542 are unique descriptors since no other employee in the company would have the same name and payroll number. Other descriptors such as research or male are associated with many records.

A file of this type may be ordered on some unique descriptor class such as name. The descriptor on which the file is ordered is usually called a primary descriptor. In searching for the records associated with John W. Jones the binary search could be used to locate the item in less than $\log_2 10{,}000$ or 14 memory cycles using a random access memory. If it were desired to find the employee or employees living at 2510 Ontario Road, a serial search of all records would be required or 10,000 memory cycles.

A tree structure could be used to reduce the number of memory cycles required for retrieval. For example, at the first level divide the records into groups according to department. Within each department group divide the records into subgroups according to job classification. Additional levels can then be used to break down the file. Suppose a request such as to retrieve the records of all electrical engineers in research living in Ames, Iowa. The subgroup corresponding to research and electrical engineers would be selected in two memory cycles. Each item in the subgroup would be searched sequentially until all living in Ames, Iowa were located. Considerably less than 10,000 memory cycles are required using a tree.

The file may also be organized into lists. Suppose the records were unordered and stored in a large random access file. The department descriptor class may have ten exclusive descriptors. The descriptor, research, is selected by some means. With the descriptor, research, an address is

placed that refers to the first record of a list of records of persons assigned to research. The list is formed by tying each item of the list together with chaining addresses. The last item on the list has an end-of-list code. Note that the lists are formed without rearranging the records in memory. Other lists are formed with other descriptor classes so that more than one descriptor class can be referred to as a primary descriptor class. Although selection based on one descriptor can be achieved rapidly, additional selection based on another descriptor requires a serial search of all items on a single list. For example, a search for all secretaries assigned to research would require a serial search of either the research or the secretary list.

Open addressing is useful when selecting descriptors in a single field. Suppose it is desired to retrieve each personnel record on the basis of name alone. Each name is a highly redundant code describing each record. If the name, alphanumeric code, is processed by a method such as multiplying by π and selecting least significant bits, a uniformly distributed set of codes is obtained. The codes are then used as addresses for the records. When retrieving the record the same process is used thus giving the same address. The complete name is stored in memory so that a check can be made each time before accepting the information. If two or more records have the same randomized code, a special algorithm is used such as storing the records in sequential unused memory locations. This method gives very rapid retrieval when only one descriptor is used.

Each of these file addressing methods is limited either by the number of memory accesses required for retrieval or by the difficulty of selection

using secondary descriptors. Associative memory techniques reduce these limitations.

Associative Memories

The primary function of an associative memory is to recognize the existence of a word or bit pattern in a memory and to select and retrieve it and any information associated with it on the basis of a key bit pattern. This recognition can be accomplished in parallel or by some other direct method.

A fully parallel or simultaneous interrogation of all words in memory is proposed by Goldberg (7). Physical limitations such as signal to noise ratio often require a parallel-by-word, serial-by-bit search as employed by Kiseda et al. (16). By utilizing more hierarchal information the memory can be broken into blocks to accomplish interrogation.

Conventional systems have been organized to incorporate associational properties. The work of Newell and Shaw (5) associates information with the use of chaining addresses. Conventional list techniques are limited in speed and space utilization. Prywes and Gray (9, 17) have developed a technique for retrieval from very large files using conventional equipment.

The semi-parallel interrogation of a description region is not limited to an identity match. Functions such as greater-than, less-than, between-limits, maximum, minimum, and mixed-search can be implemented (7, 8, 18, 19). The result of such interrogations can result in no match, a single match, or in multiple matches.

Magnetic approaches

Two ferrite cores per bit are used by McDermid and Peterson (20) and Kisead et al. (16) to perform the equivalence decision directly in memory. Comparison is accomplished serial-by-bit, parallel-by-word. Detection is accomplished by either a blocking oscillator or a multi-aperture core. Comparison and mask registers allow any or all bit positions to be examined. The memory is word organized. Match indications are encoded to XY data for readout through XY rather than word drivers. The word address can be a byproduct of retrieval.

Several memories have been designed to use transfluxor multi-aperature cores (21). Two systems have been built each having 1024 words with about 36 bits per word. Typical read and write cycle times are about 6 μsec. In each of these systems only one core per bit is required.

A semi-permanent magnetic memory is proposed by Goldberg and Green (7) based on split-C linear cores. The system under development has 1100 words of 281 bits each. The system is to be used for document retrieval. The storage is not electrically alterable. This is thought to be the largest associative memory under construction (21). Interrogation to determine if a given item exists in memory is rapid, but the subsequent reading of the associated information from memory is in the order of milliseconds. When superimposed coding is utilized only the AND function need be implemented thus requiring only one core per bit.

Other related magnetic techniques are used in the system reported by Slotnick et al. (22). Related studies of distributed logic systems are reported by Lee (23).

A magnetic film bicore memory is reported by Petschauer and Turnquist (24). A memory of this type was used by Joseph and Kaplan (25). All words and bits of the 3000 bit memory are searched in parallel. The bicore element allows rapid nondestructive readout, NDRO, but has relatively slow writing speed.

Cryogenic approaches

One of the first associative type memories was proposed by Slade and McMahon (26). This is a catalog memory system in which a word is recognized as being contained in the memory but cannot be read out. The system is based on cryotron storage. Each bit has ten cryotrons to perform the write, store, and interrogate functions. A thin-film version of the catalog memory has been reported (27) making the fabrication of such a large number of elements feasible.

Complex logic at the bit level produces manufacturing difficulties even though control circuitry is simplified.

An associative self-sorting memory was proposed by Seeber (19) in which data is sorted during input. This was later improved (28) by generating an interrogation tag that eventually brings data out of the memory in ordered form. These schemes also used complex bit logic with a relatively large number of interconnections.

Several other cryogenic memories have been proposed to perform specialized functions (8, 18, 29, 30, 31). Large cryogenic memories have severe disadvantages at the present time. Refrigerator costs are high, multiple mask manufacturing processes are expensive, and the extremely low impedances of cryotrons lead to inefficient energy transfer. Also

substrate to substrate interconnections pose many problems. As the thin-film technology advances, cryogenic memories may very well become practical due to the extremely low cost of the raw materials.

Other approaches

Techniques such as tunnel-diode scratch-pad memories, integrated transistor storage, woven screen magnetic memories, and several mechanical techniques have been mentioned (7, 21). Each of these methods is proposed to avoid some of the inherent problems of cryogenic and magnetic methods. Cores become expensive to wire when complicated logic is associated in the memory and when reading must be NDRO. No technique appears to dominate the field.

A special memory organization has been reported by Mullery et al. (4). Associative information related to each word of a conventional coincident-current core memory is placed in ten special core planes. Orthogonal read and write properties allow hierarchal information to be stored in the special core planes. As a result indirect addressing, dynamic storage allocation, and list manipulation are realized.

Large Word-Organized Memories

The majority of the memories presently in use range in size from about 10^5 bits to 3×10^6 bits with access times ranging from about 0.8 to 20 μsec. The typical word length is 72 bits. As the need for larger and faster random-access memories increases, effort is being directed toward thin magnetic film and cryogenic techniques (32). Fast random-access memories with capacities ranging up to 10^9 bits are needed in

several areas, e.g. language translation and document retrieval.

Memories of primary interest to this study are of the type described by Pohm et al. (33) and Probster (34). The long word lengths described by Pohm et al. are particularly applicable. A typical memory is proposed with 10^6 bits organized into 2000 words each having 500 bits. The memory uses a conventional DRO mode with a cycle time predicted of about 100 nsec.

A magnetic disk file with a 2×10^8 bit capacity but with relatively slow access time can also be organized into the long word structure (35). If one complete track is taken as one machine word or data unit, then many of the techniques described apply.

ASSOCIATIVE MEMORY TECHNIQUES

Several supporting topics were studied and later used in specific subsystem examples. These topics pertain primarily to indirect addressing, variable-length data handling, and long word-length memories.

Transformations

Transformation or regrouping of information files is useful in simplifying the logic required in a memory system. Those transformations directly applying to this study will be discussed. A discussion of other useful transformations is found in Goldberg (8).

A bit-to-word transformation is basic to this study. For example, suppose there are m records each with n_1 data bits and n_2 descriptor bits. If the descriptor bits are abstracted from the records, then two memory blocks can be formed. The data block contains mn_1 bits, and the descriptor block contains mn_2 bits. The descriptor block is then rotated by 90° as shown in Fig. 1. If the information is stored into a memory with a single read axis, a particular bit position of the d- -iption block will be associated with a specific word in the data :. It is not necessary that the descriptors and data be stored in t me memory. A sufficient but not necessary condition is for m and n_1 to be related by some integral multiple for efficient placement into a rectangular memory array.

Catena-to-word transformations are similar to bit-to-word transformations. Normally a word is broken up into k catenae each containing L bits. If one catena is used to describe each word, then each of the catena in a descriptor word can be associated with a particular word. See

Fig. 1. Bit-to-word transformation with the word oriented horizontally

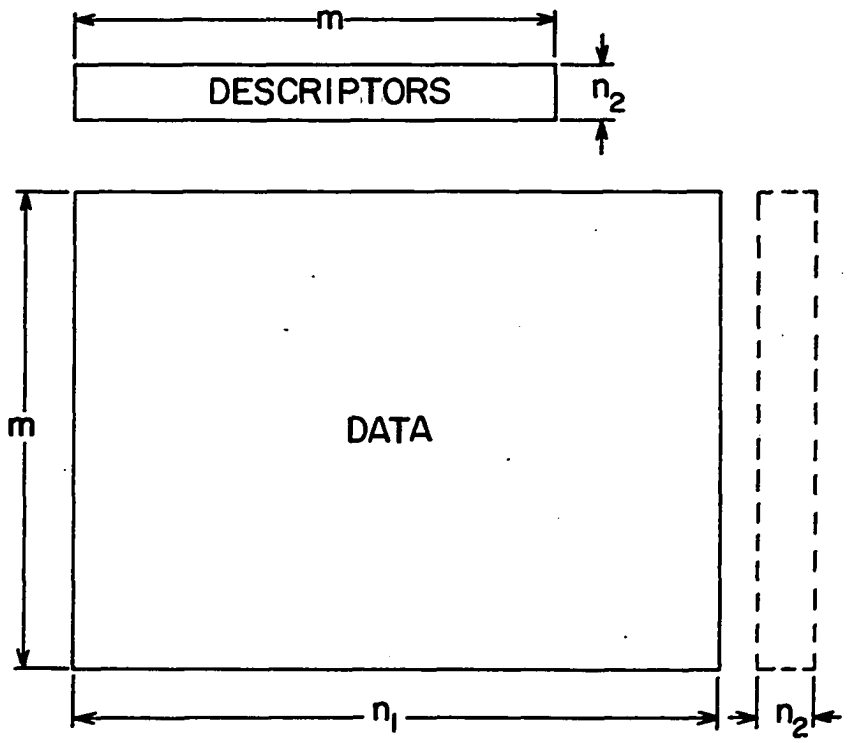

m
DESCRIPTORS
n_2
m
DATA
n_1
n_2

Fig. 2. Note that the descriptor catenae have been transformed from a two-dimensional array to a one-dimensional string.

Catena rotations are occasionally useful in connection with the descriptor block in Fig. 1. If there are L catenae with L bits each in a square matrix, it is often useful to interchange rows for columns. In the more common situation where the bit array is not square it is necessary to divide the pattern into squares or strips as in Fig. 3. Note that by combining a bit-to-word transformation with a catena rotation a set of catena-to-word mappings is obtained.

Regrouping of information is useful in simplifying hardware. For example, if a flag bit were used with each catena in a word for bookkeeping, regrouping would allow simplification of an output register. The associated flag bits can be separated from the data bits and placed at the end of the word, in another word, or in another memory. They can be rotated to facilitate addressing portions of a word in a scratch-pad memory.

Inversion or partial complementation of information is useful. If inverting logic is used, savings are sometimes realized by inverting every other bit in a word. When comparing information inversion of some data will allow, for example, the exclusive-OR decision to be used in place of the equivalence decision.

In cases with highly irregular lengths of records transformations that are not 1-to-1 are sometimes necessary. Chaining addresses, limited serial searching, and block rejection are useful in associating irregular data structures.

Fig. 2. Catena-to-word transformation

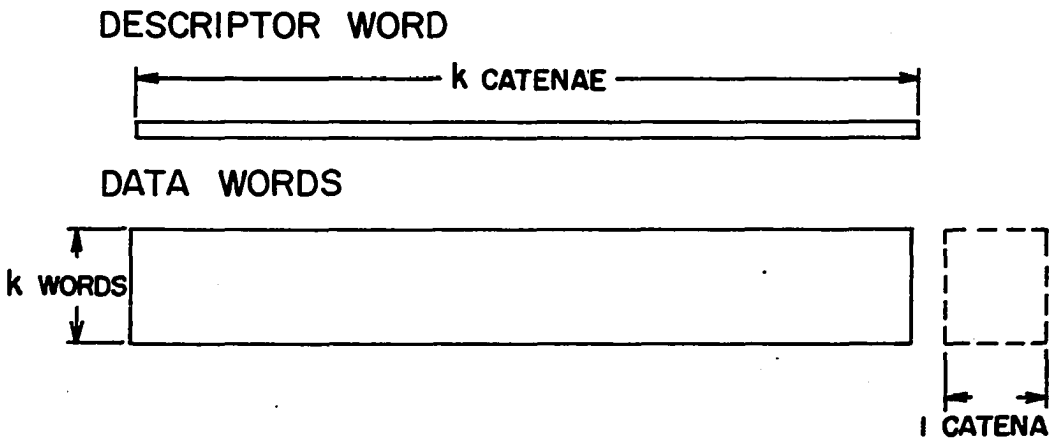
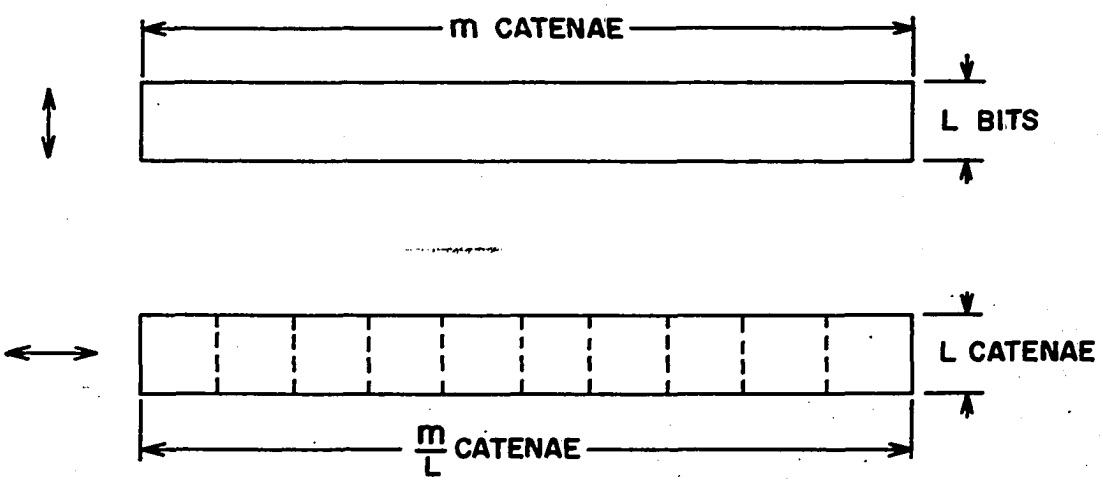

DESCRIPTOR WORD
k CATENAE
DATA WORDS
k WORDS
1 CATENA

Fig. 3. Catenae rotations with arrows indicating the catena orientation


m CATENAE
L BITS
L CATENAE
m/L CATENAE

Variable Word Length Data

Many applications require facilities to handle dispersion in record lengths. Word packing becomes necessary for satisfactory memory utilization when the average record length is of the same order or smaller than the word length.

