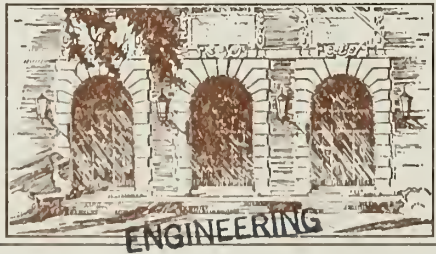


LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84

Il63c

no. 1-10



AUG 5 1976

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN


ENGINEERING

CONFERENCE ROOM

JUL 07 REC'D

AUG 04 1983

JUL 15 REC'D



Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

<http://archive.org/details/informationmanag00schu>

510.84
I263c
no.1

Engin.

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

CONFERENCE ROOM

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 1

AN INFORMATION MANAGEMENT
AND
ANALYSIS SYSTEM FOR ILLIAC IV

by

Stewart A. Schuster

December 11, 1970

AN INFORMATION MANAGEMENT
AND
ANALYSIS SYSTEM FOR ILLIAC IV

by

Stewart A. Schuster

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

December 11, 1970

510.84
IL63c
no. 1-10

ABSTRACT

This document presents the preliminary specifications for the design and implementation of an Information Management System, capable of being integrated with an extensive Mathematical Analysis System to be implemented on the ILLIAC IV computer. The system would provide multiple keyed access to several large data bases and the capability of receiving instructions and questions expressed in a simple inquiry language. Mathematical and statistical routines will be interfaced through the Analysis System. There will also be a report generator which displays the responses to questions in forms easily interpretable by researchers.

ACKNOWLEDGEMENTS

I would like to thank Professor Daniel Slotnick for sponsoring and encouraging my work in this area.

I would especially like to express my gratitude to Dr. James L. Parker of the University of British Columbia whose many ideas were germinal to the design of this system. Also, I wish to express appreciation for the careful and patient analysis and suggestions pertaining to this work by Professor Michael Sher, Peter Alsberg and Thomas Mason, who are members of the Center for Advanced Computation. Michael Randal and Jeffrey Serage, members of the ILLIAC IV Project, also made valuable contributions to the presentation of this document. Finally, my thanks to the entire secretarial staff and especially to Mrs. Shirley Brown, Mrs. Coni Allen, and Mrs. Kay Flessner for outstanding clerical assistance.

TABLE OF CONTENTS

	Page
I. The Information Management System as an Information Management Machine	1
A. Introduction	1
B. Design Goals	3
II. ILLIAC IV Computer System Characteristics that Influence Information Retrieval	7
A. Hardware	7
B. Software	8
C. Languages	9
III. Specifications for the ILLIAC IV Information Management System . .	10
A. Introduction	10
B. The Data Management System (DMS)	10
C. The Symbolic Data Structuring System (SDSS)	12
D. The Information Retrieval System (IRS)	13
E. The Mathematical Computation System (MCS)	14
F. Using the Information Management System	14
IV. List of Tasks	16
V. The Use of an Existing Data Base for Testing and Evaluating the IMS	17
A. Introduction	17
B. A Description of the Rockland State Hospital Data Base	17
Appendix A: References	20
Appendix B: A Glossary of Terms	21
Appendix C: A Suggested Design for the Data Management System (DMS) .	23
C.1 Responsibilities of the DMS	23
C.2 The DMS Information Elements	24
C.3 Data Management on the Information Elements	28
C.3.1 Specifying the I/O Scheme to the I/O Subsystem of the Operating System	28
C.3.2 Using the DMS	32
C.3.3 Garbage Collection	34
C.3.4 Comments on Direct Accessing	35
C.3.5 Comments on the DMS	36

Appendix D: A Suggested Design for the Symbolic Data Structuring System (SDSS)	38
D.1 Responsibilities of the SDSS	38
D.2 The Entities of Data Structuring	38
D.3 The Data Item Symbol Tables	44
D.4 Data Types and Attributes and Characteristic Definitions . . .	58
Appendix E: A Suggested Design for the Information Retrieval System (IRS)	62
E.1 Responsibilities of the IRS	62
E.2 The IRS Description	62
E.3 Interaction with the Mathematical Computation System (MCS) . .	64

LIST OF FIGURES

	Page
1. Conventional Micro-Programmable Computer vs. An Information Management Machine	2
2. ILLIAC IV Hardware Relevant to Information Retrieval	7
3. Diagram of a DMS Information Element	24
4. The Three Types of Information Elements	25
5. The Syntax that Governs the Occurrence of Hole, Key, and Record Elements	27
6. A Typical Information Element Distribution within a Buffer	29
7. A Beginning Quadrant	31
8. Garbage Collection	34
9. The DMS vs. Direct Access Systems Performances on Processing Batches of Elementary Questions	36
10. The Control and Information Flow Between the Various Modules of the DMS	37
11. Example of the Tree Structure of a Record	39
12. Lattice-Tree Data Structure of the IMS	41
13. Example Key and Format Declaration	43
14. Example of Entries in the IMS Symbol Table	48
15. The Data Structure Defined by the IMS Symbol Table	49
16. Example File and Record Format for the X-HOSPITAL_DATA_BASE	50
17a. Example of the Symbol Table for the X_HOSPITAL_DATA_BASE	51
17b. Example of the Symbol Table for the X_HOSPITAL_DATA_BASE	52
18. The Data Structure Defined by the Symbol Table	53
19. Key and Record Element Formats for the PATIENT_FILE	54
20. Key and Record Element Formats for the INSURANCE_FILE	55
21. Control and Information Flow of the SDSS Modules	61
22. The Control and Information Flow for the IRS	63
23. A Language Hierarchy Established for High Level Language Compilation	65
24. Hierarchy Language Control of Subsystems	66
25. The Language Subsystem Approach Applied to the IMS and MCS Systems	67

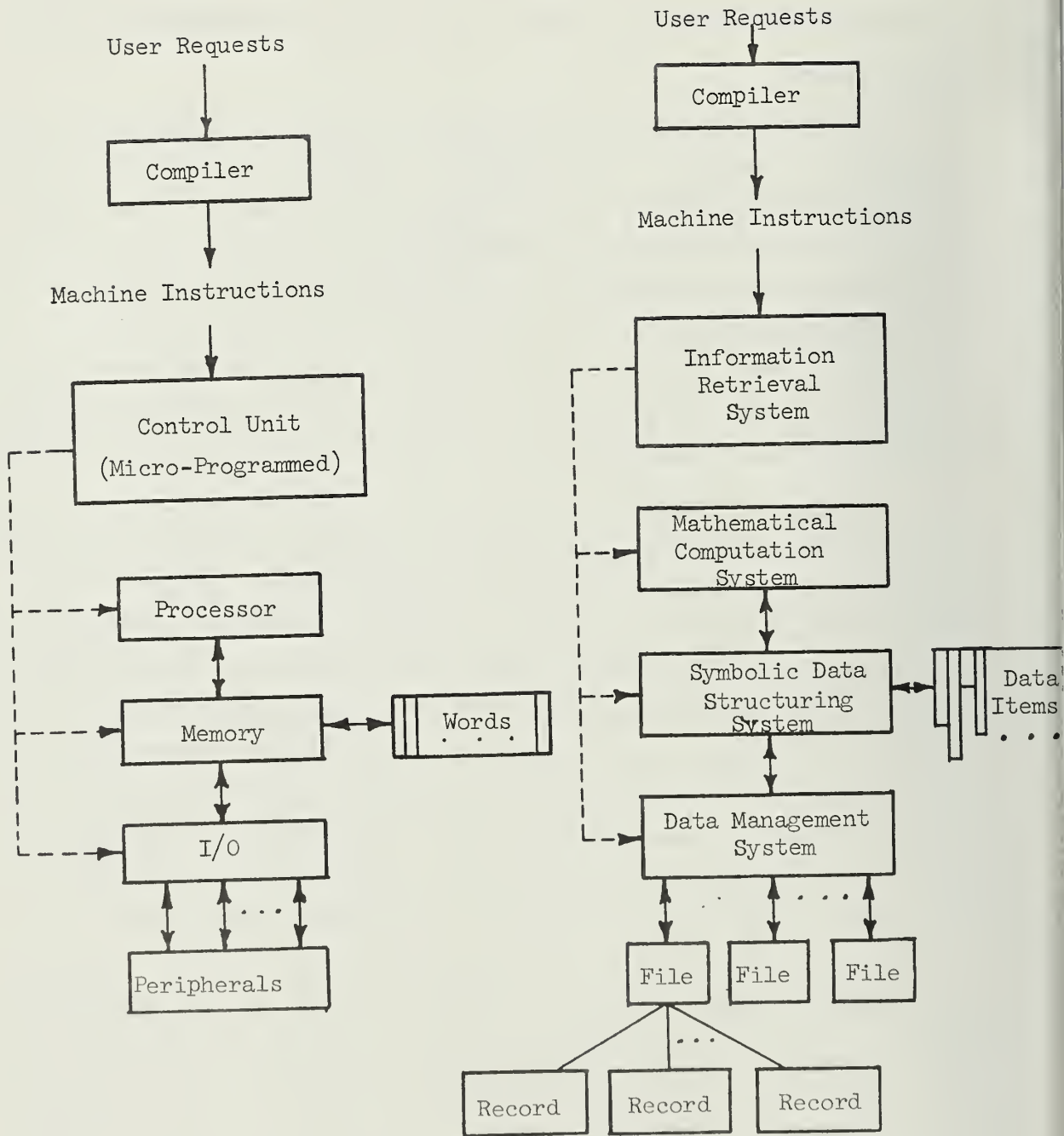
I. The Information Management System as an Information Management Machine

A. Introduction

This is a proposal for providing an Information Management System on the ILLIAC IV computer. The system will interpret programmable queries to determine which records of which files must be retrieved. From these records, data will be extracted and operated upon by statistical or other mathematical routines according to the researcher's request. The responses to these requests will then be translated and output to the user in an easily readable form. The user can interpret the results and formulate new questions to be asked of the system. All the conventional data management functions such as sorting, merging, and building files will also be provided.

It may be instructive to the reader to view this system as a conventional special purpose micro-programmable computer designed only to perform the tasks of information management and processing. Whereas, a conventional computer operates on words of data, this "machine", referred to as the Information Management Machine (IMM), will operate on values of elementary data items. An elementary data item is an alpha-numeric name of a non-divisible piece of data, e.g., SOIL_TYPE or EMPLOYEE_NO. An additional difference is that conventional machines use words of fixed size, but the IMM uses variable length elementary data items. The analogy is diagrammed in Figure 1. The IMM will be simulated on the ILLIAC IV computer system and will be known as the ILLIAC IV Information Management System (IMS).

The IMS is composed of four main interacting modules: the Data Management System (DMS), the Symbolic Data Structuring System (SDSS), the Information Retrieval System (IRS), and the Mathematical Computation System (MCS). Figure 1 is also a diagram of the control and information flow between these various modules of the IMS. Each module essentially performs the same functions as its counter-part in the conventional machine. Note that the micro-programmable computer can change its programming semantics without changing the instruction stream. Similarly, the Information Management Machine can change its function without altering the languages used between the modules. This allows the various modules to develop simultaneously and relatively independent of each other. It also reduces complex interactions between the various design groups which speeds up implementation and allows easier debugging.



Conventional Micro-Programmable Computer

Information Management Machine

—————> information flow

- - - - -> direction of control

Conventional Micro-Programmable Computer Vs.
An Information Management Machine

Figure 1

Also, the "micro-programming" facility of the Information Management Machine provides the capability for the system to respond to individual users so that user dependent optimization of searching times may easily be implemented.

At this point, a brief examination of the glossary of terms in Appendix B, page 21, would facilitate further reading.

B. Design Goals

The rationale of any information management system can only be understood in the light of the type of data to be handled and the operations to be performed on the data. A system that handles only one data base, which expands, but never changes structure, and is only concerned with routine retrievals, can be designed to be very efficient due to the lack of generality required by the users. Since the designers know a great deal about this system, many short cuts may be taken; however, the more assumptions made and the more short cuts taken, which do improve system speed, the greater the chance that the system could be rendered useless under production stresses - the data base could grow too large or queries could begin to get so complex that serial searches are constantly performed (serial searches are very slow on serial machines). Since the system was "hard" coded to take advantage of short cuts, changes to the system are more likely to require the entire retrieval and data structuring subsystems to be rewritten. Therefore, before suggesting a design, we must define, as closely as possible, the classes of data to be manipulated, the types of processing most likely to be desired, and the response times needed.

Since the IMS will be used by a wide variety of researchers in the social, physical, and life sciences, it is apparent that the system must truly be a generalized information management and modeling system. With such diverse users, no assumptions can be made about the operations that will be performed on data records, files, and data bases. The system must also be amenable to changes--some of which may be anticipated.

It is apparent that this system will be a query as opposed to a transaction system--that is, it will in general process complex requests instead of simple data item retrievals. It is more likely to answer queries

of the form, "Is there a significant difference between blacks and whites in attending health facilities in integrated slums?" rather than "What was the temperature in Champaign, Illinois on December 1, 1970?"

For a system to be truly a research tool, it must provide an effective computational facility. It must provide statistical and other modeling and simulation capabilities; these systems must be integrated into the overall information management process. This is to avoid having researchers spend months manipulating data from one system to analyze it on another.

The speed of such a system is important if interactive graphic monitoring becomes desirable. There is a growing need to be able to create experimental files during simulations and display the results as they are found. If a researcher does not like what he sees, he would like to be able to call a file which maintains a program of the model or the input data and make immediate changes and rerun the model. He wants to think about models - not about the data or the computer. If he approves of what is developing on the terminal, he could save the results for later uses. The requirement of speed, which is ILLIAC IV's major asset, becomes as important as the flexibility of the system.

The design goals are listed as follows: Terms and concepts will be explained in the referenced sections. The system must:

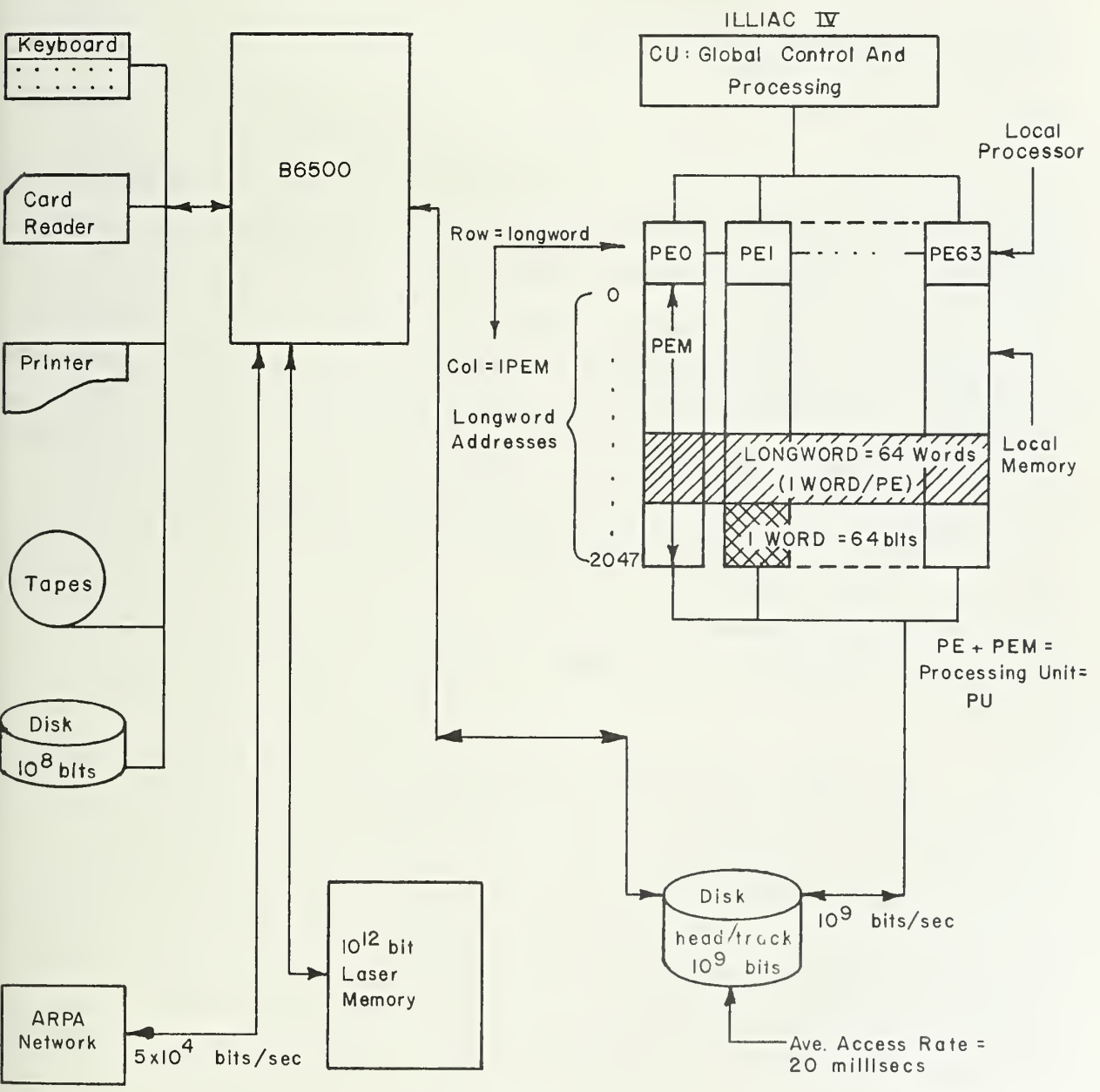
- 1) Be a modular system which may be easily changed, allows incremental implementation, and expandability as experience is gained. A subset of the system must be specified which can be implemented to begin processing data without having to wait for a complete system. (Section III).
- 2) Make efficient use of ILLIAC IV's parallelism. (Section III).
- 3) Provide the ability to dynamically maintain system measurements which will be used to suggest system "tuning" to improve the performance of the system or measure the structural characteristics of different data bases to detect inefficiencies. The number and size of I/O buffers are some elementary but very influential parameters that greatly affect performance of these types of systems. (Appendix C).

- 4) Provide the ability to view the data base record structure as a tree whose nodes correspond to data items, but also allow freely generated cross links between different nodes of the trees in different records. (Appendix D).
- 5) Allow the user to easily retrieve records by selecting any data items to be utilized as keys upon which to search for a subset of records within a file. (Appendix D).
- 6) Provide multiple keyed access to large data bases that would exist in arbitrarily constructed and dynamically formed disk-sized blocks of less than 10^9 bits (initial system size of the disk). Each variable disk block would be made up of several files, each being on the order of 10^8 bits in size (the approximate size of a tape file). (Appendix D).
- 7) Provide the user the capability of using alpha-numeric names as a means of identifying data at any level. (Appendix D).
- 8) Provide complete flexibility for entering, deleting, structuring, and naming data bases, files, records and data items that could contain information in any processable form. (Appendix D).
- 9) Provide the capability to protect data from misuses by unauthorized system users. (Appendix D).
- 10) Provide for the general alteration of data structures. (Appendix D).
- 11) Provide conditional control in the query programs. This will allow different actions to be involved depending on values of retrieval data items or the outcomes of the computational routines. (Appendix E).
- 12) Provide the capability to use the system through one uniform language. (Appendix E).
- 13) Provide a basic and expandable source of mathematical and statistical routines to operate on researcher's data bases. (Section III and Appendix E).

- 14) Allow other systems (e.g., the Linear Programming System or the Statistical System) to use the information management facilities of the IMS for temporary file maintenance, system program overlaying, and data definitions. (Appendix E).
- 15) Provide comprehensive report generation facilities. (Appendix E).
- 16) Provide the capability for simulated "time-shared interactive" use. (Appendix E).

II. ILLIAC IV Computer System Characteristics that Influence Information Retrieval

A. Hardware



ILLIAC IV Hardware Relevant to Information Retrieval

Figure 2

Figure 2 is a sketch of relevant ILLIAC IV hardware to aid in the discussion. The ILLIAC IV computer can process 64 memory blocks (PEM's) simultaneously by executing the same instruction stream (modified locally) in each processor (PE). Each PE has uninhibited access to its own PE memory (with access time of 200 ns) but may only access other PE memories indirectly (with access times ranging from 300-900 ns) by loading the data into a PE and then routing it to the demanding PE. Therefore, indirect PE addressing is to be avoided, if possible. Consequently, it appears more fruitful to process a logical block of data, such as a record, within one PE instead of across PE's. Memory is, therefore, viewed as 64 independent data streams composed of concatenated, variable length, logical blocks of data.