One method to accomplish word packing is by providing more than one descriptor block per data block. The records are then stored sequentially by catena into the data words. A flag bit is associated with each catena to signify end of records. Facilities are provided to allow records to be split and placed in one or more words. A dead space is provided so that a record will not be started if there is only a small fraction of a word available.

Memory utilization efficiency

The memory utilization efficiency was studied on the basis of a particular storage algorithm. See Fig. 4. The number of descriptor levels, α, and the dead space, μ, are the variable parameters of the system. The algorithm allows for sequential storage and retrieval of records with the facility for automatic assignment of memory space. The process attempts to store all parts of a given record in the same level. Fig. 5 shows a possible level profile for a system with $\alpha = 3$. The machine word length is assumed fixed and the nature of the record lengths is determined by the particular application. The storage efficiency is defined as

Fig. 4. Input algorithm

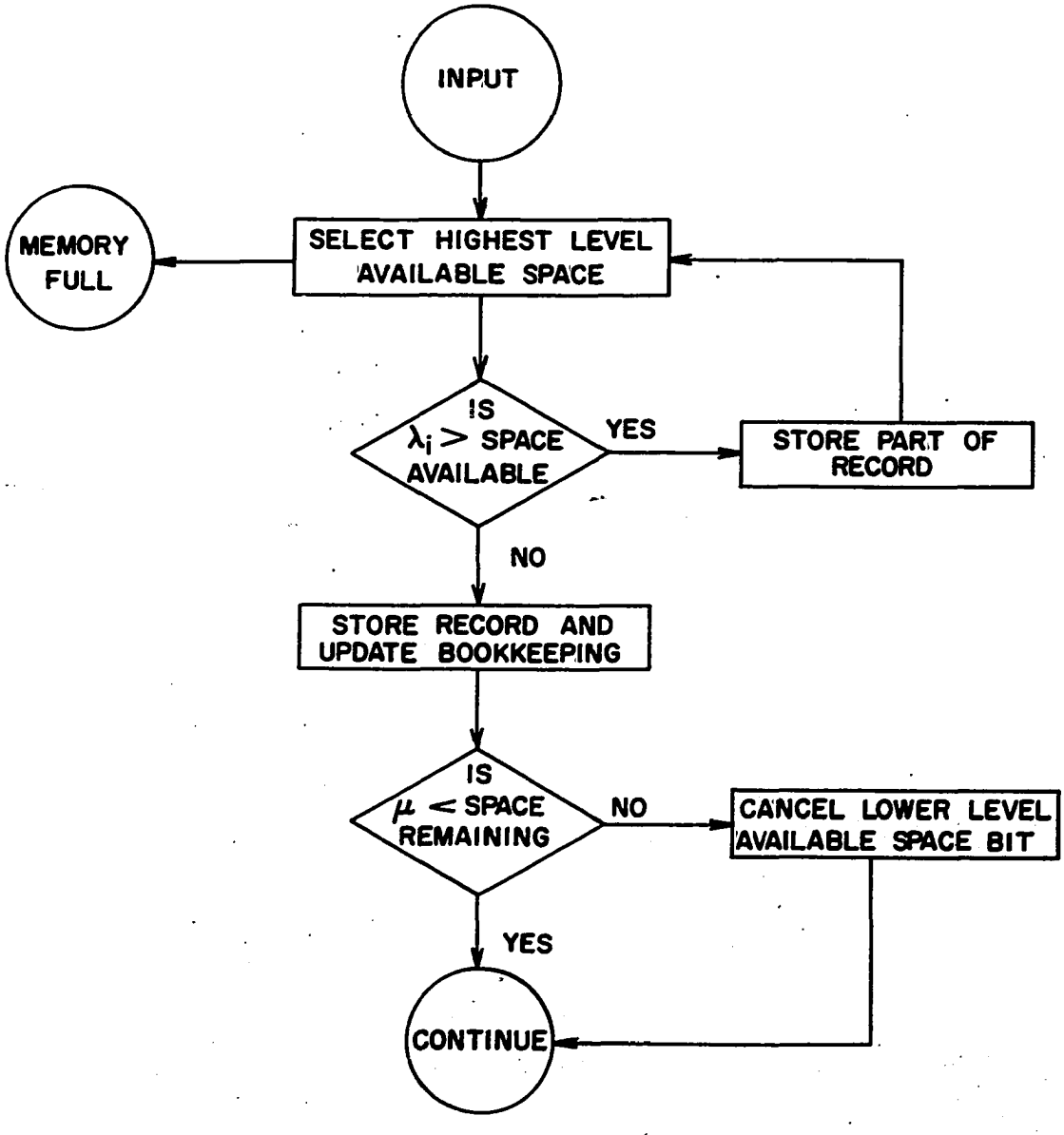
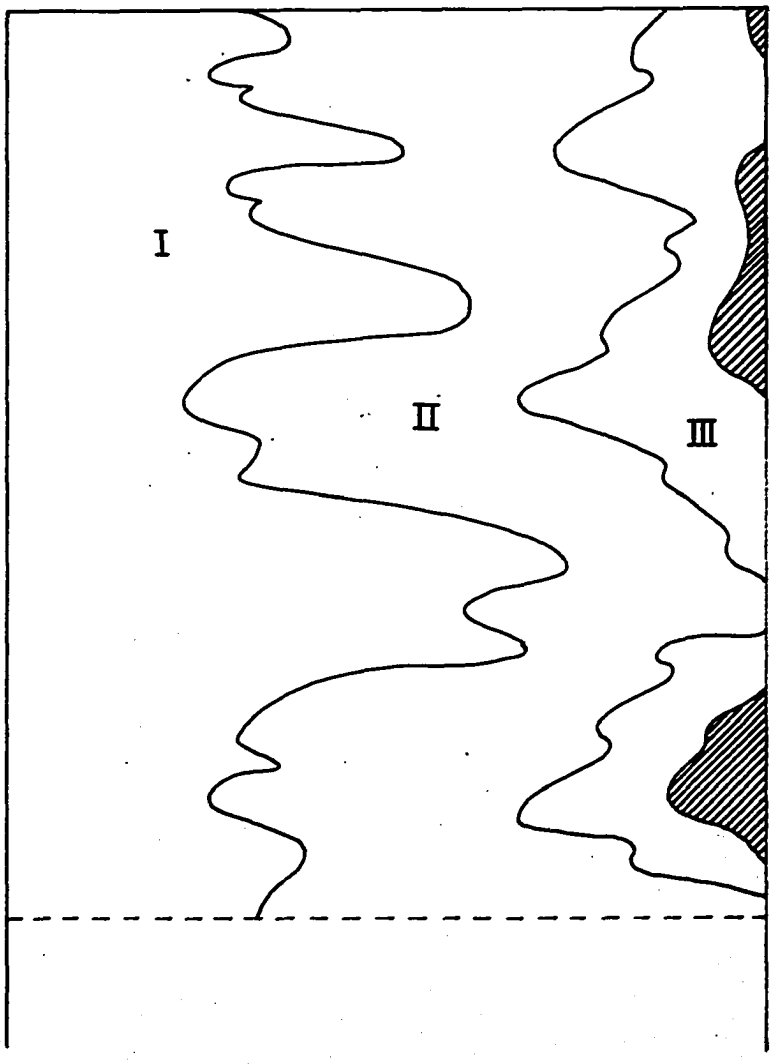

INPUT
SELECT HIGHEST LEVEL AVAILABLE SPACE
MEMORY FULL
IS $\lambda_i >$ SPACE AVAILABLE
YES
STORE PART OF RECORD
NO
STORE RECORD AND UPDATE BOOKKEEPING
IS $\mu <$ SPACE REMAINING
NO
CANCEL LOWER LEVEL AVAILABLE SPACE BIT
YES
CONTINUE

Fig. 5. Typical memory profile with variable length data and $\alpha = 3$ where the words are oriented horizontally, Roman numerals indicate record levels, and hashed areas represent unused memory space


I
II
III

$$E = \frac{\text{data stored + unduplicated descriptors}}{\text{data space available + bookkeeping space}} .$$

Memory space containing no information or redundant information can arise from (a) not filling a word when all descriptor levels have been used, (b) dead area at the end of a word that cannot be used for another record, (c) bookkeeping space such as available space identification, (d) duplicated descriptions since they contain no new information about the record, and (e) unused description areas arising from a full word with less than α records.

To determine the relationship between the data characteristics and the memory utilization factor a model was chosen. Typical values for memory parameters are M = 400, L = 20, k = 20, and N = 2500 where M is the word length, L is the number of bits per catena, k is the number of catenae per word, and N is the number of words in the memory. One bit from each catena is used as a flag bit to signify the end of a record. Each descriptor has 15 bits allowing for 32,000 possible descriptions. Another bit is used for each word per level for available space designation. The following calculations are based on this model.

Monte Carlo simulation

A Monte Carlo technique was used to simulate the storage of records with dispersion. A normal distribution of record lengths was approximated by adding three uniformly distributed numbers in the range +1 to -1 and then scaling to adjust the variance and the mean. A standard deviation of $\bar{\lambda}/4$ was chosen where $\bar{\lambda}$ is the mean. Approximately 1000 samples were used for each set of system parameters. The input algorithm was simulated by

monitoring pseudo data stored and keeping track of the amount of memory space available. A slight inaccuracy was introduced by using a continuous distribution of record lengths rather than a quantized distribution. The quantized distribution would exist when the information is stored into catenae.

The Monte Carlo simulation was performed on the Iowa State University Cyclone Computer.

Parameter effects

The efficiency, E, as a function of $\bar{\lambda}$ is given in Fig. 6 for $\alpha = 1$, 2, 3, and 4. The parameters of the model were assumed and the dead space was set at 10%. The record lengths were normalized with respect to a word length. A byproduct of the simulation gave the average number of memory words per record. See Fig. 7. Compare this with the staircase function obtained with $\alpha = 1$ and zero dispersion.

Since the model is somewhat arbitrary, the efficiency curves without bookkeeping effects are given in Fig. 8. Fig. 9 is a collection of curves that shows the effects of changing μ.

A memory system is often designed so that it is normally operating below capacity. By selecting available space starting with the highest level, i.e., $\alpha = 1$, the average number of words per record will be on the lower curve in Fig. 7. To observe the effect of below capacity operation note the differences between the curves of Fig. 8. With $\bar{\lambda}/M = 0.8$ the memory is 64% full before the second level is used. The average number of words per record, r, is 1.16 for the first level. With the memory up to 91% full the first two levels are used and r increases to 1.5. To

Fig. 6. Memory utilization efficiency curves for the word packing model

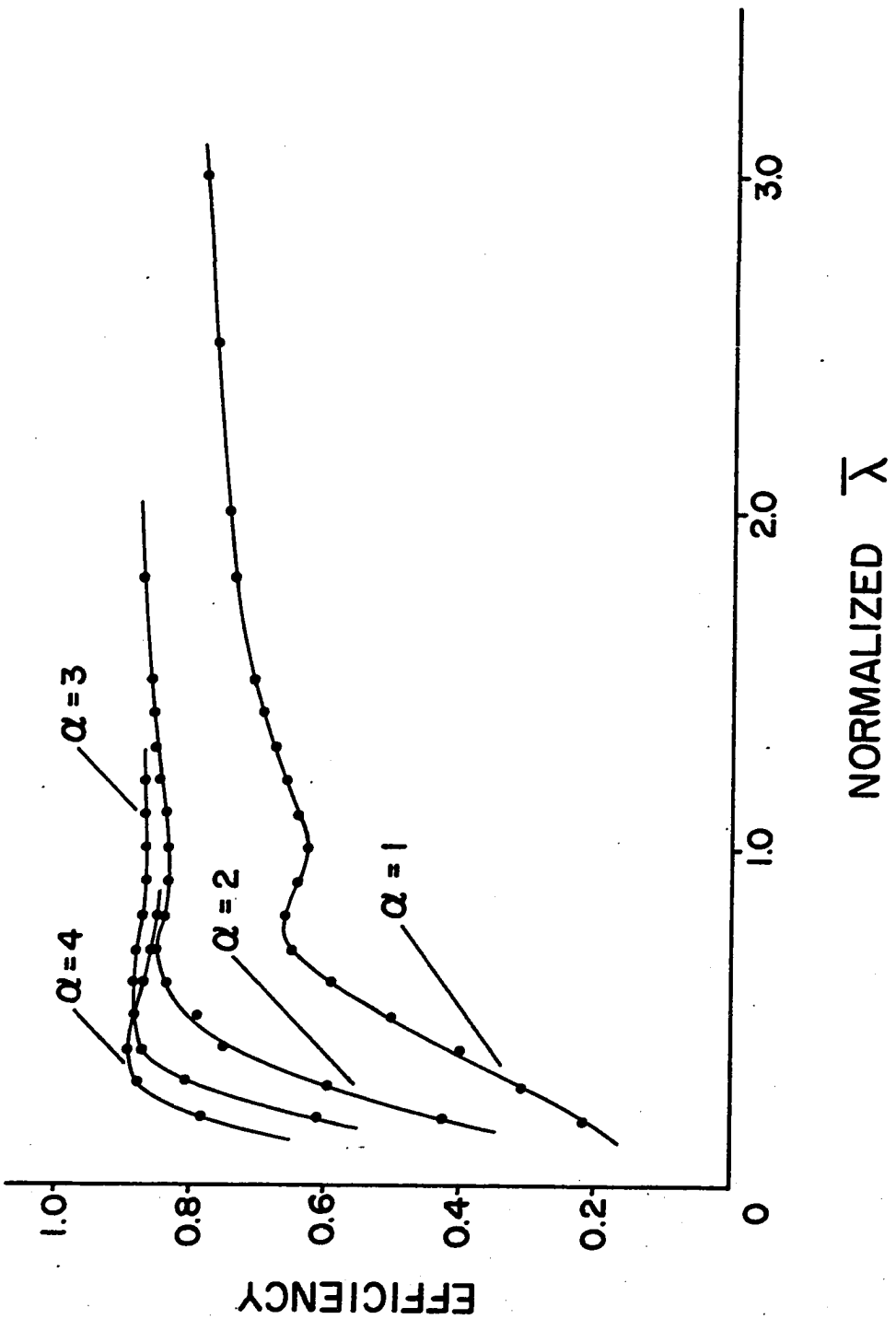

EFFICIENCY
1.0
0.8
0.6
0.4
0.2
0
1.0
2.0
3.0
NORMALIZED $\overline{\lambda}$
$\alpha = 4$
$\alpha = 3$
$\alpha = 2$
$\alpha = 1$

Fig. 7. Average number of memory words per record for the model

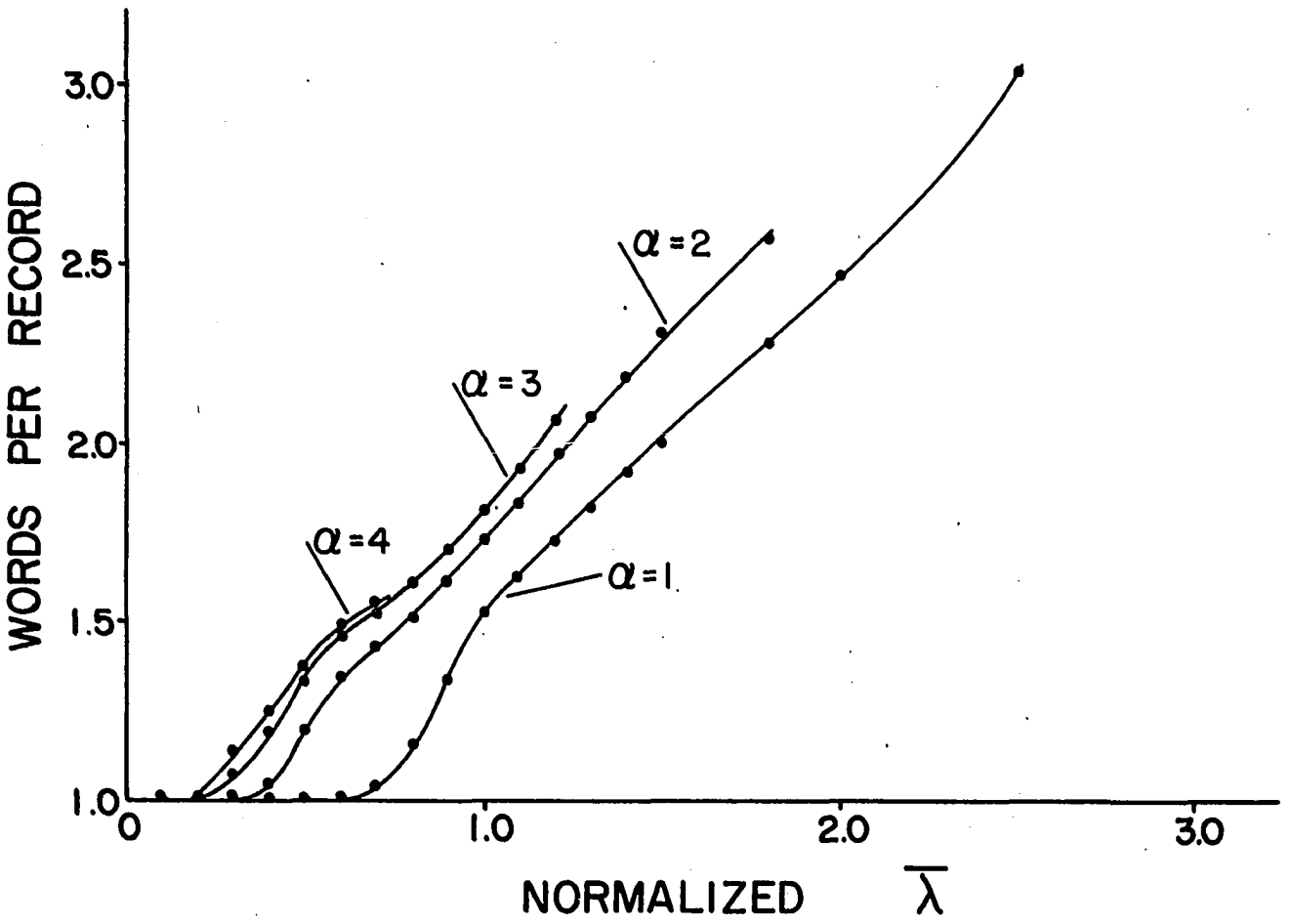

WORDS PER RECORD
3.0
2.5
2.0
1.5
1.0
0
1.0
2.0
3.0
NORMALIZED $\overline{\lambda}$
$\alpha = 2$
$\alpha = 3$
$\alpha = 4$
$\alpha = 1$

Fig. 8. Efficiency curves neglecting bookkeeping space

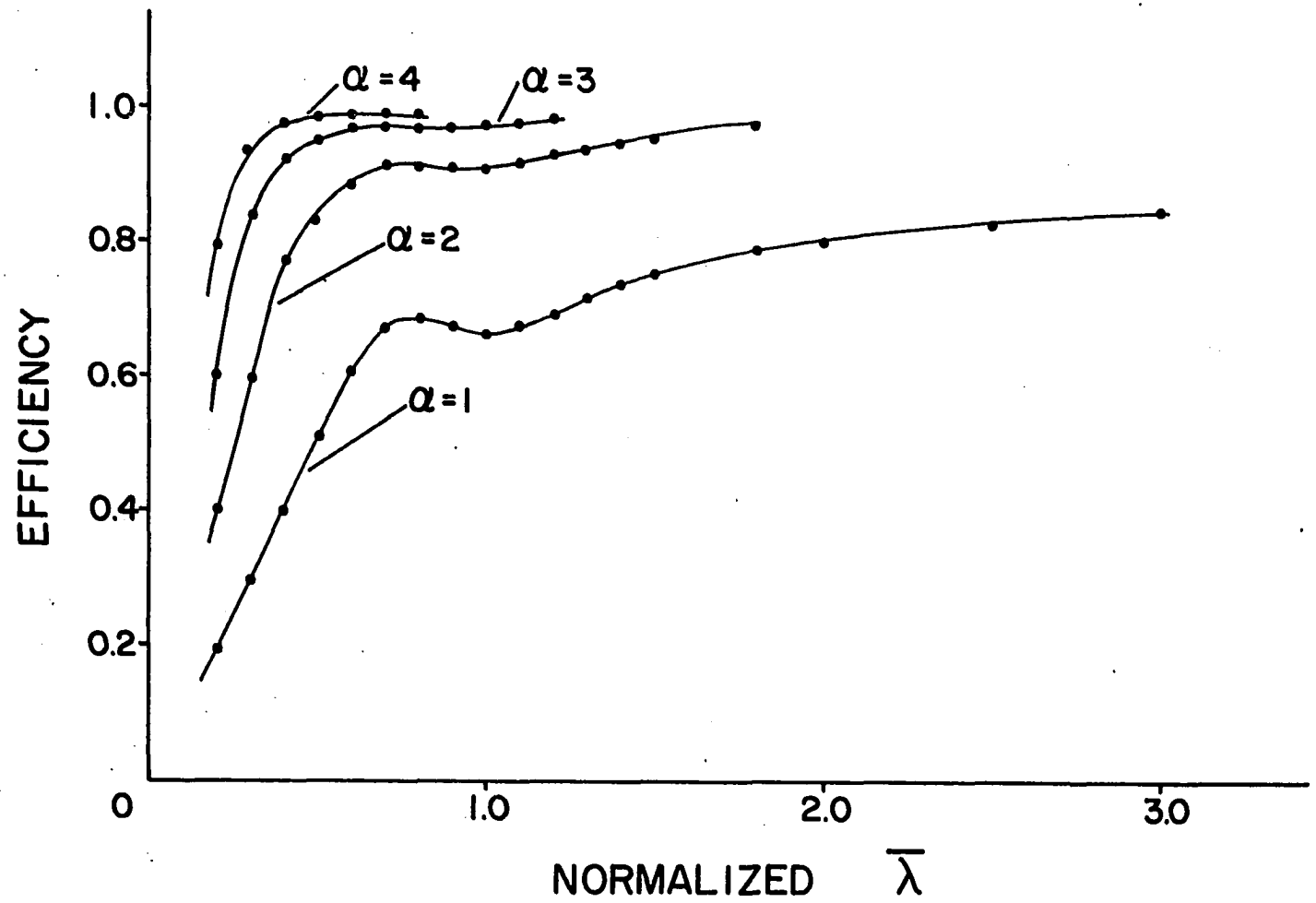
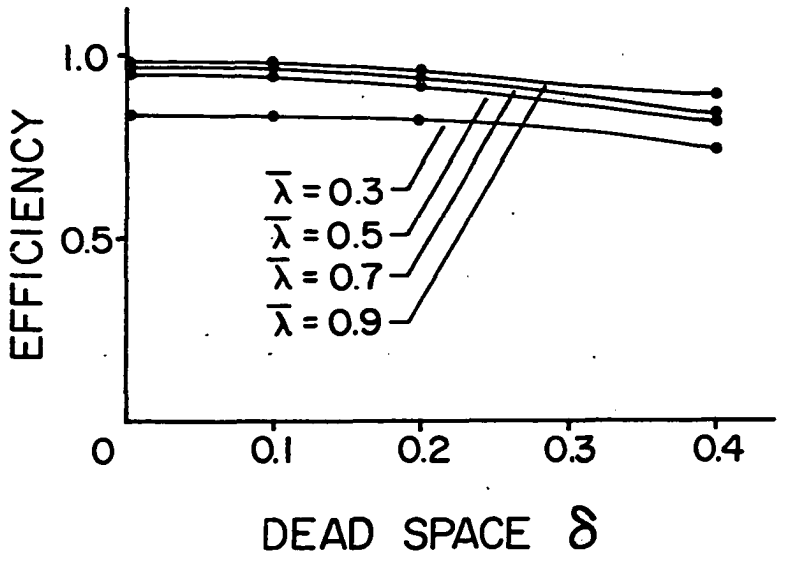
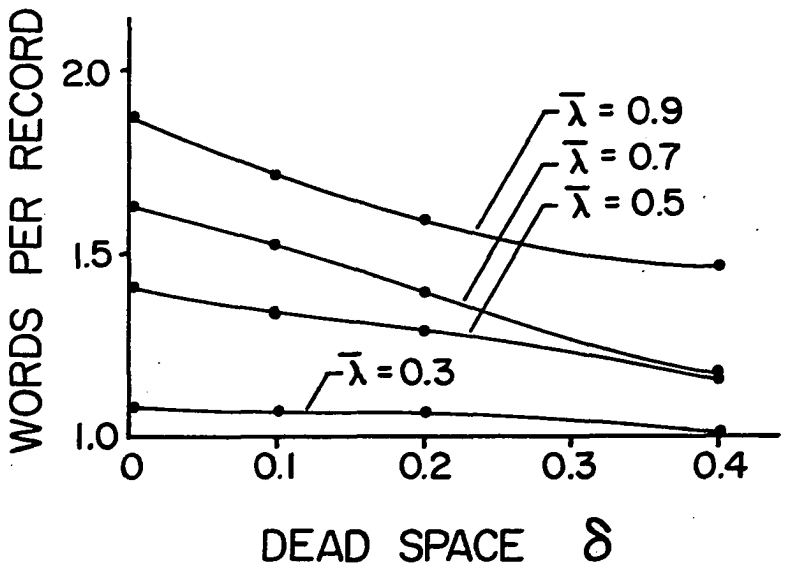

α = 4
α = 3
α = 2
α = 1
1.0
0.8
0.6
0.4
0.2
0
EFFICIENCY
1.0
2.0
3.0
NORMALIZED λ̄

Fig. 9. Curves showing the effects of dead space on the memory efficiency and the number of memory words per record when $\alpha = 3$


EFFICIENCY
1.0
0.5
0
0.1
0.2
0.3
0.4
$\bar{\lambda} = 0.3$
$\bar{\lambda} = 0.5$
$\bar{\lambda} = 0.7$
$\bar{\lambda} = 0.9$
DEAD SPACE δ


WORDS PER RECORD
2.0
1.5
1.0
0
0.1
0.2
0.3
0.4
$\bar{\lambda} = 0.9$
$\bar{\lambda} = 0.7$
$\bar{\lambda} = 0.5$
$\bar{\lambda} = 0.3$
DEAD SPACE δ

increase r from 1.16 to 1.5, r for the second level only is 2.3. Beyond 91% full more levels are utilized. For another example, choose $\bar{\lambda}/M$ = 0.5. For the first level r = 1.0, 1.5 for the second, and 2.4 for the third level. The implications of increasing r are not clear until a particular system is examined.

A comparison between this storage technique and several currently being used is given in the Appendix. Upper bounds on the memory utilization efficiency are given for several techniques and compared to upper and lower bounds for the long-word method.