Each PE is about twice as fast as a CDC 6600 processor; at full efficiency (i.e., all PE's working) the ILLIAC IV is two orders of magnitude faster than the CDC 6600. The ILLIAC IV system maintains a direct access 10^9 -bit-head-per-track disk file system (expansion facilities could provide up to 20×10^9 bits) with a transfer rate of 10^9 bits/sec and an average access time of 20 ms. The disk file is actually composed of 13 disks called Storage Units (SU) - each contains 8×10^7 bits. A 10^{12} -bit laser memory, accessible by file names, can transfer a complete ILLIAC IV disk load of 10^9 bits, i.e., approximately 10 magnetic tape files, in 2 minutes or a tape equivalent in 12 sec.

The ILLIAC IV processors have been estimated to be slightly I/O bound. Based on an average ILLIAC IV instruction time, the processors can each execute 16 instructions in the time it takes a load 1 long word (64 words -- 1 word/PEM). Although I/O bound, as are most of the large scale computers while engaged in information processing, the I/O rate is extremely fast -- about 400 times faster than the IBM 2314 disk system. With the 10^9 bit/sec transfer rate from disk to core, the ILLIAC IV can search a complete disk load, i.e., 10^9 bits or about 10 magnetic tapes full of data, in 1 second.

B. Software

A most influential characteristic of the I/O operating system is that I/O proceeds across rows with a minimum transfer of 16 words (1/4 of a longword). This means that to access single records on the disk they must

be stored into memory across rows. To process records stored across rows presents many complex programming problems. Thus, another approach would be to transpose each record from across rows into a single column PE memory. This process is slow due to routing requirements. However, the IMS could preprocess records into a transposed form on the B6500 or ILLIAC IV before they enter the laser memory. This approach is discussed further in Section III B and Appendix C.

C. Languages

Presently, the ILLIAC IV system is supporting two languages, ASK and GLYPNIR. ASK is an assembly language with macro facilities and GLYPNIR is an ALGOL-like list processing language. Since GLYPNIR can call ASK sub-routines, it will be used as the major language to implement the IMS. For routines that rely on the most efficient processing available, assembler sub-routines would be invoked.

A FORTRAN-like language has been specified, and is in the implementation stage. It will possess all the special facilities of GLYPNIR.

III. Specifications for the ILLIAC IV Information Management System

A. Introduction

The functions of the Information Management System (IMS) may be separated into three areas: file handling, computational facilities, and query analysis with report generation. File handling capabilities are provided by the Data Management System (DMS) in conjunction with the Symbolic Data Structuring System (SDSS). The Mathematical Computation System (MCS) provides an extensive source of mathematical and statistical routines that can operate on specific or random samples of data items from files. Incoming queries are analyzed by the IRS, which controls the actions of the other subsystems and causes reports to be generated to answer the queries.

The functions of each of the modules will be discussed in the following sections. The emphasis has been placed on specifying their responsibilities. Suggestions for each module's design will be found in Appendices C, D, and E. Research is required to determine the performance of these designs and whether new approaches should be proposed.

B. The Data Management System (DMS)

Present information management systems spend most of their time in bookkeeping overhead. Systems that do not provide direct access facilities spend large amounts of processing time in sorting, matching and merging files before any productive processing can be done. This is a result of having to use magnetic tapes (a tape is usually larger than most disks) for bulk storage. There are systems that provide some direct access facilities on subpages of files that have been placed on disks; however, these systems are plagued by extensive tables that point to each record (e.g., the TDMS system developed by System Development Corporation). These systems are clogged by the queuing of I/O requests. This is especially true in time-sharing facilities where many data sets of widely varying characteristics must be kept and maintained for variable lengths of time. Also, many systems are forced to have complicated and costly garbage collection penalties. As a system grows, the amount of code generated by these overhead routines may easily reach the critical point where it becomes necessary to overlay system programs. For many systems this paging becomes so costly that the system must be rewritten.

To avoid these problems, we would like to create a simple data management system (DMS) which has a high access rate to data and is designed to minimize queuing problems. The proposal of such a design and its details are given in Appendix C.

The DMS provides the lowest level support for the Information Management System. It has been designed to provide an easily useable general purpose file acquisition system to bring files from the laser memory to the ILLIAC IV disk. Further retrieval structures called key elements provide search arguments which are matched against other key elements in order to retrieve a subset of records from a file residing on the disk. These records are then passed to the SDSS in ILLIAC IV memory for analysis.

A separate module is available in the DMS to transfer card and tape files to the laser memory and provide any necessary reformatting so that these files can be directly fed to disk memory upon request. File reformatting is required by the DMS because it is designed to handle a very simple form of data structure called the information element. The reformatting process need only be done once; restrictions are placed on the flexibility of the file structure.

There are three types of information elements: key, record, and hole elements. Key elements contain data items upon which searches can be made to find collections of related data items. A collection of related data item values is called a record. Records are maintained in record elements. Whenever a record is deleted, a space is left in the data stream. This space is identified by a hole element. The DMS will provide a complete set of operations to form, maintain, or delete these elements.

The system may periodically scan the disk to consolidate all the hole elements whenever efficient performance is threatened by not having available hole elements that are large enough to accept new records. This will be handled by separate routines that will be invoked, unknowingly by users, whenever the performance monitor senses the need.

The performance monitor is a module that maintains all the DMS parameters and can suggest changes to designers or users when the system encounters inefficiencies due to unusual data base characteristics or if the

original design was faulty. These parameters could be the number of I/O buffers and buffer sizes.

The DMS will receive requests from the SDSS and must translate them into a sequence of simple DMS commands. The DMS will notify the SDSS when the commands have been finished.

C. The Symbolic Data Structuring Systems (SDSS)

To utilize the DMS directly as a comprehensive information retrieval system would force the user to code and decode all of his file names and data structures. This is because no system exists to handle record element decoding. A user would have to know all the details of his data and how it related to other data. Therefore, the DMS is not a significant tool by itself but requires a second level of software support, the SDSS, which treats the DMS as an interface to the ILLIAC IV Operating System. The responsibility of the DMS will be to maintain the gross aspects of information management such as file-record maintenance, retrieval and dissemination. It will be the responsibility of the SDSS to record in a set of tables, called Symbol Tables the particular structuring of all files, records and data elements throughout the system. The details for a suggested design of the SDSS may be found in Appendix D.

The SDSS will be designed to provide any level of accessing to a file contained within the system. It uses the DMS to perform its retrieval operations. It will give the user the ability to locate an arbitrary field or subfield, i.e., a location within a record where data items are stored, in order to locate a data item's value and provide that value to the user in ILLIAC IV core. It also recognizes the subfields of keys and can enter the identification file codes (a code which identifies the file which the associated record belongs to at the beginning of a key).

The SDSS will use a set of Symbol Tables to define the tree structure of data items in records. There exists one Symbol Table that defines the structure of all the other Tables. This is called the IMS Symbol Table. Each class of users for each data base, will have a Symbol Table that is considered to be part of the IMS Symbol Table. The first entries in the tables are a list of alpha-numeric names for all the different data items available

in a data base. By maintaining information, such as, what level in the tree a data item resides, the data item's parent and sisters, whether or not it is a key, etc., the structure of each file is uniquely defined. Whenever the DMS retrieves a record to be examined, the SDSS looks at the keys to determine to which file this record belongs. Data may be accessed at any level by following pointers through the Symbol Table.

The SDSS will note any changes made to the entries in the Symbol Table and will invoke the proper routines to restructure the files containing these entries. Once this scheme has been debugged, complicated file dependent routines cannot cause information losses due to incorrect file recopying.

The attributes of all data items will also be specified in the Symbol Tables. Any system designed to handle arbitrary data bases must be able to process data in any form, e.g., vectors, matrices, symbolic coding, English text, etc. To allow complete flexibility for cross-referencing data, the data type "relational link" will be utilized. It is essentially a pointer that identifies a relation held between two nodes of the same or different data item trees. Algorithms will be developed to take advantage of such linking schemes to reduce the times to process certain queries. Appendix D.3 describes a proposed design for the Symbol Tables.

Any data items that require further semantic definitions will have pointers to a descriptor file. The descriptor file maintains several levels of English text that may be used to explain the meaning and uses of an item in a data base.

D. The Information Retrieval System (IRS)

The previous two subsystems, the DMS and the SDSS, have been designed to isolate the housekeeping functions of information retrieval from the users and designers of the IMS. These systems will provide languages to control their functions. The function of the DMS will be to supply in core to the SDSS a set of qualifying records for a request. The SDSS will then supply the specific data items in question to the IRS. It will be the function of the IRS to translate user queries, call upon the required mathematical and statistical routines to operate on the retrieved data, and answer the queries in various report forms. The details for an initial approach to the system design are found in Appendix E.

A general search and control language will be provided. The IRS will interpret the instruction stream and invoke the SDSS and Mathematical Computation System when necessary. The SDSS must be requested to save any intermediate files generated through the interactive mode. This is necessary because of the simplified nature of the first version of the operating system. Appendix E contains further details.

E. The Mathematical Computation System (MCS)

The IMS will allow researchers to retrieve data and, without additional intermediate steps, call mathematical and statistical computation routines to perform analyses on the data. The Mathematical Computation System (MCS) is made up of several subsystems: the Statistical System, the Linear Programming System, and the Modeling and Simulation System. Appendix E suggests an overall interface design between these systems which would allow their development to proceed independently but which will allow easy integration into the IMS so that they may form a highly interactive system.

F. Using the Information Management System

The ease with which researchers could use the IMS should be emphasized. A hypothetical example will be used to demonstrate this system's ability to be used for research.

Suppose that an urban sociologist has recorded data on several hundred variables relating to the social and economic factors of a large city. Perhaps he wishes to create a model which predicts the growth of crime as the city expands. He may begin his study by reading his data into the IMS. With the aid of a data manager, he would determine an efficient way to structure the data based on his proposed research requirements. Later he may want to change this structure. The change could easily be implemented by a few short commands to the IMS.