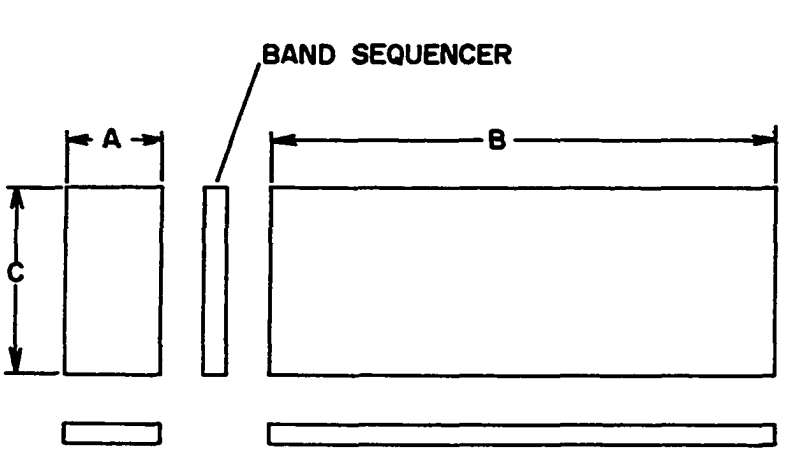
Word Selection

A diode or transistor word selection matrix is normally the last stage of selection in a word organized memory. Conventionally an address is broken into two parts, an X and a Y portion. The address is decoded and fed into the selection matrix. The outputs of a decoder represent, for example, a "1" on one line and a "0" on all other lines. In order to minimize the amount of hardware necessary in the decoders the X and Y addresses are equal in length. This results in a square selection matrix.

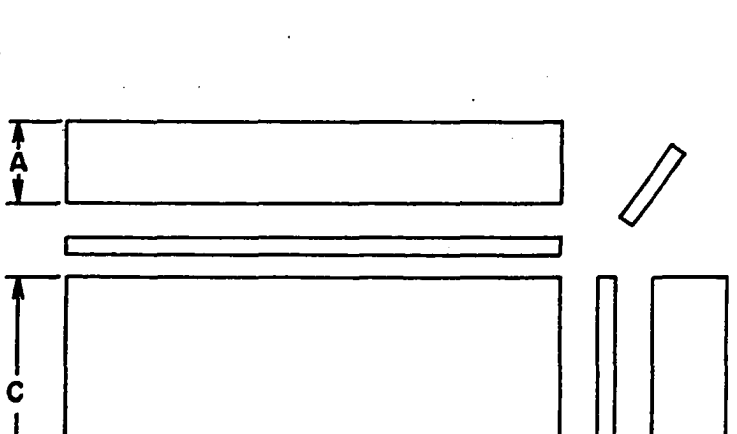
For the present application the address does not appear in coded form thus removing the need for a square selection matrix. For an example, assume that the memory is divided into various bands in which A words map into B words. In searching for information the A words are interrogated, the results of which are a decoded address containing B bits. If there are C bands, then band sequencing can be accomplished with a single register or ring counter regardless of whether the first or the second region is being referenced. In Fig. 10a the selection matrices for this

Fig. 10. Word selection methods, (a) two level system and (b) complex three level system

(a)


BAND SEQUENCER
A
B
C
DESCRIPTION SEQUENCER
DECODED ADDRESS

(b)


A
C
B

two level organization are given. The selection matrices can refer to the same or different memories.

Strobing of the selection matrix to activate the desired word line is normally required along only one axis. This is accomplished by using the shorter register thus allowing the long register to use simpler circuitry.

The technique of sharing registers for referencing different portions of the memory can be extended to higher level systems. Fig. 10b shows a three level system requiring only three input registers. The information appearing in a register originates from (a) decoding of an address, (b) indirect addressing derived from comparisons, (c) sequencing such as in a ring counter, or (d) carry propagation as used when unique addresses are not specified. The purpose of this illustration is to show the flexibility of complex word selection schemes.

Similar techniques are useful in manipulating a long word. For example, parallel logic at the 500 bit per word level is prohibitive. By transforming the one-dimensional output register into a two-dimensional array the catenae can be read out serially by using the flag bits to control a sequencing register.

Descriptor Organization

The description that is associated with each record or group of records can be organized in several ways. The following list describes types of retrieval and methods of effectively storing the description into memory.

A. All bits of a unique descriptor are compared with a search descriptor, and a single record is selected. This is used in applications

such as list processing and tree processing.

B. The description can be divided into multiple descriptors in cases where a record possesses several descriptor categories. Suppose there are three unique categories R, S, and T. A unique category refers to a descriptor class in which all bits must be compared even though more than one record can possess a particular descriptor. A retrieval request might consist of logical statements such as $R_i S_j T_k$, $(R_i + R_j)S_k$, $S_j \overline{T}_k$, or $R_i + S_j + T_k$. An example of a use of multiple descriptors is the description of a military personnel file where categories such as rank, age, and job classification are used.

C. A partial description can be used effectively in a manner similar to random or open addressing (15). For reasons of conserving space in the associative memory only a part of the description is stored in the description region. When several responses are received in response to a partial retrieval request, they are sequentially tested until the desired record is found. The complete description of each record would be stored in the data portion of memory. Limited serial scanning is often desirable in combination with a partial associative search. An example would be referencing a dictionary where the first two letters are placed in the description section and the remainder in the data store.

D. Characteristic descriptions are useful when particular characteristics of the record form the description. Examples of record characteristics are parity, bit distribution, initial character, and record length. This could be useful in character recognition, speech recognition, or radar tracking.

E. Partial comparison is involved with masking part of the description so that only a selected portion of the description is compared. This could be used with logic or best-fit retrieval.

F. User identification can be part of the description of a record in systems where several users store information into the same memory system. This might consist of one or more bits in the form of a code that is associated with each semi-independent problem.

G. Hierarchy descriptions are used by Mullery et al. (4). The rank and character of the record are recorded in the associative part of memory.

H. Superimposed descriptors are used by Goldberg and Green (12) when a record has several descriptors belonging to the same category. This is found in document retrieval problems.

ASSOCIATIVE MEMORY SUBSYSTEMS

Several subsystems are described that illustrate methods of implementing memory subsystems with associative properties. Numerical values are introduced to indicate orders of magnitude and to permit comparison with existing systems. These values are by no means considered optimum for any particular application unless specifically stated. The functions considered are input, retrieval, and deletion.

Two Level System

A two level system is any system in which the information is divided into two major groups or levels, namely description and data. In order to associate the organization with a particular application a large memory bookkeeping system is described. This serves as a model that can be adapted to several other applications.

Memory organization

A large random-access data memory, DM, with about 10^8 bits is to be controlled by a smaller associative memory, AM. The AM has about 10^6 bits. The DM is organized into long words of about 2000 bits each. The words are grouped into bands, each containing 500 words. The DM then has 100 bands with 10^6 bits each.

Each word has a name or description which is coded into 20 or more bits. The description organization is arbitrary depending on the application. See the section on description organization. The cycle time of the AM is small compared to the cycle time of the DM. Typical values for the memory cycle times are 100 nsec. for the AM and 10 to 100 μsec. for

the DM.

A general block diagram of the large memory bookkeeping system is given in Fig. 11. It is assumed that the description and available space records will occupy 20 bits. The transformation in Fig. 1 was used with $m = 500$, $n_1 = 2000$, and $n_2 = 20$.

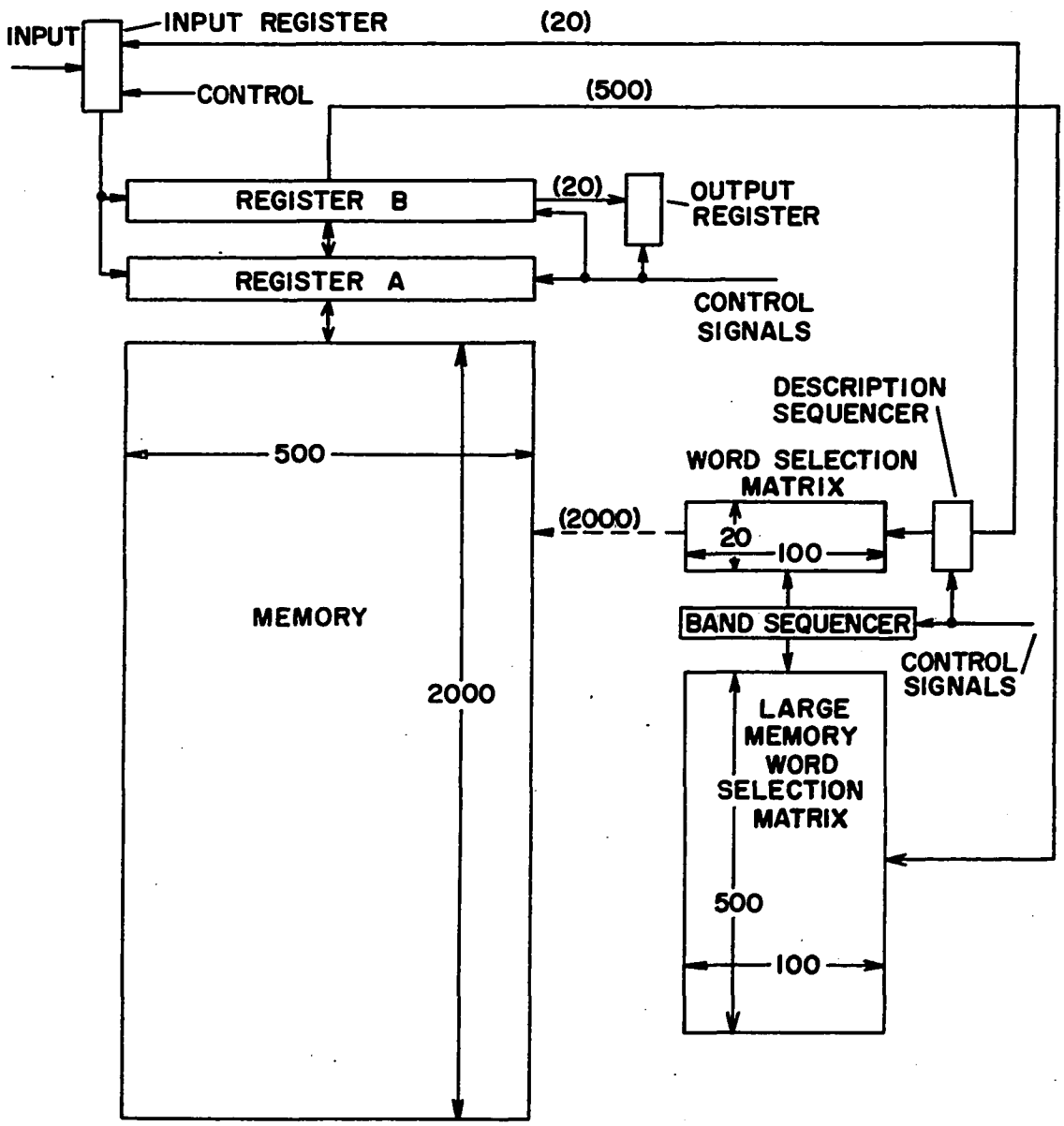
Automatic space allocation

One or two words in the AM are used to record the availability of each word in a band. Each of the bits in the available space words are associated directly with a word. The automatic space selection will be accomplished via the following sequence of operations:

1. Clear registers A and B.
2. Select a band on the basis of program information or predetermined sequencing.
3. Initiate a memory cycle in the associative memory with the available space word selected.
4. Initiate a carry propagation in register B.
   a. When the first "0" is encountered in register B, propagation will terminate, and one line of the DM word selection matrix will be active.
   b. If no "0's" are encountered, a signal will cause another band to be selected and transfer to step 2.

After completion of this cycle several functions occur to update the bookkeeping, store a description, and store the data. If the memory is full this is indicated after the band selector has completed a cycle.

Fig. 11. Large memory bookkeeping subsystem


INPUT
INPUT REGISTER
(20)
CONTROL
(500)
REGISTER B
(20)
OUTPUT REGISTER
REGISTER A
CONTROL SIGNALS
DESCRIPTION SEQUENCER
500
WORD SELECTION MATRIX
(2000)
20
100
MEMORY
BAND SEQUENCER
CONTROL SIGNALS
2000
LARGE MEMORY WORD SELECTION MATRIX
500
100

Input

An input signal implies that a record is available in the input buffer to be stored in the DM. The record appears serially by catena and has an arbitrary length. The description to be associated with the record appears in the input register of the AM. Depending on the nature of the data the word packing technique using multiple descriptions may or may not be needed. For the purposes of this discussion assume that there is only one description region per block, i.e., $\alpha = 1$. Automatic band sequencing is used with a ring counter for selection. For the present the counter is reset from one operation to the next. The input function consists of the following sequence of operations:

1. Perform the automatic space selection function.
2. Complete the AM cycle by changing the selected bit position from the available to the not-available designation and restoring the available space word.
3. Perform the store descriptor function as follows:
   a. Set the descriptor sequencer.
   b. Start a memory cycle and modify the bit signified by register B with the selected descriptor bit.
   c. Complete the memory cycle.
   d. Advance the descriptor sequencer and go back to step b until finished.
4. Load the input buffer of the DM.
5. Execute a DM storage cycle.
6. If complete record is stored, then continue; otherwise go back to step 1.

More complex input operations such as adding to or modifying an existing record will be discussed later. Other provisions such as redundant error checking may be added.

Retrieval

The retrieval response may consist of zero, one, or several records. See the section on description organization. The description will appear in the input register of the AM. The primary need for two output registers is evident when using a DRO memory for comparison. The retrieval command consists of the following primitive operations:

1. Select a band on the basis of program information or sequencing.
2. Reset registers A and B.
3. Execute an AM cycle.
4. Use exclusive-OR comparison to set register B.
5. Advance descriptor sequencer and return to step 3 unless done; then continue.
6. Initiate carry propagation in register B.
   a. When the first match, e.g., "0", is encountered in register B, propagation will terminate and one line of the DM word selection matrix will become active.
   b. If no matches are encountered, a signal will cause another band to be selected and control to be transferred to step 2.
7. Execute a data memory cycle. The memory subsystem will remain idle until the analysis is completed by the main

processor.

a. If the main processor detects that retrieval is not complete, control is returned to step 6 where propagation is allowed to restart just after the point where it stopped.

b. If the retrieval is complete, the subsystem is ready to process another command.

If a more sophisticated retrieval, such as logic retrieval, is used, step 5 becomes more elaborate.