Once his data is stored he may wish to perform a correlation study to determine the most promising predictors. He would produce a set of instructions directing the IMS to retrieve all data to be studied and to form

a new file in the form of a matrix. The command to perform a correlation analysis and output the results on his remote terminal would be in the same instruction stream. Based on the results, he could select the data associated with the variable he would like to study further and reduce the original matrix to a smaller one as input to a regression program. Based on the equation returned and the standard errors, the sociologist may select several variables to be multiplied together to create a nonlinear model. Commands to the system will combine these observations and resubmit the new matrix to the regression program.

If an equation is found to his liking, he may wish to randomly generate data to test out his model. All intermediate generated files could be saved for future comparisons.

It should be noted that this entire exploration could have been done with one set of instructions utilizing the conditional branching commands. Complex operations like these could be accomplished in a short turn-around time because of the computational speed of the ILLIAC IV. Research could proceed at an accelerated rate and the researcher would not have to worry about the details of data handling and computer methodology.

IV. List of Tasks

The following list is a brief outline of the tasks to be performed to implement the IMS. Once task (3) has been completed, we will have a basic information retrieval system which can begin processing data, although the full power of the system lies in the completion of the IRS and the MCS.

- 1) Study and simulate methods of accessing data through ILLIAC IV.
- 2) Define the DMS and SDSS according to the findings of (1).
- 3) Implement the DMS and SDSS.

1, 2, and 3 provide the first level of implementation for the IMS.

- 4) Define the IRS.
- 5) Concurrently define the MCS.
- 6) Implement the IRS and MCS as one system (i.e., the IMS).
- 7) Test and evaluate the IMS on an existing data base.

V. The Use of an Existing Data Base for Testing and Evaluating the IMS

A. Introduction

Before establishing a production mode, the system should be thoroughly debugged and evaluated. Most systems will use or simulate some subset of a data base to test the system. Under such contrived situations, it is impossible to actually determine the areas of the system that need more work until real data bases are used. This is highly undesirable to the user and causes anxiety for the system designers.

For these reasons, an existing production data base has been sought. Dr. R. Cancro of the University of Connecticut, in cooperation with Dr. E. Laska and Dr. E. W. Logeman of the Information Science Division Research Center of the Rockland State Hospital, has secured permission to transfer the current state of the Rockland data base to the ILLIAC IV computer. This will provide a practical test of the design and implementation of the system as well as providing a valuable service to mental health researchers.

B. A Description of the Rockland State Hospital Data Base

The Rockland State Hospital data base is a file of information on the admittance and treatment of mental patients of six states in the northeastern U.S. This data base currently contains records on 120,000 patients. There are between 500 and 1,000 records of new admissions added to this data base every week. The data is a two-file data base with the master file being the patient record filed by location. The other, auxiliary file, is a file keyed only by patient number which is used to find the current location of a patient--either in one of the buildings of one of the hospitals or at his home. There are many different information segments within the master file. Some of these are the admission segments, the treatment segments, drug treatment and response segments, transfer segments, termination segments, and psychiatric evaluation segments. The basic structure of the file also changes periodically with the addition of new segments.

There are, on the average, approximately 500 characters per patient record in this file. This produces an active patient data base of approximately 5×10^8 bits. This is 50% of the capacity of the ILLIAC IV disk.

This data base is expected to expand by a factor of 10 over the next five years. The present Rockland Information System is not capable of searching the entire data base for research purposes. With the complexity and richness of the patient file, it is clear that there is a considerable amount of valuable information buried in this data base on the effectiveness of various drug treatments on different diseases and the effectiveness of other treatments on mental illnesses. It is desirable for researchers to be able to access this data base in a highly general way for the purposes of studying treatments. Currently, it is prohibitively expensive to determine the effects of drugs on a patient population over a short period of time. The only way in which this can be done is to take a small sample of patients and trace their histories. With these small samples, there are certain variations which simply cannot be noted. This is because the current Rockland System does not have direct access to the entire patient population. For example, if there is a particular mental disease which responds particularly well to applications of special drugs, unless one asks this question in advance of structuring the data, this piece of information would be too costly to obtain.

The ILLIAC IV Information Management System would allow data bases on the order of the Rockland State base to be extensively explored by professional researchers. This would allow doctors and other professionals to study the data base on the basis of "hunches"; that is, they could ask a single question about a given drug across a whole population of 120,000 patients. If they suspected that the drug has a particular set of characteristics and if the response indicated that their hunch was true, but might be true for a particular mental illness only, they could then follow this up with additional questions and find out exactly what the effect of this drug was on a particular patient population. They could also make tests which would differentiate between the effectiveness of the drugs between different treatment patterns, such as given in the morning or night. This type of research must generally be done on an evolutionary basis. A professional man's hunch in this case, if it can be followed up with a reasonable set of questions which can be formulated and answered in a fairly short amount of time, greatly enhances the ability to extract knowledge from these large-scale data bases.

A current problem in the research area is that the form of the question has to be highly specific and coded in a machine-dependent way. The researcher is forced to learn in detail the nature of the particular machine he is working with. He is also exposed to a tremendous amount of frustration when errors are made in coding the problem for the machine. Usually, only the most mundane questions may be asked and generally the studies require months to perform.

Appendix A: References

1. Parker, James L. "Data Management on ILLIAC IV," ILLIAC IV Document No. 206. Urbana, Illinois: ILLIAC IV Project, University of Illinois at Urbana-Champaign, (January 21, 1970).
2. Parker, James L. "A Logic Per Track Disk System Utilized for Information Retrieval," Ph.D. Thesis. Urbana, Illinois: Department of Computer Science, University of Illinois at Urbana-Champaign. (In progress as of October 1970).
3. Schuster, Stewart A. "A Statistical System for ILLIAC IV," CAC Document No. 2. Urbana, Illinois: Center for Advanced Computation, University of Illinois at Urbana-Champaign, (December 11, 1970).
4. Parker, James L. "The ILLIAC IV Statistical System," Proposal submitted to the Graduate College by the ILLIAC IV Project, University of Illinois at Urbana-Champaign, (February 1970).
5. Sameh, A. "Mathematical Programming System," Proposal submitted to the National Science Foundation by the Center for Advanced Computation University of Illinois at Urbana-Champaign, (December 1970).
6. Sameh, A. "Automatic Solution of Large Systems of Ordinary Differential Equations," Proposal submitted to the National Science Foundation by the Center for Advanced Computation, University of Illinois at Urbana-Champaign, (December 1970).

Appendix B: A Glossary of Terms

The following is a list of terms and their definitions which have been referred to in the course of this document. The list has been inserted because the uses of many data management terms are not standardized and they may be used here in a different sense than that in which the reviewer is accustomed.

IMS	Information Management System
DMS	Data Management System
SDSS	Symbolic Data Structuring System
IRS	Information Retrieval System
MCS	Mathematical Computation System
elementary data item	a name of an atomic piece of data
group data item	a name associated with several closely related elementary data item or several other group data items
data item	an elementary or group data item
record	any group of related data items
key	any data item which is used to identify a record
value	the actual data associated with a data item, i.e., characters, numbers, logical values
name	alpha-numeric string of characters
file	all the records that correspond to a single set of associated keys
data base	any set of files
hole element	a bit string that contains no information, sometimes used by the DMS where data elements have been deleted
key element	a bit string that contains a key, used by the DMS
record element	a bit string that contains a record, used by the DMS
data element	several key elements and the associated record element
subfield	refers to the actual bit location of an elementary data item value in a record element

field	a collection of subfields or other fields
information element	a hole, key, or record element
garbage collection	the grouping of smaller hole elements into larger ones
Symbol Table	a table that defines the structure of data items within records of a user's data base
query	any sequence of instructions to the IMS
elementary question	given the value of a data item, which identifies a subset of records of a file, what are the values of an arbitrary subset of data items from the identified records?
key mask	defines the relationships necessary to fulfill a match for a key
I/O buffer	one of a set of cyclic buffers in core used to receive incoming streams of data
processing buffer	a buffer used to save records as they are found during a general search
characteristic	a name associated with a specific boolean combination of data items and other characteristics whose values are restricted by some relation
SCL	the Search and Control Language for the IMS
format	the information that defines the structure of data element fields

Appendix C: A Suggested Design for the Data Management System (DMS)

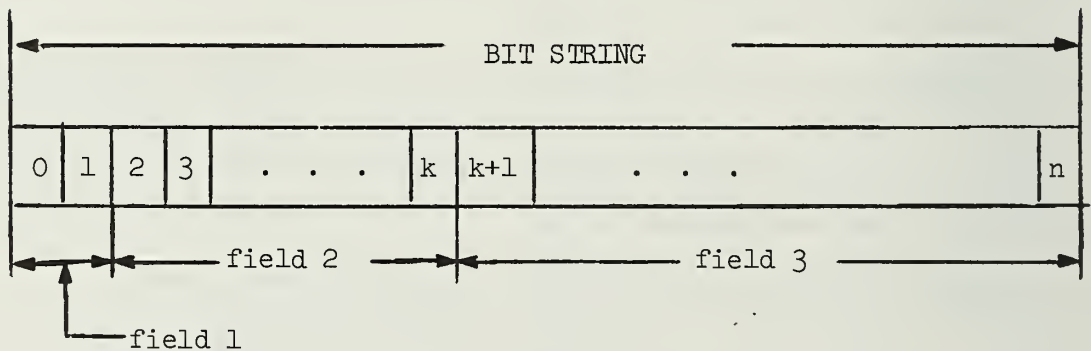
C.1 Responsibilities of the DMS

The DMS provides the lowest level support for the IMS. The following is a list of responsibilities assumed by the DMS.

- 1) It must have a preprocessing module to transfer card, tape, and other peripheral files to the laser memory.
- 2) It must transfer requested files between the laser memory and the disk and requested records between the disk and ILLIAC IV memory.
- 3) It must provide the capability of creating and interpreting key elements from specified key formats, and record elements from specified record formats, and associate them to create data elements for disk and laser dissemination.
- 4) It must provide the necessary set of operations to be performed on record, key, and hole elements, files, and data bases, e.g., reformatting, deletion, insertion, searching.
- 5) It must provide automatic disk maintenance, i.e., hole element collection that is periodically invoked by the performance monitor module. (The user has no need to know of the existence of such a routine.)
- 6) It must provide a simple language to specify the operations to be performed by the DMS.