Deletion

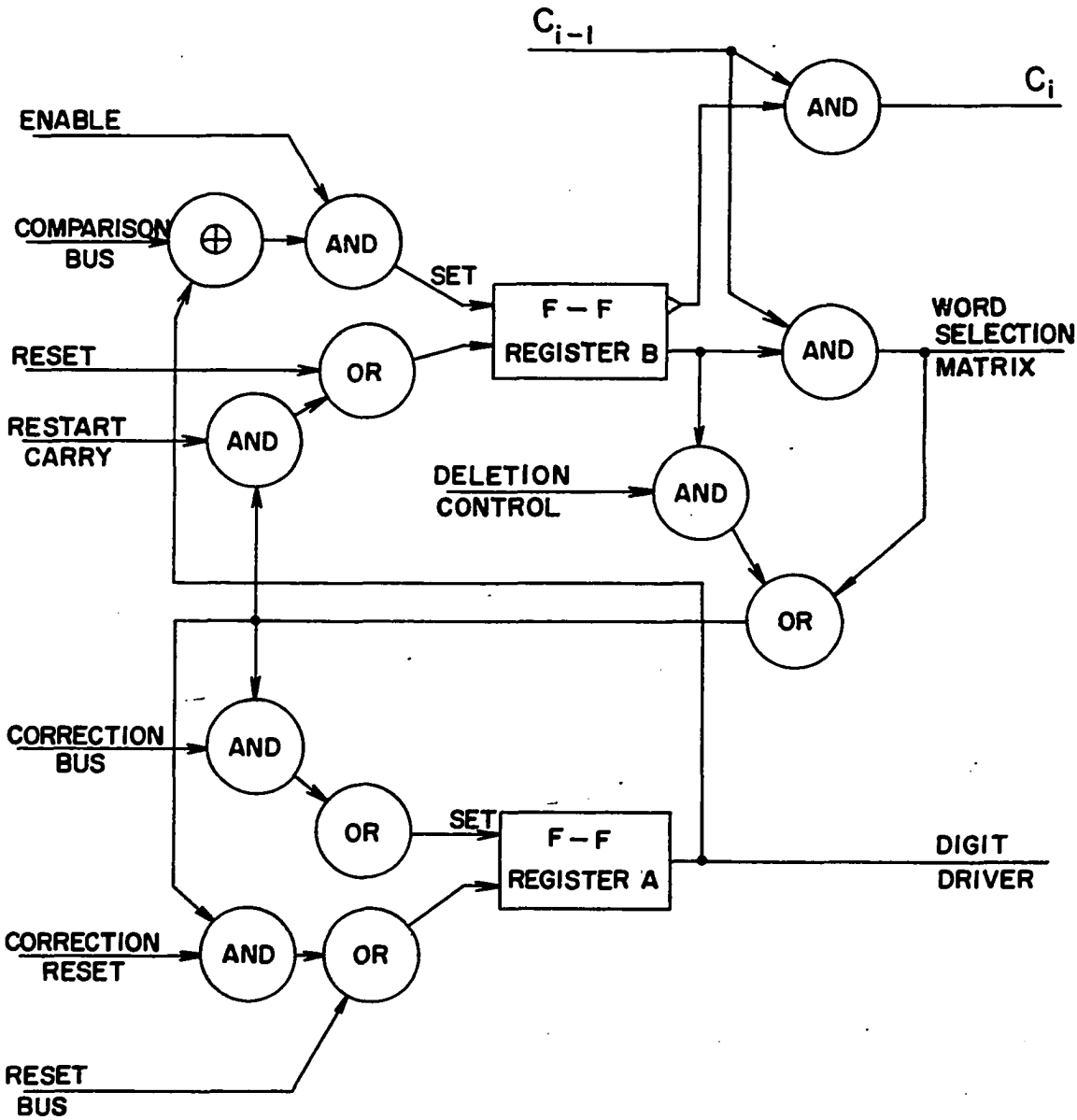
The deletion operation is started by a retrieval request unless register B already holds the desired information. After the proper locations are selected, the available space bits are modified to restore the space to usable storage. There is no need to null any of the description or data. It is possible to delete a number of records in the same band with a single memory cycle once the records have been identified.

Hardware requirements

In Fig. 11 it is observed that most of the logic circuitry is associated with registers A and B. The logic required per bit to perform the operations of input, retrieval, and deletion is given in Fig. 12. The control circuitry is not complex compared to the two output registers. By using high speed circuitry most operations can be performed during parts of the memory cycle. For example, correction, comparison, or deletion occurs between the read and write portions of the DRO memory cycle.

The input and output circuitry for the data memory consists of a

Fig. 12. Logic diagram for one bit position of registers A and B


C_{i-1}
AND
C_i
ENABLE
COMPARISON BUS
AND
SET
F – F REGISTER B
AND
WORD SELECTION MATRIX
RESET
OR
RESTART CARRY
AND
DELETION CONTROL
AND
OR
CORRECTION BUS
AND
OR
SET
F – F REGISTER A
DIGIT DRIVER
CORRECTION RESET
AND
OR
RESET BUS

minimum of logic at the word level. A single output register is used to convert the information from words to a serial chain of catenae. Most of the system processing is accomplished at the catena or subcatena level.

Variations

A perfect 10^8 bit random-access memory is expensive to fabricate. If a method were available to tolerate a few defects such as bad bits or words, the memory could be fabricated at a considerably reduced cost. Two methods are proposed to handle memory defects.

The first method uses a semi-permanent portion of the AM to store an additional available space word. This word contains a "1" in each bit position corresponding to a defective word. All other positions contain "0". After step 2 of the automatic space selection sequence the second available space word is combined by means of the AND function with the first available space word. By this method a defective word is never selected. In place of a semi-permanent store the defect information could be placed in the AM before starting each problem.

The need for the second available space word is not clear until the retrieval and deletion processes are examined. The problem arises when a record is deleted. It is desirable to delete a record by just changing the available space bit for that word from not available to available. In this case whenever a record is retrieved, the available space word is ANDed with the comparison output before the words are selected. If some description happened to appear in the region associated with a defective word, it could conceivably be selected. Also the defective location

could in some cases be restored to the available list.

The second method uses only one active available space word with a pre-stored defective word pattern. The description associated with a defective position is always null. The deletion process is also specifically designed never to disturb a defective word position.

The small associative memory is assumed perfect. Dynamic updating of the defective list is implemented by using check sums or some other means. Once an error is detected, a special diagnostic program could test the memory and update the defective list.

Multiple Level System

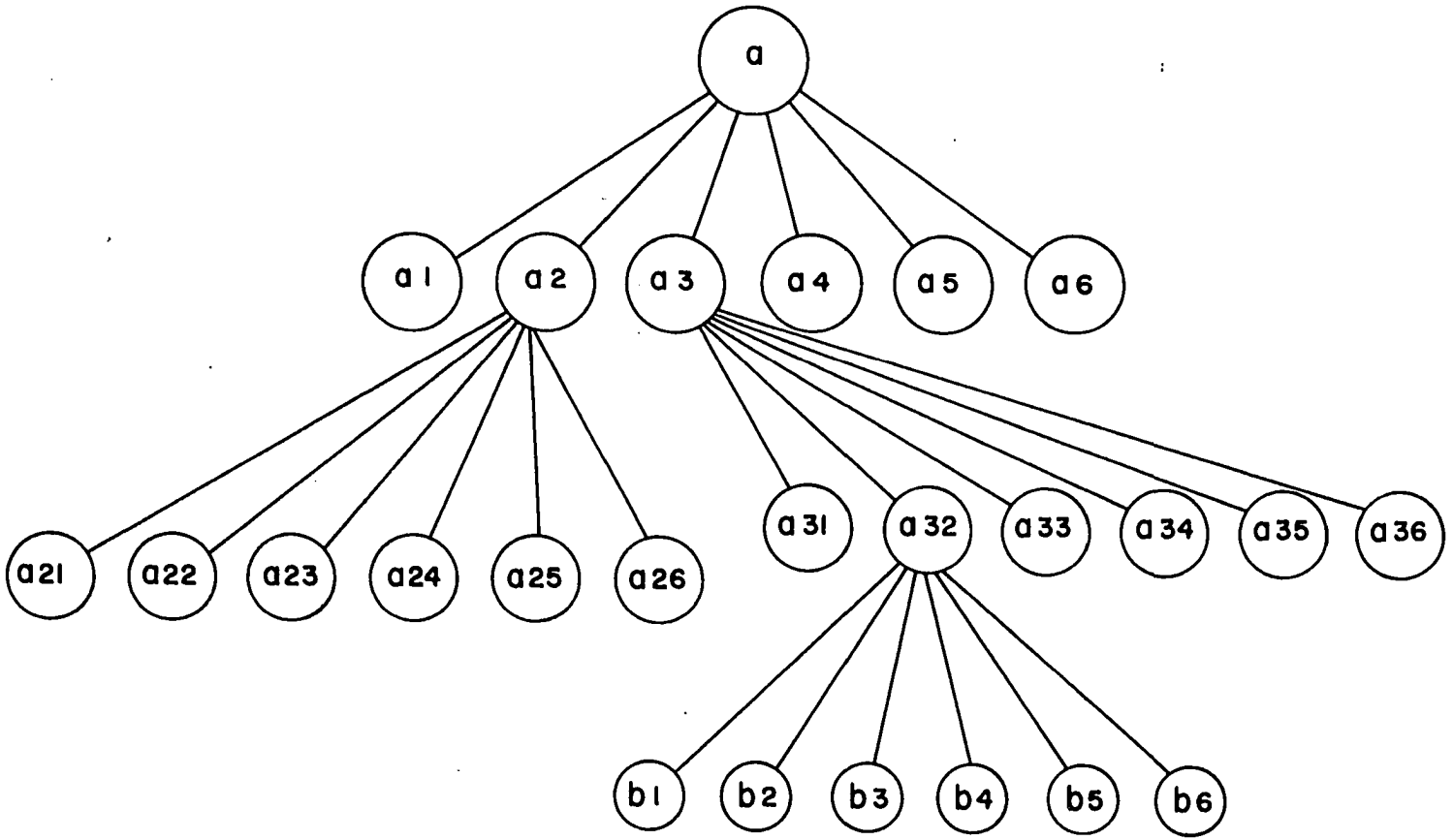
A multiple level system is one in which information is organized into hierarchy levels such as a tree. An illustrative example is based on implementing a multiple level tree using the catena-to-word transformation of Fig. 2.

Memory organization

A medium sized, random-access memory with about 10^6 bits is used for the storage of description and data. The word-organized, DRO memory typically has 2048 words 512 bits long. The memory array has the approximate proportions of four to one.

With k catenae per word one word is associated with k words. Each catena contains L bits. This serves as a node in a tree with k branches. In Fig. 13 an example is given of an unbalanced tree with up to four levels. In the example k = 6. In the proposed system k would typically be from 20 to 30. Modifications are described later which allow the

Fig. 13. Tree structure with k = 6


a
a1
a2
a3
a4
a5
a6
a21
a22
a23
a24
a25
a26
a31
a32
a33
a34
a35
a36
b1
b2
b3
b4
b5
b6

number of branches per node to vary.

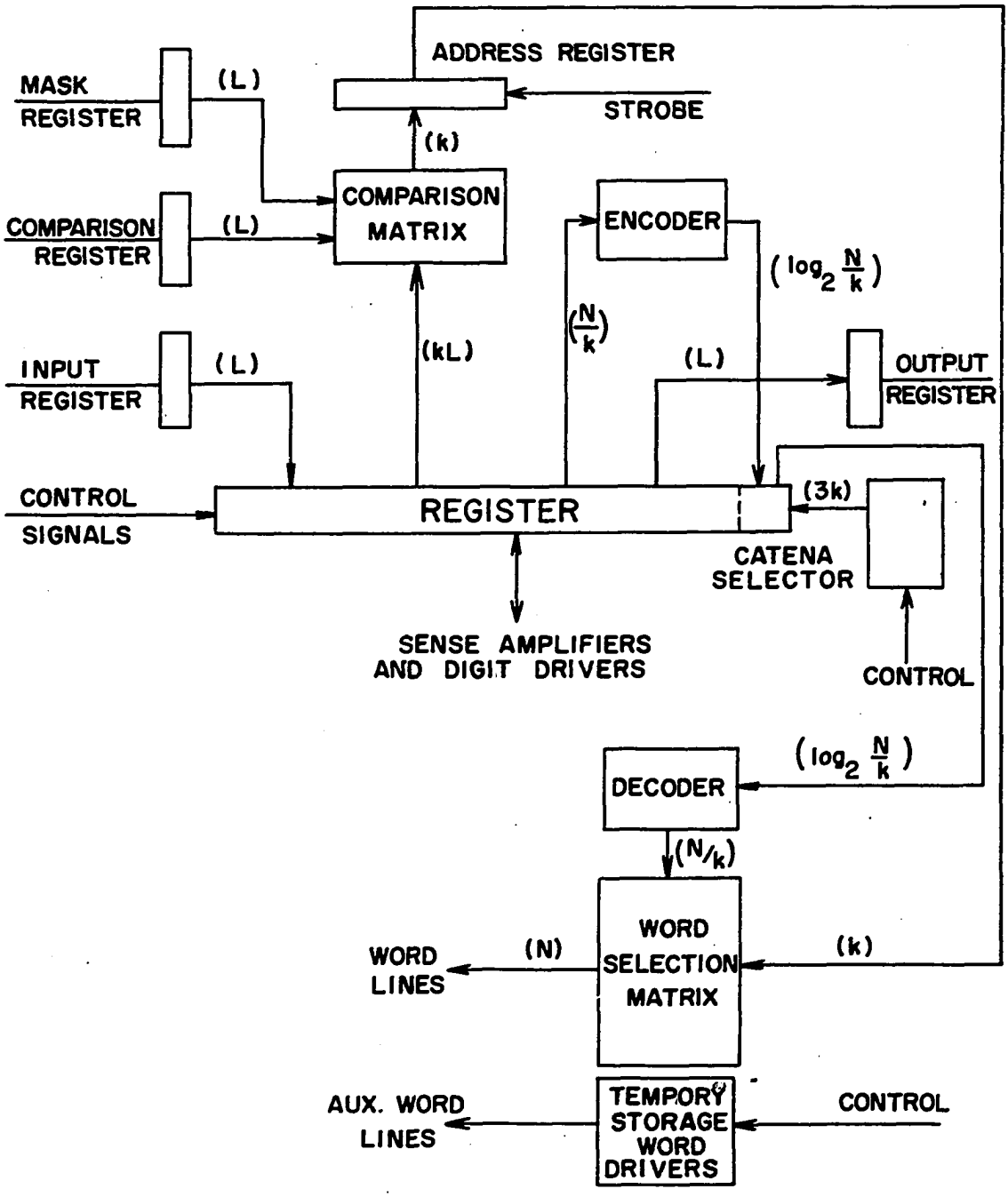
Each word contains k associative or data catenae, a transformation address, and bookkeeping bits. The number of bits per catena is governed primarily by the number of bits needed for a practical descriptor. In Fig. 13 a record has several descriptors associated with it to define the path through the tree, e.g., a, a3, a32, b3.