C.2 The DMS Information Elements

To obtain efficiency and ease of data handling, the DMS requires that all of the data it processes exist in a highly simplified form. It will be apparent that any data structure can be represented this way. An information element consists of a bit string in which there are two fixed length fields and one variable length field. The first field is a two-bit indicator of the type of information that exists in the third field. The second field specifies the length of the variable length third field. To the DMS, the third field is an arbitrary bit pattern. Figures 3 and 4 are diagrams of information elements.



field 1: two indicator bits; specifying to DMS the 3 types of information elements, see Figure 4.

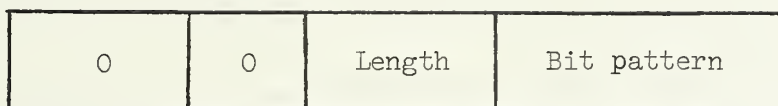
field 2: binary integer indication of number of bits in field 3

field 3: arbitrary bit pattern

Diagram of a DMS Information Element

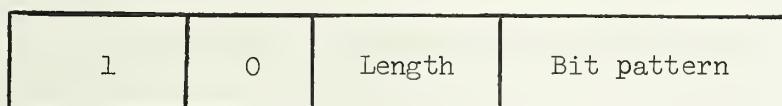
Figure 3

1. hole elements



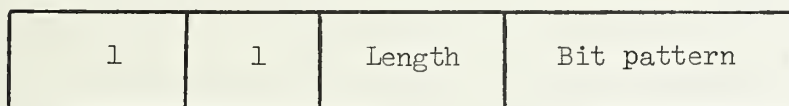
The bit pattern contains no information and is, therefore, called garbage. It indicates that this information element is available for key or record elements.

2. key elements



This pattern represents a key and its value. A key is any data item used to identify a class of related data items called a record.

3. record elements



This bit pattern represents a record which is a collection of structured data items.

The Three Types of Information Elements

Figure 4

Several key elements and a single related record element make up a single data element. Since a data element is composed of an arbitrary number of concatenated key elements, there exists a multiple keying potential for accessing any record.

The bit patterns in data elements are interpreted to be fields of data item values by the Symbolic Data Structuring System (SDSS) described in Appendix D. There are special fields in a key element's third field that identify, through a table, the key and record formats for the associated data element.

The syntax that governs the occurrence of hole, key, and data elements on the disk is shown in Figure 5.

`<global bit string> ::= <data base segment> <">...<">`, there will be 6 of these
`<data base segment> ::= <quadrant bit string>...,<">`, there will be 4 of these
`<quadrant bit string> ::= <parallel bit string> <">...<">`, there will be 64 of these
`<parallel bit string> ::= <parallel bit string>(<hole string>/<data element>)/NULL`
`<hole string> ::= <hole string> <hole element>/NULL`
`<hole element> ::= <ZERO BIT> <ZERO BIT> <length> <bit pattern>`
`<data element> ::= <key string> <record element>`
`<key string> ::= <key string> <key element>/<key element>`
`<key element> ::= <ONE BIT> <ZERO BIT> <length> <bit pattern>`
`<record element> ::= <ONE BIT> <ONE BIT> <length> <bit pattern>`
`<length> ::= <bit pattern>`, fixed length string that represents a positive binary integer
`<bit pattern> ::= <bit pattern> (<ZERO BIT>/<ONE BIT>)/NULL`

The Syntax that Governs the Occurrence
 of Hole, Key, and Record Elements

Figure 5

This syntax implies that the 10^9 -bit disk will be broken into 6 segments (2 SU's per segment, see Section II.A), each of which may contain one or more files from one data base--depending on the size of the files. Each of these segments is further broken into four starting locations where referencing can begin.

It is the physical association of keys and records to make data elements residing on the disk, along with a total disregard of actual record addresses, that eliminates large amounts of overhead experienced by other systems. Yet there is an average retrieval time of only 1/2 second to find an arbitrary record among a 10^9 -bit data base or just 1/20 second on the average to find an arbitrary record among a 10^8 -bit file (about the size of a tape).

C.3 Data Management on the Information Elements

C.3.1 Specifying the I/O Scheme to the I/O Subsystem of the Operating System

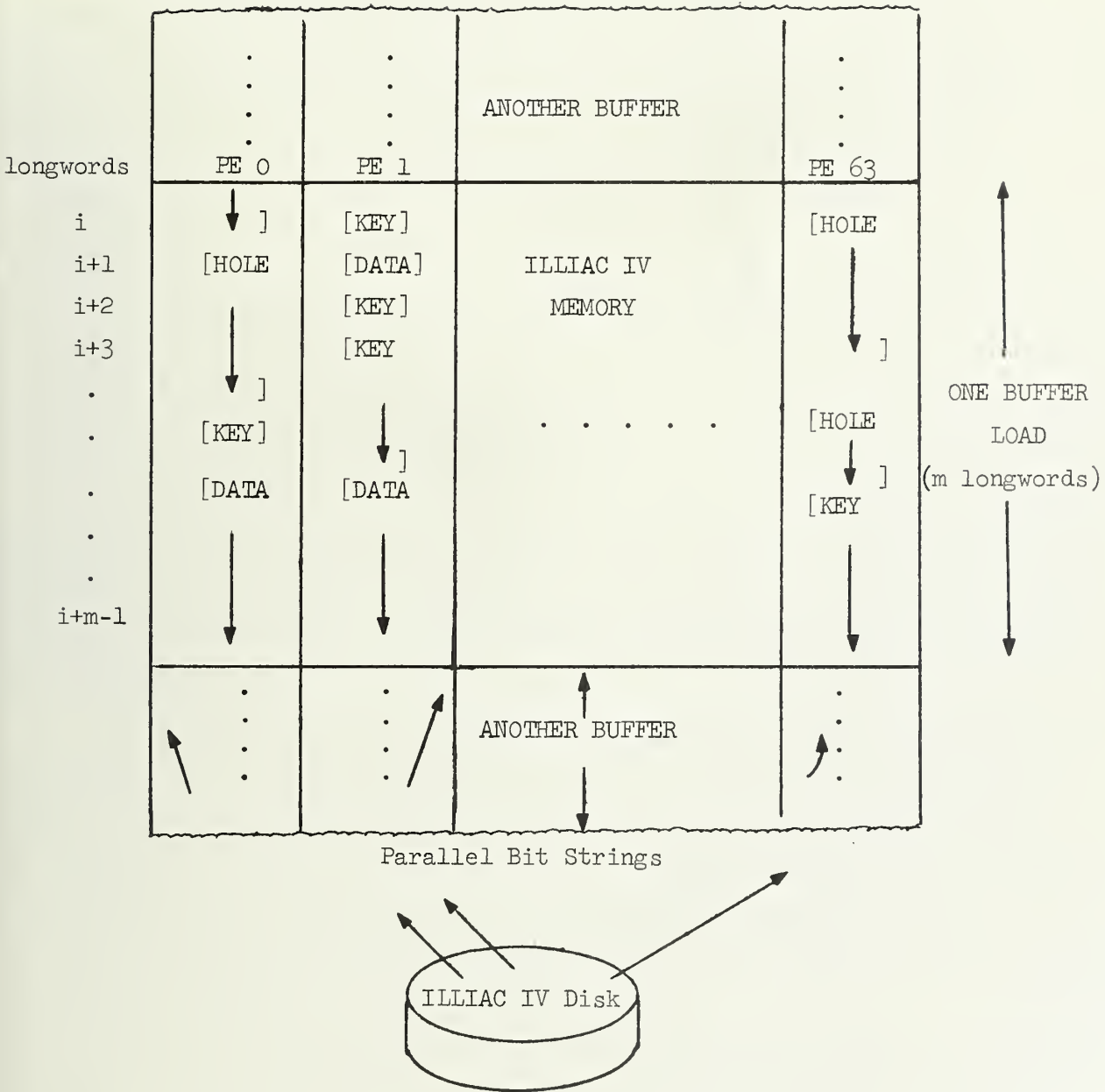
To specify an I/O scheme to the operating system, the data base segments (see syntax in Figure 5 of Section C.2) are broken at equal-length points to form disk blocks. Each block has been declared as a fixed-length record to the ILLIAC IV operating system and, as such, it is addressable. Each data base segment is declared as a fixed-length file; therefore, it is addressable by name. The DMS will issue an almost continuous sequence of I/O requests in order to stream hole and data elements of a data base segment into ILLIAC IV PE memories [one parallel bit string (see syntax in Figure 5 of Section C.2) in each PE memory] to be processed.

For simplicity, all hole, key, and record elements could be arranged to begin on word boundaries and exist in multiple word lengths. However, the information element lengths will still be specified in bits to determine the garbage bits in the last word of each element.

Figure 6 shows a typical information element distribution within a buffer after it has been read into ILLIAC IV memory. ILLIAC IV buffers correspond to disk blocks. The preprocessing module could reformat all incoming files before they are entered on the laser memory. This would avoid any need to column transpositions by the ILLIAC IV which is time consuming, as stated in II.B. It must be understood that a record's data no longer remains contiguous on the disk as in other schemes. This is because records are written "down" PE's while I/O proceeds across rows. Records, therefore, can only be referenced by the buffers in which they reside and entire buffers must be read to find a particular record (also, 63 other parallel bit strings are brought in).