The tree organization of Fig. 13 is placed in memory as shown in Table 1. The data associated with the descriptors a, a3, a32, b3 is found using the following sequence. First, address i is referenced by some method such as random addressing on the basis of the descriptor a. As the information in address i is interrogated, a3 is found to be associated with the third word in the block starting with address j. Word j + 2 is interrogated by descriptor a32 to reference word n + 1 which in turn refers to word n + 8. The desired data appears in location n + 8. Flag bits are used to give the availability of each catena in a descriptor word and the end of record information in a data word. Each block of k words is associated with one of the reference addresses. In Table 1 the descriptors, data, and reference addresses are all that are actually recorded in memory. The reference descriptor is not stored with the word.

The descriptor word is differentiated from the data word by some flag information. A null reference address is one method of identifying data words.

A general block diagram of the multiple level system is shown in Fig. 14. The memory is divided into N/k bands where N is the number of words in memory. Each band is addressable via conventional address coding.

Fig. 14. Multiple level system block diagram


ADDRESS REGISTER
MASK REGISTER
(L)
STROBE
(k)
COMPARISON MATRIX
COMPARISON REGISTER
(L)
ENCODER
$(\log_2 \frac{N}{k})$
$(\frac{N}{k})$
(kL)
INPUT REGISTER
(L)
(L)
OUTPUT REGISTER
CONTROL SIGNALS
REGISTER
(3k)
CATENA SELECTOR
SENSE AMPLIFIERS AND DIGIT DRIVERS
CONTROL
$(\log_2 \frac{N}{k})$
DECODER
(N/k)
WORD LINES
(N)
WORD SELECTION MATRIX
(k)
AUX. WORD LINES
TEMPORY STORAGE WORD DRIVERS
CONTROL

Each word within a band is addressed indirectly as in the two level system. The details of Fig. 14 are discussed in the following sections.

Space allocation

Space is allocated on the basis of block and word availability. Block availability is associated with the establishing of a set of branches at a node. The memory is divided into N/k blocks of k words each. Each block is directly addressable while each word within a block is indirectly addressable. The last catena of each descriptor word contains the address of the block being referenced. The block availability is handled in a manner similar to that in the two level system. The selected available word is coded, stored in the description word, and decoded for future use in the word selection matrix. The available space words are stored in a special section of memory using separate word selection circuitry. Also temporary storage is available for use by the subsystem for storing intermediate information. See Fig. 14.

Within a block of k words it is necessary to record the availability of a given word or branch. Bookkeeping bits are stored with each word or with the descriptor word. It is normally desirable to place the available space bits for k records in the single descriptor word. The same set of bookkeeping bits in a data word is used to record end-of-record tags.

Regrouping transformations are used freely within a descriptor word to simplify the output registers.

Subsystem operations

A record is associated with a series of descriptors, one for each level of the tree. A retrieval request or an input command begins in essentially the same manner. The first descriptor word is interrogated, and,

Table 1. Tree structure of Fig. 13 where k = 6 with dashed lines representing data records

Address	Reference Descriptor	Descriptors and Data	Reference Address
i	a	a1 a2 a3 a4 a5 a6	j
j	a1	----------/	0
	a2	a21 a22 a23 a24 a25 a26	k
	a3	a31 a32 a33 a34 a35 a36	n
	a4	----------/	0
	a5	----------/	0
j + 5	a6	----------/	0
k	a21	----------/	0
	a22	----------/	0
	a23	----------/	0
	a24	----------/	0
k + 5	a26	----------/	0
n	a31	----------/	0
	a32	b1 b2 b3 b4 b5 b6	n + 6
	a33	----------/	0
	a34	----------/	0
	a35	----------/	0
n + 5	a36	----------/	0
	b1	----------/	0
	b2	----------/	0
	b3	----------/	0
	b4	----------/	0
	b5	----------/	0
n + 11	b6	----------/	0

on the basis of the transformation and comparison results, the next descriptor word is selected. This process is repeated until, in the case

of retrieval, the desired data is read out of the memory subsystem serially by catena. The retrieval process is asynchronous so that an operation is terminated by external interruption by the main processor or by recognition of an end-of-record mark. As more complex variations in the structure of the tree are used, the retrieval process may involve several partial retrieval operations to select complete and related records.

Original input operations require detailed algorithms. For example, if the number of branches per node is known beforehand, the process of reserving space and establishing branches is not complicated. If the tree structure is unknown, a more complex algorithm is necessary. The basic input operation is a combination of a retrieval and storage operation. If a record has four descriptors, the first three levels may be already established so that the retrieval operation would be used through the third level. After the third level is located, available space is selected, and the forth descriptor, transformation addresses, and the record are stored.

Insertion and addition input operations require a change in the tree structure. Insertion requires chaining of one type or another to preserve order. Chaining addresses, modified descriptors (name chaining), and flag codes are some of the methods useful for insertion chaining. An addition to the end of a record is easier since natural sequencing can be used. When a record occupies only a few words, it is sometimes practical to shift information thus allowing natural sequencing to be used.

Deletion starts with a retrieval request to locate the information. Next the deletion is accomplished at the descriptor level by modifying bookkeeping bits. It is not necessary to null the record or descriptor

information or to introduce jump coding.

Hardware requirements

The comparison of each descriptor catena with a search descriptor is the fundamental operation for indirect addressing within a band of k words. In Fig. 15 comparison elements are given. If comparison masking is desired, an additional AND gate is required for each bit position to block the "don't care" comparisons.

Register A performs the following operations: (a) selects an available band with the use of a carry chain and feeds the address encoder, (b) accepts signals from the sense amplifiers and drives, (c) outputs information serially by catena to the output register, (d) feeds the comparison circuitry, (e) accepts information serially by catena from the input register, and (f) updates available space and other bookkeeping facilities. The logic functions required per bit of the output register are given in Fig. 16.

The logic circuits associated with the catena and catena bit have somewhat severe loading requirements. Figs. 17 and 18 show the necessary logic per catena and catena bit.

Due to the number of variations in bookkeeping methods the control circuitry was not studied in detail. Since most of the logic is associated with the output register, an order of magnitude of the required hardware is obtained by considering only the output registers and selections circuits.

Variations in tree structure

This system in its present form gives rise to inefficient memory

Fig. 15. Output comparison logic

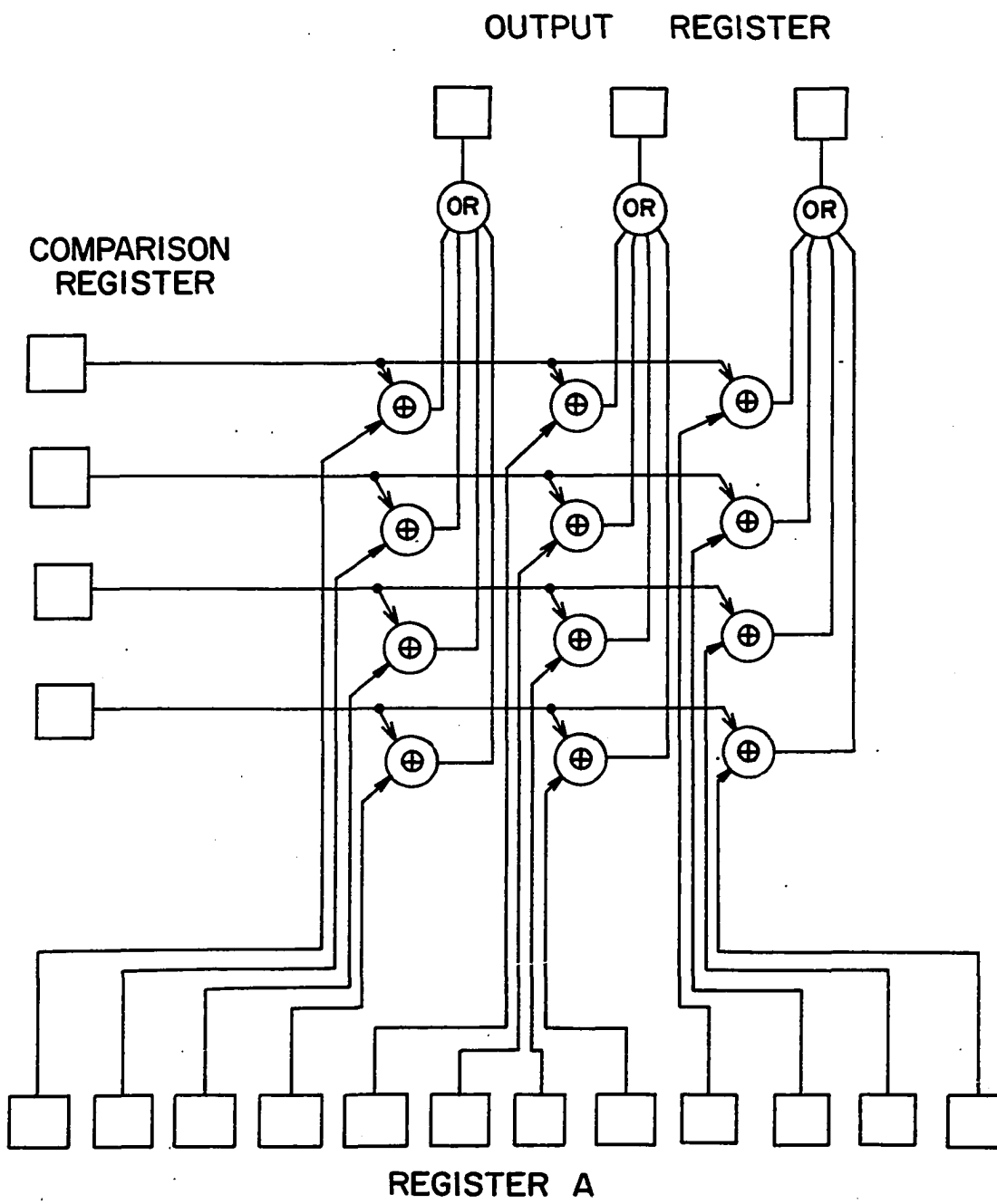

OUTPUT REGISTER
OR
OR
OR
COMPARISON REGISTER
REGISTER A

Fig. 16. Output logic required per bit

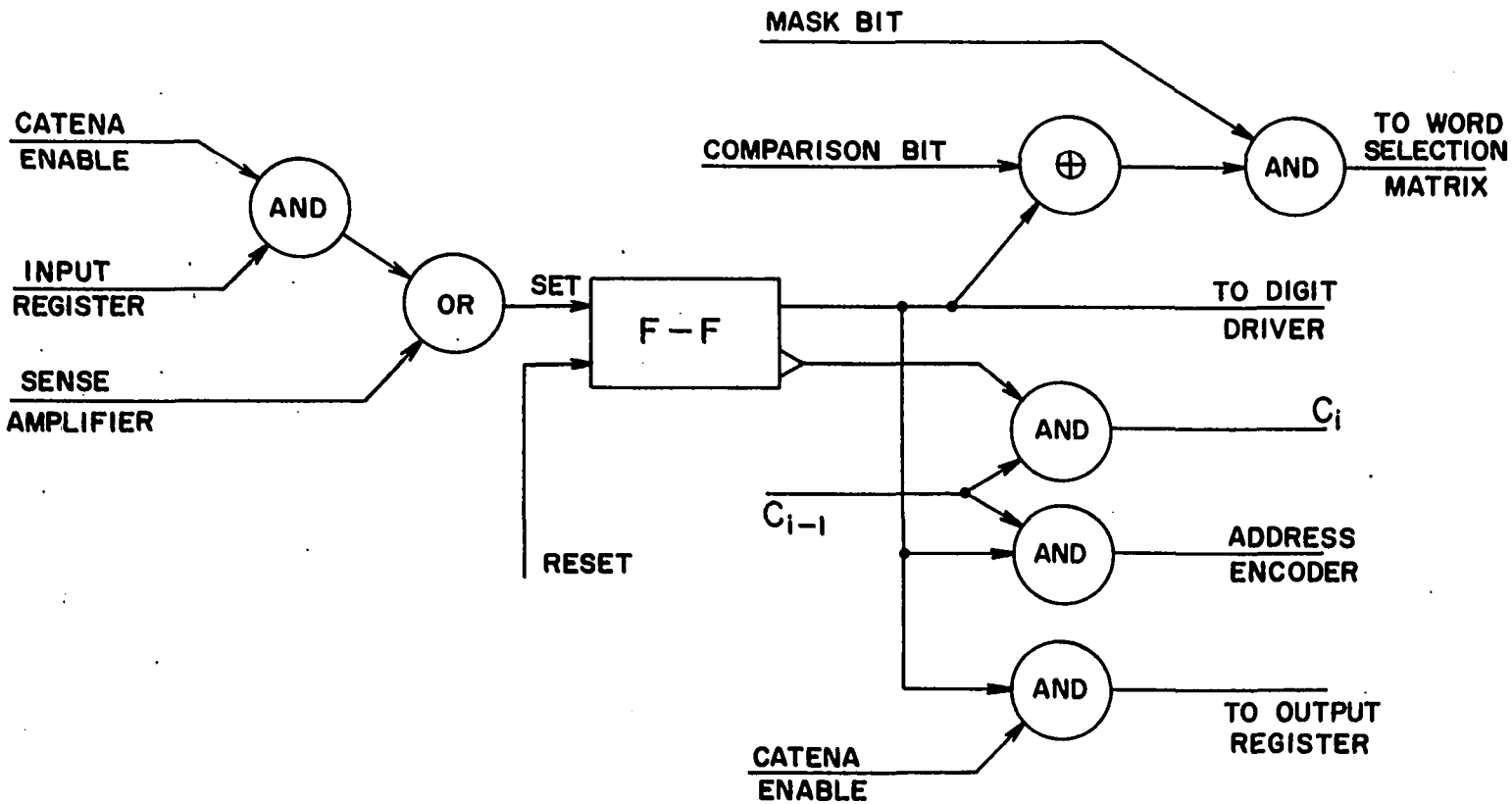

MASK BIT
CATENA
ENABLE
AND
INPUT
REGISTER
OR
SET
F – F
SENSE
AMPLIFIER
RESET
COMPARISON BIT
⊕
AND
TO WORD
SELECTION
MATRIX
TO DIGIT
DRIVER
AND
C_i
C_{i-1}
AND
ADDRESS
ENCODER
AND
TO OUTPUT
REGISTER
CATENA
ENABLE