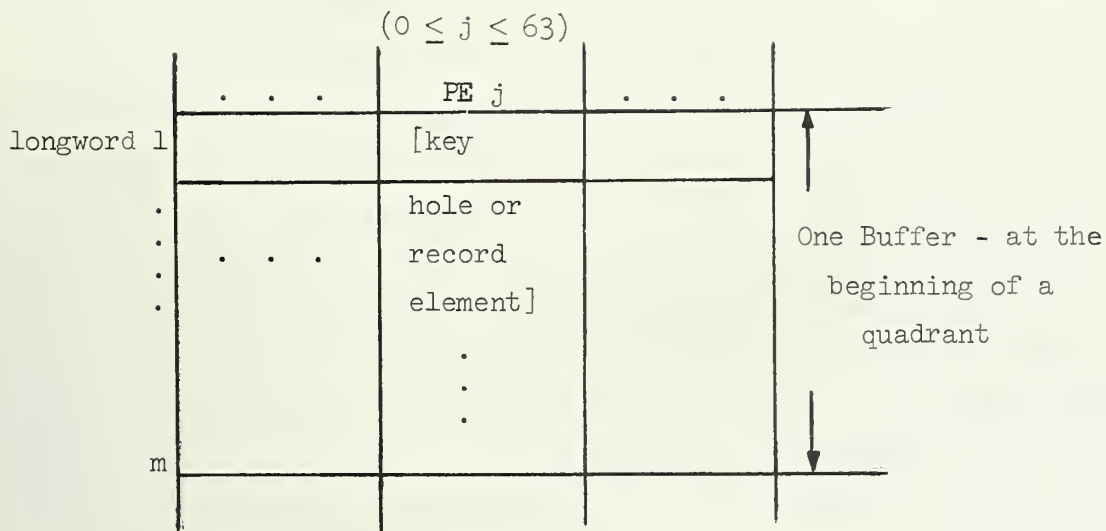
Blocks of rows (i.e., longwords) of ILLIAC IV memory are treated as a set of I/O buffers which accept the quadrant bit string buffers. This means that the operating system reads or writes one I/O buffer while the DMS is processing other I/O previously filled buffers.

This system is penalized by disk latency as well as the fact that all searches must be started at the beginning of a parallel bit string. We must have a known starting point to pick up the system elements where they



A Typical Information Element Distribution within a Buffer
 Figure 6

is a beginning of a hole, key, or record element in each parallel bit string; otherwise, we cannot tell what the bits represent. Thus, there would be an average disk latency of $1/2$ rotation to get to a starting point. The latency is reduced by introducing four starting points, i.e., quadrant bit strings, which divides each SU of the disk into four sections. The expected starting latency is now only $1/8$ rotation (5 ms.). Only buffers that begin on quadrant boundaries will have the beginning of an information element in each word of the first longword. An example of such a buffer follows in Figure 7.



A Beginning Quadrant

Figure 7

The end of a quadrant must have at least k bits, (unless an element runs to the end of the last buffer) see Figure 3, in each parallel bit string in order to declare a length hole. We cannot "write" information elements across quadrant boundaries.

I/O status is to be synchronized by the control unit (CU) via the I/O monitor. If I/O buffer processing exceeds a critical time, I/O must be stopped - an entire disk revolution (40 millisecc.) is lost until processing can begin at the same point. Such systems parameters as buffer size, buffer phasing on the disk, and the number of processable elementary questions per batch could be varied according to the particular query or user class in order to avoid I/O catastrophies, i.e., stopping I/O.

Each PE maintains the status of I/O buffer processing, i.e., current bit counts, information element type, keys on which the search is being conducted, and status on key comparisons being made. Searches and comparisons are executed on groups of 64 or less bits (word size) at a time.

C.3.2 Using the DMS

All requests processed by the DMS involve keys and files. These are specified by name to the SDSS which must check a Symbol Table to determine protection requirements, validity, structure, and location. If the location of a key is determined to be within a file that is not presently on the disk, the off-disk file maintenance module must locate it in the laser memory and pass it to the disk. This assumes that the files are formatted so that they can be directly fed into holed-out (available) buffers on the disk. It may be necessary to collect garbage within a data base segment before such a file request is made. This would create room for the file. New files should be pre-processed so that they will be compatible with the rest of the DMS file structures before being entered on the laser memory. This could be done by B6500 routines so that the ILLIAC IV processors are not wasting valuable time with mundane housekeeping - although some cannot be avoided.

Another system parameter is the maximum number of keys (in a boolean combination of keys) on which any single search may be conducted. Boolean combinations of different keys will be allowed as search arguments. Searching may be done on part of keys, e.g., the last name in a full name key or can be one of many relational forms. For example, equal to, greater than, not equal to and boolean combinations of these are possible searching conditions. Partial pattern matches could be performed on alphanumeric data. These conditions will be represented by a key mask to accompany each key element being used as a search argument.

Requests to the DMS will be translated into sequences of DMS instructions. The requests are viewed as a high-level language program that must be compiled into "DMS machine code" (i.e., DMS instructions) to drive the "DMS machine", that is, a "machine" which is simulated by ILLIAC IV code. The requests will come in some form of key element, key mask, and command triples. Some samples are:

1. INSERT KEY = X_DATA_ELEMENTS WITH A KEY = X AND MASK = (...)
2. READ RECORDS WITH KEY = AND MASK = (...) OR KEY = Z AND MASK = (...)

3. FIND HOLE WITH LENGTH = <1024 BITS
4. WRITE FILE = HOSPITAL FROM DISK TO LASER
5. COLLECT GARBAGE

When a match is located on a read request, for example, the record is moved from the I/O buffer into a processing buffer in the same PE, if possible. If a PE finds more records than it can accommodate, the I/O monitor is responsible for any necessary packing or routing. Pointers are returned to the calling modules to indicate where retrieved records are located.

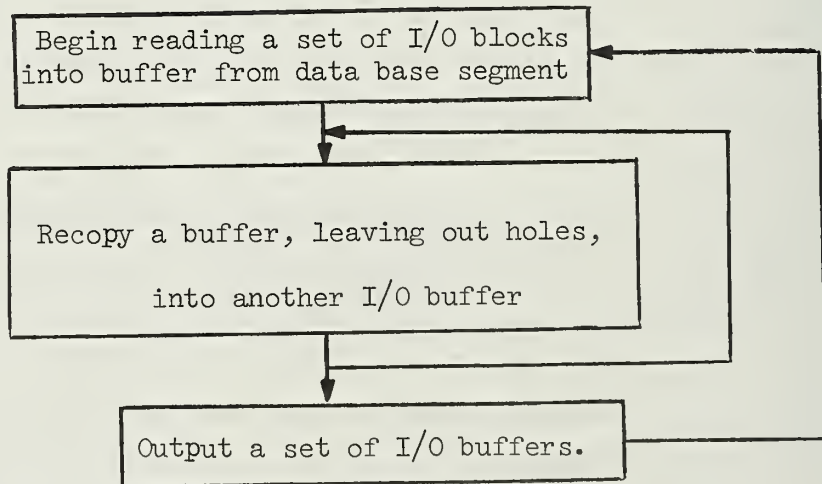
A curious result of this system is that, after some processing, files that are currently residing on the disk no longer have any physical resemblance to the general file concept of concatenated records since their records are scattered randomly throughout their residing data base segment. This happens when records are retrieved, processed (causing an expansion, contraction, or update), and then rewritten on the disk. The write request has issued a search for the first hole larger than the record which could be anywhere in the segment. This shuffling of records from files of the same data base would be advantageous for simultaneous multiple file processing. Searching time for finding two complementing records from different files would, on the average, be reduced.

Although records from different files from the same data base are completely shuffled in a data base segment, there is no deterioration experienced by the system and there are no additional routines required to unscramble records since the DMS is not concerned with where, who, or what is on the disk between off-file requests. Thus, two or more users may be simultaneously accessing the same file, but can be completely unconcerned with the increased file activity and complexity being experienced by the DMS.

When a file is to be returned to the laser memory, if a change has taken place, records in that file are brought together in consecutive disk blocks. The blocks become records for the laser file. A file write from ILLIAC IV disk to laser is then requested.

C.3.3 Garbage Collection

Disk information element maintenance will primarily be in the form of garbage collection--the collapsing of small hole elements into larger ones. It is desirable to collect garbage, often without the user's knowledge. This avoids deteriorating performance because of a lack of usable disk space due to the existence of several hole elements too small to be of use. This condition happens because whenever a write-a-record-element request is executed, the system finds the first hole equal to or larger than the record within the corresponding data base segment. If the hole is larger, a new hole is created at the end of the record that is smaller by the length of the record. Eventually, smaller and smaller holes become randomly distributed along the segment. (The garbage collection process is shown in Figure 8.)



Garbage Collection

Figure 8

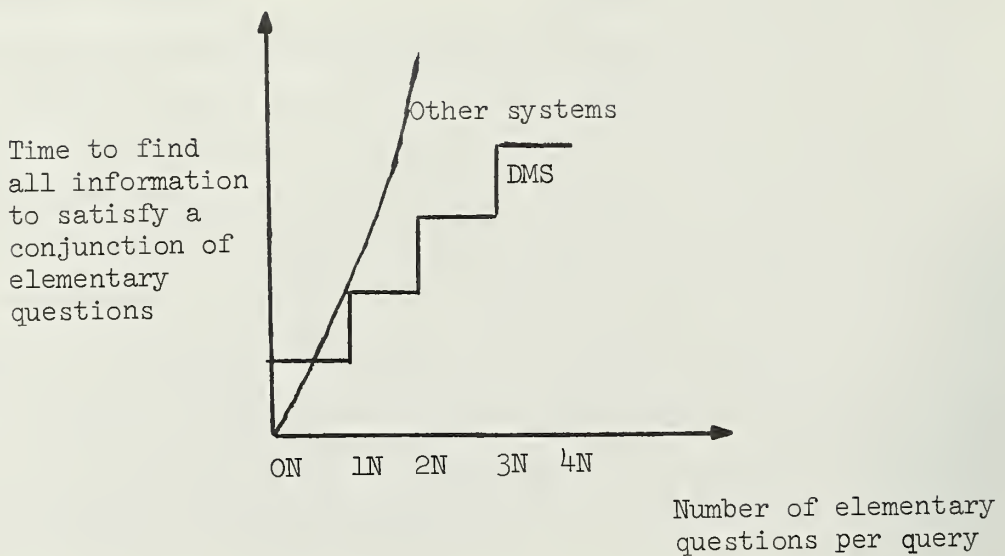
This tends to push holes to the "end" of the data base segments on the disk which suggests a good place to start looking for holes. The entire process can be completed in one pass over the segment.

C.3.4 Comments on Direct Accessing

If a direct access system would greatly increase efficiency for some special problem, then in one pass over the disk (one second or less) a temporary inverted file can be created. This file would tell what records are stored in each block by saving the key values of the records in a table along with all the corresponding addresses to these records. Addresses are denoted by the block number, PE number, and displacement from beginning of the block.