Fig. 17. Circuitry required per catena bit

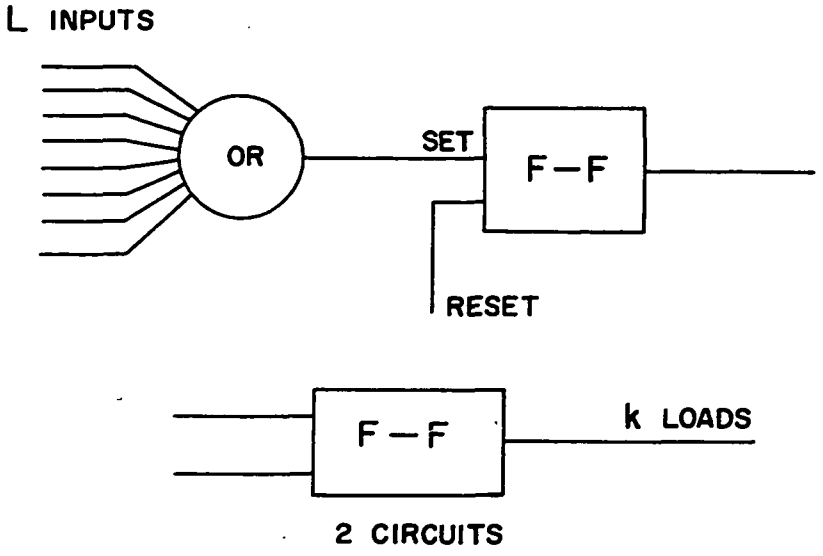
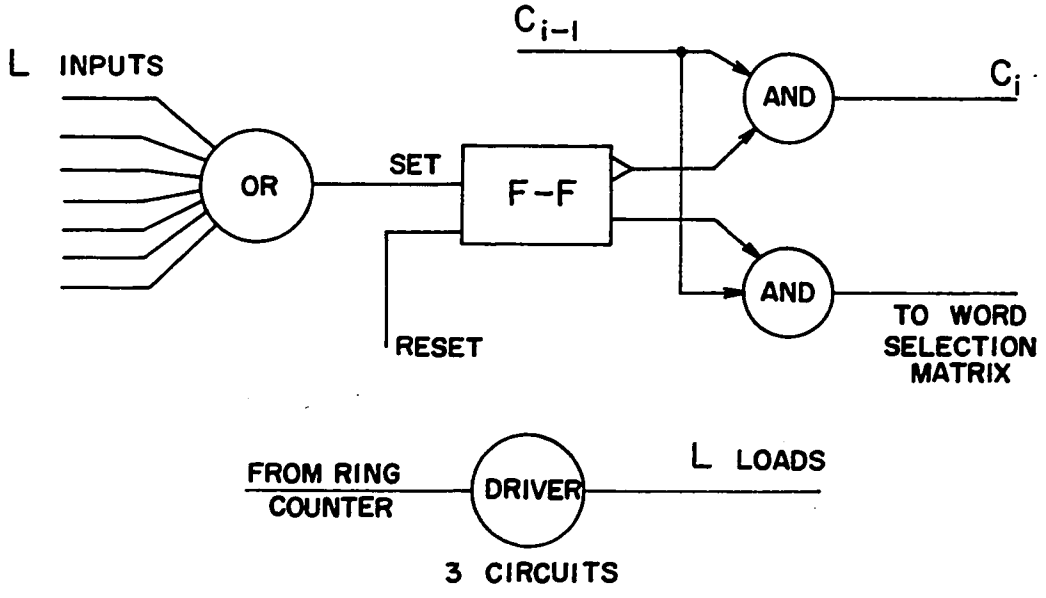

L INPUTS
OR
SET
F—F
RESET
F—F
k LOADS
2 CIRCUITS

Fig. 18. Logic required per catena


L INPUTS
OR
SET
F-F
RESET
C_{i-1}
AND
AND
C_i
TO WORD SELECTION MATRIX
FROM RING COUNTER
DRIVER
L LOADS
3 CIRCUITS

utilization unless the data has a rigid organization. Introduction of variable length records and branching with more or less than k branches requires one or more of the following techniques.

A. Records longer than a memory word require chaining in one form or another. The chaining information can be in the form of an address or code placed with the data or description. For the example in Table 2 sequential ordering, duplicated descriptors, and end-of-record marks provide the chaining information. For example, consider record a, a3 in Table 2. From the information in location i it is implied that, at least, location j + 3 contains some information about a3. Upon referencing location j + 3 and determining that it is data, the information is read out of memory serially by catenae. No end-of-record mark appears at the end of location j + 3, so control is transferred back to the indirect address register and the next word containing any information about a, a3 is retrieved. This process is continued until the end-of-record mark is found in location j + 5. Note that it is not necessary for the different parts of a record to be stored in consecutive locations as long as they are sequential.

B. It is sometimes necessary to have more than k branches per node. The same type chaining described in part A applies on a higher level. In Table 2 a node with 17 branches is implemented when k = 9. Although only 9 records are stored, a referencing capability for 17 words is necessary. Note that record a, a6 is stored in two separate blocks. When chaining on any level is used, retrieval is complete only upon recognition of an end-of-record mark or external intervention.

Table 2. Memory configuration with k = 9 illustrating multiple word records and a large number of branches per node

Address	Reference Descriptor	Descriptor and Data	Reference Address
i	a	a1 a2 a2 a3 a3 a3 a4 a5 a6	j
i + 1	a	a6 a6 a7 a7 a8 a9 a9 a9 0	k
j	a1	----------/	
	a2	----------	
	a2	----------/	
	a3	----------	
	a3	----------	
	a3	----------/	
	a4	----------/	
	a5	----------/	
j + 8	a6	----------	
k	a6	----------	
	a6	----------/	
	a7	----------	
	a7	------/	
	a8	----------/	
	a9	----------	
	a9	----------	
	a9	----------/	
k + 8	0	/	

C. Space sharing is useful to allow more efficient use of space within a fixed structure memory. In Table 3 two nodes with four branches each share a single mapping with k = 8. The descriptors are placed so as to keep the nodes independent.

D. Word packing similar to that described for the two level system is also useful in a multi-level system. Two descriptor words refer to

Table 3. Memory configuration with k = 8 for space sharing with a variable number of branches per node

Address	Reference Descriptor	Descriptors and Data	Reference Address
i	a	a1 a2 a3 a4 0 0 0 0	k
j	b	0 0 0 0 b1 b2 b3 b4	k
k	a1	----------/	0
	a2	----------/	0
	a3	----------/	0
	a4	----------/	0
	b1	----------/	0
	b2	----------/	0
	b3	----------/	0
k + 7	b4	----------/	0

the first and second records respectively in each data word. An example in Table 4 shows word packing with $\alpha = 2$ and record overflow.

E. A more complicated example in Table 5 introduces a variable number of branches per node without space sharing. The available space bookkeeping and addressing for this scheme is more complicated but should have merit in some applications. Note the flexibility in reference addressing. Shifting within the comparison output register could be used for word selection.

F. A more sophisticated tree structure in which several nodes can have common branching is easily implemented using the technique in Table 3. Mutual exclusion is not required allowing complex interlaced branching. This tree structure could have application where there are several unique descriptor classes per record.

Table 4. Word packing example with k = 8, α = 2, and record overflow

Address	Reference Descriptors	Descriptors and Data	Reference Address
i	a	a1 a2 a3 a4 a4 a5 a6 a7	j
i + 1	b	b1 b2 b2 0 b3 b3 b4 b5	j2
j	a1 b1	----------/----------/	0
	a2 ba	----------/----------	0
	a3 b2	----------/----------/	0
	a4	----------	0
	a4 b3	--------/----------	0
	a5 b3	----------/--------/	0
	a6 b4	----------/--------/	0
j + 7	a7 b5	----------/----------/	0

These six variations in tree structure give an idea of the flexibility that is possible in a multi-level system. Since there are so many possibilities, no attempt is made to describe the additional hardware required for these variations. In most cases adding a few bookkeeping bits and changing some control circuitry would be all that is required.

Other Memory Systems

To this point discussion has been restricted to DRO, word-organized memories having long word lengths. These associative memory techniques are certainly not restricted to this type memory. The properties of three specialized memories are discussed.

Nondestructive readout memories

Nondestructive readout, NDRO, memories offer significant simplifications in external circuitry when retrieval is the dominant function. The

Table 5. Tree structure using an unequal number of branches per node

Address	Reference Descriptor	Memory Contents	Reference Address
j	a	a1 a2 a3 a4 0 0 0	j + 1
j + 1	a1	----------/	0
	a2	----------/	0
	a3	----------/	0
j + 4	a4	----------/	0
j + 5	b	b1 b2 b3 b3 b4 0 0	k
k	b1	----------/	0
	b2	c1 c2 c3 0 0 0 0	k + 5
	b3	----------/	0
	b3	----------/	0
k + 4	b4	----------/	0
k + 5	c1	----------/	0
	c2	----------/	0
	c3	----------/	0

comparison function for a band of information requires two output registers when a DRO memory is used. Only one register is required for NDRO operation since it is not necessary to hold the information coming out of memory. The memory read cycle is faster than a read-write cycle, although the access times are similar.

For some technologies, e.g., ferrite cores, it is more difficult to build a NDRO memory. Unless there is a significant advantage of NDRO over DRO, the latter is normally chosen.

A subclass of NDRO memories is of the read-only type. In this case it is impossible or time consuming to alter the contents of memory. Fabrication is normally simpler allowing very large capacities to be realized.

For applications such as dictionary storage or document retrieval this area is receiving considerable attention.

Logic memories

A logic memory has the property of being able to perform the exclusive-OR or equivalence decision directly in memory. The attractive feature comes from the capability of interrogating a complete band of information in a single memory cycle. Noise limitations are severe in some cases reducing the number of words that can be accessed simultaneously. Pohm in a personal interview has described a thin film technique to implement a logic memory in which up to about ten words can be processed simultaneously. A logic memory would be useful in a system similar to the two level system described. Fabrication problems may limit the usefulness of this technique.

Orthogonal read-write memory

A small orthogonal read-write memory has some interesting applications. With orthogonal read and write axes an inversion of a bit matrix is easy. The applications of this are in control, bookkeeping, and associative descriptor storage. For the two level system described, a significant time saving is realized when there is a high input-deletion duty cycle. The memory would be associated with a hierarchy of memories.

APPLICATIONS

A strictly qualitative discussion of the applications of associative memory subsystems is given. The manner in which the techniques of this study apply is described. The properties described are, in general, not directly documented and therefore should be taken in this sense.

General Uses of Associative Memory Systems

Business problems

Business problems have the following general characteristics: (a) several levels of description, (b) nonuniform distribution of descriptors within a descriptor class, (c) variable length information, (d) alphanumeric information, (e) multiple descriptor classes per record, (f) considerably redundant data, and (g) common need for ordered retrieval. Most business problems do not require real time operation so that the upper bound on the retrieval time need not be close to the average retrieval time. Retrieval is not necessarily unique. Large volumes of data occur in some cases. Since this is a large class of problem types, three specific examples are given.

A military personnel file has natural organization. Natural levels of hierarchy such as regiment, company, squadron, etc. or rank classifications exist. Other categories of classification are job description, longevity, and personal status. Unique classifications such as serial number and name are also associated with each record. In a problem of this type it is difficult to call a particular descriptor class a primary class. The records may not be ordered in a manner useful for business

processing; e.g., the data may be ordered chronologically according to enlistment data. Logic retrieval would be useful at some level in the processing. Frequency of usage information is useful in optimizing the organization, e.g., rank might appear in most retrieval requests while blood type may seldom be specified. Classification of descriptor classes into several broad groups such as primary, secondary, and serial scan items is useful. A complex tree structure exists here with a need for methods such as multiple referencing, i.e., several nodes referencing the same set of records.

Material records are similar to personnel records in that they have multiple descriptor categories and natural hierarchy. The number of descriptors per item varies considerably. For example, a screwdriver needs little description while a complex part for a jet aircraft engine requires a detailed description. Input, updating, and deletion have a high duty cycle in personnel files.

Banking records are similar to the above examples except that the record lengths have greater dispersion. Input, updating, and deletion are the principle functions. There is normally only one descriptor per record.

Document retrieval

Real time literature searching is one of the popular applications of an associative memory. The primary characteristics are (a) a fraction of a large amount of data serves as the description, (b) each record has an arbitrary number of descriptors, (c) there may or may not be any natural hierarchy, (d) unique retrieval is uncommon, and (e) retrieval dominates

the duty cycle. It is normally unnecessary to preserve the order of the records.

The descriptors belong to a single class. If the descriptors are placed in a series of fields, retrieval becomes difficult since the descriptors in the retrieval request may not have the same order or placement. See Goldberg and Green (7). An associative memory of the type described is useful if documents have some natural or artificial higher level classification that allows block processing. See Moore (3).

Real time processing is desired since best fit retrieval is often used. An operator can modify the retrieval request until the desired information is obtained.

Medical diagnosis

Medical diagnosis problems are related to document retrieval in the sense that a number of descriptors are associated with each record. The descriptors in this case are symptoms while the records are diseases. The descriptors fall into exclusive and nonexclusive categories. A code book may be used to reduce the redundancy in symptom descriptions. A natural classification exists which allows a tree structure to exist.

Translation

Dictionary storage is the first part of a translation problem. Words are organized into natural and artificial levels of classification. Natural hierarchy comes from characteristics of words, e.g., part of speech. Artificial hierarchy is derived from frequency of usage and spelling. A convenient organization is to branch on the first two letters and then use another technique for further division. Sequential

searching is useful in the final stages of selection. A semi-permanent, read-only memory can be used since virtually all operations are of the retrieval type. A memory hierarchy with a high and a low speed memory is useful to take advantage of frequency information. Common words are placed in a fast memory while less frequently used words are placed in a slow memory. If there are a number of blocks in a system at a particular level, there is no need for all the blocks to be in the same memory.