The main arguments against the general use of the direct access facility described above is that tables can grow very fast for large data bases. These tables must be paged in and out adding to the general queue of I/O requests. Furthermore, single direct access I/O requests are seldom experienced in any real query processing since we are generally concerned with a set of records.

It is not efficient to process single elementary questions that require the location of only one record. Elementary questions are of the form: Given a data item and its value (which identifies a record out of a set of records) what is the value of another data item in this set of records? Instead, it is to our advantage to batch the many elementary questions which are needed to answer any translated query, and retrieve all the required records on one data base segment pass. The system will not deteriorate continuously as the number of elementary questions increases. This is because the time to process one batch is roughly the time it takes to pass over the data base segment -- 1/10 second or less (which is adequate for interactive purposes). At some threshold sized batch, there is a discrete jump requiring two passes. However, for a Table driven system, the performance is approximately continuously related to the complexity of the query, i.e., the number of elementary questions generated. This is partly due to overhead costs for table maintenance due to repeated accesses to tables, searches through the tables, table maintenance routines, and I/O requests associated with table swapping and paging. This is not a linear relationship; procedure times increase faster than the increasing number of records they are processing. Also, the restricted queueing of I/O requests for individual records allows only a few accesses to be performed on a single rotation of the disk. Therefore, we can reasonably expect the following relationships shown in Figure 9.



The DMS vs. Direct Access Systems Performances on Processing Batches of Elementary Questions

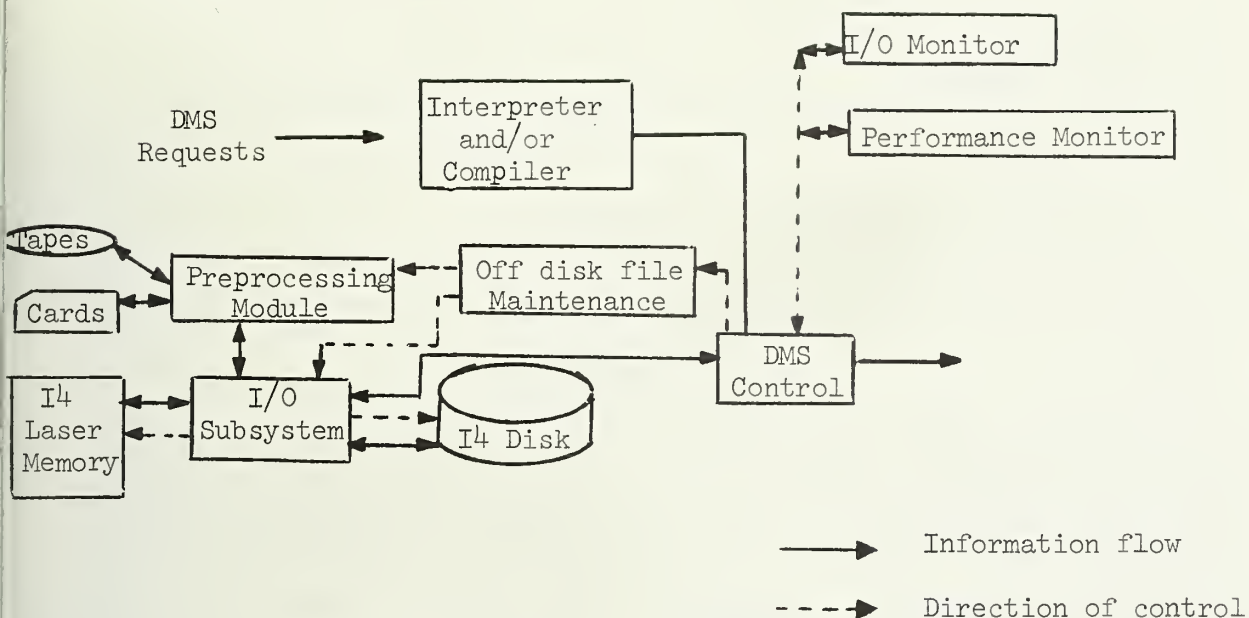
Figure 9

For most batching, the system would be expected to operate at some number of questions just below a multiple of N to optimize query answer times.

C.3.5 Comments on the DMS

Figure 10 is a diagram of the data flow and control between various modules of the DMS. Further analysis is required to determine which modules can be executed efficiently on the ILLIAC IV or should run on the B6500.

The system performance monitor module exists for two purposes. Firstly, it allows simplified system development by localizing the control over system parameters. Secondly, it provides for constant system evaluation on a production level and can suggest parameter value changes for different user requirements.



The Control and Information Flow Between

The Various Modules of the DMS

Figure 10

Some general statements can be made about the advantages of this system in comparison to other schemes. Garbage collection is not time consuming nor is the associated code long. It does not expose files to loss due to recopying -- associated with varying types of file maintenance routines. There is no need to segment large records into overflow areas. The performance does not deteriorate continuously as data activity increases. Vast assortments of large tables are not needed nor are the many different associated table maintenance routines. The system is conceptually simple which eliminates anxiety for users as well as for design implementors. Most important is that it provides a conceptually solid, yet simple foundation for more sophisticated levels of software to be built into the system.

Appendix D: A Suggested Design for the Symbolic Data Structuring System (SDSS)

D.1 Responsibilities of the SDSS

The following responsibilities are assumed by the SDSS:

1. The creation and maintenance of the structure of each record, i.e., the record format.
2. The input, retrieval, deletion, and testing of arbitrary data items within records.
3. The specification of any data items as keys.
4. Provide file security.
5. Provide minimum risk for information loss when restructuring or recopying files.
6. Allow for all forms of data and their various attributes.
7. Maintain sufficiently descriptive information on all data.
8. Allow for naming and retention of user defined "data characteristics" as boolean combinations of relational conditions on data items.

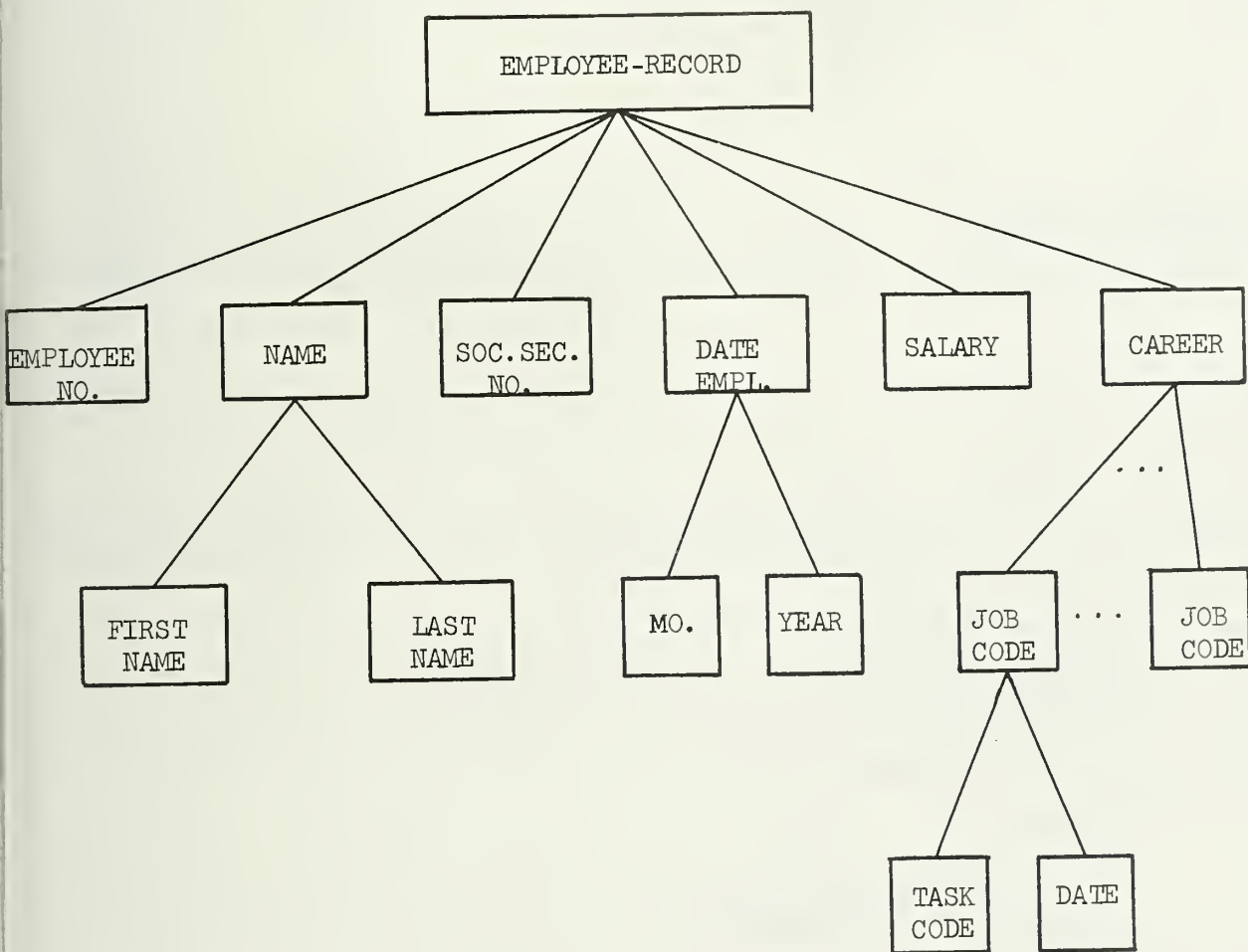
D.2 The Entities of Data Structuring

This system is designed to deal with the following entities:

1. Fields
2. Subfields
3. Elementary Data Items
4. Group Data Items
5. Records
6. Record Formats
7. Keys
8. Key Formats
9. Files
10. Data Bases

At the lowest level are elementary data items which are names of the least divisible pieces of data, such as EMPLOYEE NO, MONTH OF EMPLOYMENT, and DAY OF EMPLOYMENT. A group data item is a name associated with several elementary data items. Group data items may also be names for a collection of other group data items. For example, DATE OF EMPLOYMENT is a name for the

group of elementary data items MONTH OF EMPLOYMENT, DAY OF EMPLOYMENT, and YEAR OF EMPLOYMENT. EMPLOYEE RECORD is a group data item which includes several other group data items. When it is not necessary to distinguish between elementary and group data items, the term data items will be used. A record is any group of related data items that is structured in a hierarchical fashion. This structuring yields a tree data structure for records where each node represents a data item; the leaves represent elementary data items. Figure 11 is an example of the tree structure of a record.