The actual translation process uses elaborate algorithms and associative trees to analyze the text. A good translation would require examination of blocks of texts such as paragraphs at one time. Associative memories of the type described are very useful for a correlation and association problem such as translation.

Translation is not limited to language-language but includes language-machine, machine-machine, and speech-text translations. Each of these problems can be simplified with the use of associative techniques.

List processing

List processing of the type described by Newell and Shaw (5) and Prywes and Gray (17) is simplified by the use of a multi-level system of the type studied in this work. The elimination or reduction of the use of chaining addresses along with a flexible branching technique increases the efficiency and flexibility of a list processor. Push down lists are easily implemented with the long word structure.

Newell and Shaw (5) give seven characteristics that a list type memory should have. The desire for an item to appear on any number of lists simultaneously would require a more complex bookkeeping facility, especially

if a satisfactory deletion property is desired. The other characteristics should not be difficult to implement.

Large system bookkeeping

In any large system with a hierarchy of memories bookkeeping tasks become time consuming and can require considerable hardware. In the two-level system addressing, space allotment, and small defects for a mass storage media are monitored and controlled by a smaller associative memory. In a system with time sharing, priority control could be accomplished using associative techniques. The cycle time of a bookkeeping memory would normally be fast compared to the cycle time of the memory to be controlled.

Learning problems

In the vast area of learning problems, associative techniques are required to simulate many thinking properties. It is not within the scope of this study to consider this topic in detail. A few words can be said about pattern and speech recognition.

Weighted threshold logic appears to be one of the primary techniques being used in recognition schemes. Reward and punishment are used to weigh one descriptor more heavily than another. The associative memory could be used to store the answers to a recognition request with weighting information associated with the description. Best fit or threshold procedures would be used to establish the match.

Other applications

Simplifications can be realized when solving numerical problems by employing associative techniques. The solution of differential equations

by relaxation techniques involves structured data. Long word length processing offers improvement in processing speed, memory utilization, and intermediate sampling of results.

Other alphanumeric applications involve matrix manipulation, sorting, collating, and table look-up. Orthogonal transfers of information would apply to some matrix problems. Similar addressing techniques can be applied to conventional memories.

Process control, radar tracking, and other real time applications require a minimum of dispersion in retrieval times. This can be provided with an associative memory.

Summary

Each of the applications described have characteristics such as amount of storage required, average record lengths, dispersion in record lengths, number of levels of hierarchy, and number of descriptors per record. For comparison purposes Table 6 was compiled to give orders of magnitude of the system parameters. When two numbers for an entry are given this implies a typical range of values.

The input-output duty cycle implies changing the tree structure or its contents. The retrieval response implies the type of retrieval, i.e., single or multiple.

Table 6. Characteristics of several problems that can use associative memory techniques

Application	Storage Required (bits)	Average Record Length (bits)	Levels of Hierarchy	Average Descriptors per Record	I/O Duty Cycle	Dispersion of Record Lengths
Personnel File	10^6-10^8	10^2-10^3	2-4	5	medium	low
Banking Data	10^6-10^7	10^2-10^4	2	1-2	high	high
Material Records	10^6-10^8	10^2-10^3	3-5	3-5	high	medium
Document Retrieval	10^8-10^9	10^2-10^3	2-3	5-15	low	medium
Medical Diagnosis	10^6-10^8	10^2-10^4	3-5	5-15	low	medium
Dictionary Storage	10^7-10^8	10^2-10^3	2-3	1-2	low	medium
List Processing	10^5-10^7	10^2-10^4	5-15	1	high	very high
Pattern Recognition	10^4-10^6	10^2-10^4	2	--	medium	low
Memory Bookkeeping	10^4-10^5	20-50	--	1	medium	zero

CONCLUSION

By studying present limitations of conventional computer memories it is conceded that associative memories will play an important role in future computers. The topics stressed in this study were (a) organization based on long word lengths, (b) hierarchal data structures, (c) implementation of tree structures, and (d) storage and retrieval based on description rather than specific location. The relationship of these techniques to memory subsystems was studied.

In making associative or content-addressed memories the notion of parallel or simultaneous searching is often dominant. Caution must be taken not to lose or ignore built-in hierarchy that is present in most data structures. If serial or tree searching can be done rapidly, a 10 or 100 cycle search may be a lot easier to implement than a strictly parallel search. If there are multiple responses, the information resulting from a parallel search must be held in some sort of large register. As a compromise the search is often broken up into blocks for interrogation. When this is the case, the natural hierarchy of the data should be investigated and possibly used.

A conventional memory organization was assumed for most of this study. The point that is to be stressed is that it is not necessary to have sensing on more than one axis of a memory with associative properties. As long as detection circuits are costly, a detector for each word or record appears impractical for large memories. A single read axis is sufficient in most cases.

The techniques for processing long words are relatively new. With

long word lengths word packing is an important factor. The processing of long words serially-by-catena is necessary. A transformation of a long register from a one dimensional string to a two dimensional array is useful.

The basic transformations, bit-to-word and catena-to-word, were used to implement different tree forms. In the design of tree structures it was found that the number of variations and hence the flexibility was virtually unlimited. With this in mind the problem of effective utilization of this facility is paramount. A few of the more challenging applications appear to be in the areas of machine learning and structured machine languages.

The techniques described apply to some extent to a wide spectrum of memories from small 10^3 bit control memories to 10^9 bit bulk storage. The application dependence of the system parameters has been illustrated. With no firm application constraints it was virtually impossible to attempt a detailed comparison and evaluation.

The place of an associative memory subsystem in a large data processor is not clear. The concept of a hierarchy of memories appears to dominate system thinking. There will not be a rapid break from conventional memory organization so in the near future associative memories will probably be introduced only as supporting systems.

LITERATURE CITED

1. Kellogg, Charles. The FACT compiler: a system for the extraction, storage, and retrieval of information. Western Joint Computer Conference Proc. 17: 73-82. 1960.

2. Miller, L., J. Miner, W. G. Reed, and W. E. Shindle. A multi-level file structure for information processing. Western Joint Computer Conference Proc. 17: 53-60. 1960.

3. Moore, Robert T. A screening method for large information retrieval systems. Western Joint Computer Conference Proc. 19: 259-274. 1961.

4. Mullery, A. P., R. F. Schauer, and R. Rice. Adam: a problem-oriented symbol processor. [To be published in Spring Joint Computer Conference, Detroit, May 21-23, 1963, 23rd Proc. ca. 1963.]

5. Newell, A. and J. C. Shaw. Programming the logic theory machine. Western Joint Computer Conference Proc. 11: 230-240. 1957.

6. Booth, A. D. and A. J. T. Colin. On the efficiency of a new method of dictionary construction. Information and Control 3: 327-334. 1960.

7. Goldberg, J. and M. W. Green. Large files for information retrieval based on simultaneous interrogation of all items. In Yovits, Marshall C. ed. Large-capacity memory techniques for computing systems. pp. 63-77. Address, Association for Computing Machinery. 1962.

8. Goldberg, J., M. Green, H. Heckler, E. Van DeRiet, R. Singleton, and E. Frei. Multiple instantaneous response file. U. S. Government Research Report AD 266 169. 1961.

9. Prywes, Noah S. and Harry J. Gray. The multi-list system for real-time storage and retrieval. Unpublished paper presented at International Federation of Information Processing Societies Congress, Aug. 1962. Mimeo. Philadelphia, Pennsylvania, Moore School of Electrical Engineering, University of Pennsylvania. 1962.

10. Cochran, Robert. Information retrieval study. Western Joint Computer Conference Proc. 15: 283-285. 1959.

11. King, Gilbert W. Table look-up procedures in language processing. I. The raw text. IBM [International Business Machines] Journal of Research and Development 5: 86-92. 1961.

12. Booth, Andrew D. Use of a computing machine as a mechanical dictionary. Nature 176: 565. 1955.

13. Briandais, Rene De La. File searching using variable length keys. Western Joint Computer Conference Proc. 15: 295-298. 1959.

14. Landauer, W. I. and N. S. Prywes. A growing tree for descriptor language translation. Unpublished paper presented at Symposium of Symbolic Languages in Data Processing, Mar. 1962. Mimeo. Philadelphia, Pennsylvania, Moore School of Electrical Engineering, University of Pennsylvania. 1962.

15. Peterson. W. W. Addressing for random-access storage. IBM [International Business Machines] Journal of Research and Development 1: 130-146. 1957.

16. Kiseda, J. R., H. E. Petersen, W. C. Seelbach, and M. Teig. A magnetic associative memory. IBM [International Business Machines] Journal of Research and Development 5: 59-62. 1961.

17. Prywes, Noah S. and Harry J. Gray. A report on the development of a list-type processor: I. Unpublished paper. Mimeo. Philadelphia, Pennsylvania, Moore School of Electrical Engineering, University of Pennsylvania. 1962.

18. Mann, Horace T. and John L. Rogers. A cryogenic "between limits" associative memory. National Aerospace Electronics Convention Proc. 1962: 359-362. 1962.

19. Seeber, Robert R. Associative self-sorting memory. Eastern Joint Computer Conference Proc. 18: 179-187. 1960.

20. McDermid, W. L. and H. E. Petersen. Magnatic associative memory system. IBM [International Business Machines] Journal of Research and Development 5: 59-62. 1961.

21. Corneretto, Alan. Associative memories: a many-pronged design effort. Electronic Design 11, No. 3: 40-55. Feb. 1, 1963.

22. Slotnick, Daniel L., W. Carl Borck, and Robert C. McReynolds. The Solomon computer. Fall Joint Computer Conference Proc. 22: 97-107. 1962.

23. Lee, C. Y. Intercommunicating cells, basis for a destributed logic computer. Fall Joint Computer Conference Proc. 22: 130-136. 1962.

24. Petschauer, Richard D. and Rodney D. Turnquist. A nondestructive readout film memory. Western Joint Computer Conference Proc. 19: 411-425. 1961.

25. Joseph, Earl C. and Albert Kaplan. Target track correlation with a search memory. Unpublished paper presented at Sixth National Convention on Military Electronics, June, 1962. Mimeo. St. Paul, Minnesota, Univac Division, Sperry Rand Corporation. 1962.

26. Slade, A. E. and H. O. McMahon. A cryotron catalog memory system. Eastern Joint Computer Conference Proc. 10: 115-120. 1956.

27. Slade, A. E. and C. R. Smallman. Thin film cryotron catalog memory. Symposium on Superconductive Techniques for Computing Systems Proc. 1960: 213-229. 1960.

28. Seeber, R. R. and A. B. Lindquist. Associative memory with ordered retrieval. IBM [International Business Machines] Journal of Research and Development 6: 126-136. 1962.

29. Lewin, Morton H. Retrieval of ordered lists from a content-addressed memory. RCA [Radio Corporation of America] Review 23: 215-229. 1962.

30. Newhouse, V. L. and R. E. Fruin. A cryogenic data addressed memory. Spring Joint Computer Conference Proc. 21: 89-99. 1962.

31. Davies, Paul M. A superconductive associative memory. Spring Joint Computer Conference Proc. 21: 79-88. 1962.

32. Rajchman, Jan A. Computer memories: possible future developments. RCA [Radio Corporation of America] Review 23: 137-151. 1962.

33. Pohm, A. V., R. J. Zingg, G. A. Watson, T. A. Smay, and R. M. Stewart, Jr. Large, high speed DRO memories. International Conference on Nonlinear Magnetics Proc. 1963: 9-5-1--9-5-14. 1963.

34. Probster, W. E. The design of a high-speed thin-magnatic-film memory. International Solid-State Circuits Conference Digest 5: 38-39. 1962.

35. IBM 1301 disk storage with IBM 1410 and 7010 systems. IBM [International Business Machines] Systems Reference Library. Form Number A22-6670-1. c1962.

ACKNOWLEDGEMENT

The author wishes to express his sincere thanks to Dr. A. V. Pohm for many enlightening conservations related to this investigation. The author also wishes to thank the staff members of the Electrical Engineering Department and Cyclone Computer Laboratory for their assistance.

This work received partial support from the Iowa State Affiliate Research Program in Solid State Electronics and the author's Title IV National Defense Education Act Fellowship.

APPENDIX

The following calculations give a brief comparison of various types of information storage. Upper and lower bounds on the memory utilization efficiency are given for the long-word technique, and upper bounds are given for other techniques.

For the proposed long-word technique assume that 4000 words with 400 bits each are available for the primary storage of information. Allow a maximum of four references per word giving a maximum of 16,000 possible record names. Each name requires 14 bits in a non-redundant code and one available space bit. The minimum information stored is

$$(4 \times 10^3)(400)\frac{19^2}{20^2} + (4 \times 10^3)(14) = 1.5 \times 10^6 \text{ bits}$$

for a full memory loaded in the least efficient manner. This assumes that the average record length is greater than k/α. The maximum information stored is

$$(4 \times 10^3)(400)\frac{19}{20} + (1.6 \times 10^4)(14) = 1.74 \times 10^6 \text{ bits.}$$

The factor of 19/20 arises from the use of a flag bit in each catena and the $\frac{19^2}{20^2}$ comes from the flag bit and the restriction on the minimum number of catenae necessary to start a new record. The total memory space used is

$$(4 \times 10^3)(400) + (1.6 \times 10^4)(15) = 1.84 \times 10^6 \text{ bits.}$$

The resulting efficiencies are 81.5% minimum and 94.5% maximum.

The list processor designed by Newell and Schaw (5) uses a two-address machine. Assume that one 40-bit data word is available. 4×10^4 addresses or 16 bits would be required for an address. The remaining 8 bits in the instruction word would be used for bookkeeping or instructions. An upper bound on the efficiency is obtained by neglecting the selection scheme and partly filled words. The resultant maximum efficiency is 50%.

A one-address machine can also be used in a list processor. Assume a word has 40 bits for data and 16 bits for chaining addresses. This is the same as the Newell and Schaw method except that the reference to the data word is eliminated. Neglecting selection and bookkeeping facilities the maximum efficiency is 72%. Increasing the length of the word will not help the efficiency appreciably because of the problem of word packing.

In the memory described by Mullery et al. (5) 10 bits are used for bookkeeping and hierarchal information. The total word length is 72 bits. Since the 10 bits do not contain a description of the data, the maximum efficiency could be considered to be $\frac{62}{72}$ or 86%. If the seven bits of structure information were considered to contain information, the redundancy would limit the efficiency to $\frac{65}{72}$ or 90%. Again selection, chaining addresses, and partly filled words are neglected.