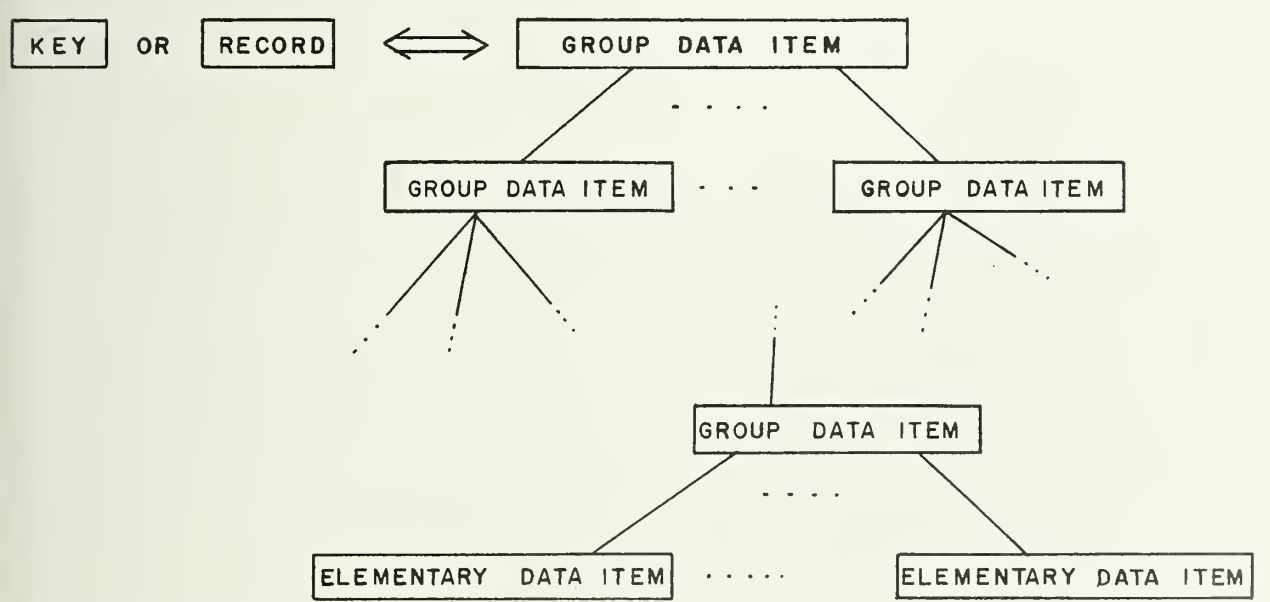
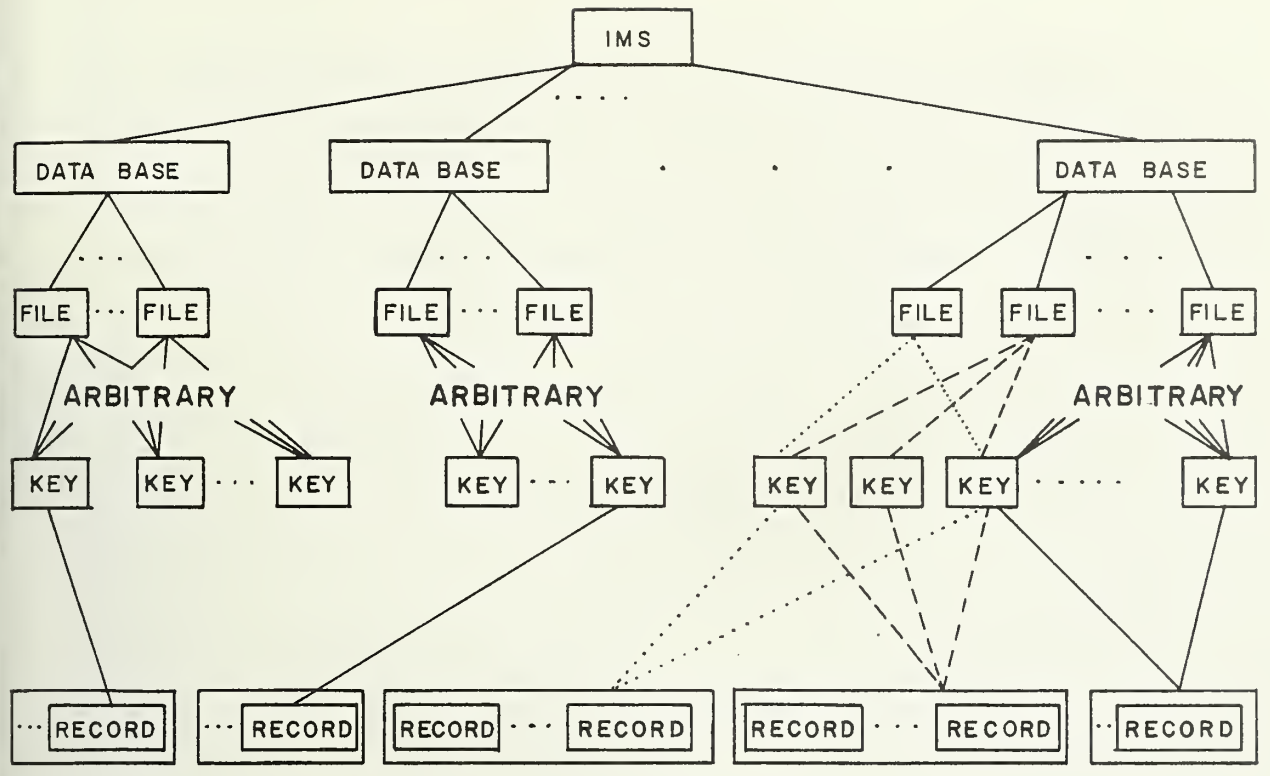
Example of the Tree Structure of a Record

Figure 11

Any data item may be used as a key. A key in this context is any data item which is used to identify other data items. The value of a key identifies a particular subset of records out of a set of records. The DMS structured the input data so that each record can be accessed by an arbitrary number of keys as decided upon by the people who are generating the data base. For example, a man's patient number may identify a whole group of information about him. This group could include name, address, insurance policies, drug types, and others. There may be several keys by which one wants to find his data. One would certainly want to find it by the man's name. It is also possible that this data might be located by his illness (but perhaps he has several). This last example is indicative of a more general form of key, i.e., a key which has several values per record in contrast with a key that has only one--such as the man's name. This general type of key presents no additional burdens on the IMS.

It will not be required for the user to know which data elements are keys. He will be able to specify any data element as a search argument when looking for the value of an associated data item. However, in structuring the original records, the user should choose keys in a manner which he feels would be the most efficient structure for his data with respect to searching. For frequent users, it would be beneficial for them to be aware that a generalized search on a data item which is not indicated to be a key could be relatively expensive. This is true because every record in every file containing this data item would have to be examined to determine the match. The DMS will permit several keys to be associated with a record. A file is then defined as all the data item values of all the records that correspond to a single set of associated keys.

A data base is defined as an associated set of files. Each class of users can maintain one data base. The IMS then can maintain several data base for different classes of users. The lattice-tree structure of the IMS is shown in Figure 12. Any further cross referencing of nodes is accomplished through elementary data items whose values have the data item-type link. This is elaborated on in Appendix E.



Lattice-Tree Data Structure of the IMS

Figure 12

Physically, the data structure of group and elementary data items are represented by fields and subfields, respectively. These are the actual groups of bits that contain the values of data items. The slicing of the key elements and record elements into fields and subfields is specified by key and record formats. An example of a key and record format is shown in Figure 13.

```
00 FILE_NAME IS PATIENT_FILE
  01 PATIENT_RECORD
    02 NAME KEY
      03 FIRST_NAME CHARACTER (VARIABLE)
      03 LAST_NAME CHARACTER (VARIABLE)
    02 PATIENT_NO INTEGER (9) KEY
    02 LOCATION
      03 BUILDING_NO INTEGER (2)
      03 ROOM_NO INTEGER (3)
    02 ILLNESS REPETITIVE
      03 ILLNESS_NAME CHARACTER (15) KEY
      03 DRUG REPETITIVE
        04 DRUG_NAME CHARACTER (15)
        04 DRUG_FREQUENCY CHARACTER (1)
```

Example Key and Format Declaration

Figure 13

Note the use of the data item types, REPETITIVE and VARIABLE.

Repetitive data items are those that exist in a variable number for each record but each repetition has the same structure. Variable data items are those that have variable lengths. Included as variable length data are those items that don't always have values for each record. Notice also, that any element containing a variable or repetitive element is, in turn, variable. These features allow for efficient packing of data into records by avoiding many null entries caused by allowing only fixed formatting of records. ILLIAC IV processing speed easily warrants this flexibility. There is plenty of time to pack, unpack, and pack again.

D.3 The Data Item Symbol Tables

The structure and names of all data bases, files, keys records, and data items will be found in a group of Symbol Tables. The IMS has a Symbol Table that names all the data bases and each data base in turn has a Symbol Table (i.e., each class of users has a Symbol Table). The IMS Symbol Table location is always known to the IMS system. It allows for the boot-strap location of a users' Symbol Table which is then placed in core. Each user Symbol Table is a manageable file and thus can be manipulated by the DMS, i.e., it can be modified with the same set of instructions as those used to modify other files. The structure of the Symbol Tables is known to the system by referencing the IMS Symbol Table while the structure of user data is known by referencing a Symbol Table.

An important feature of the system is that a modification of the Symbol Table implies that the structure of a file in the system has been changed. Changes in a Symbol Table force file modifications to take place. This means the Control Module of the SDSS will detect the changes to the Symbol Table, i.e., entries, deletions, corrections, updatings, etc., and will call the proper algorithms to carry out the implied modifications.

Every subfield (data item) of a user's data is described in a Symbol Table. Some of the entries indicate the length of the subfield, position in the data tree, type and attributes. Among other things, the character (alphabetic) name of the data item is present so that once a data item's subfield has been recorded it can thereafter be referenced by name. In this fashion, the user will never have to become "bogged down" with the computer jargon for field locations.

The table also allows the system to be aware of keys and data items which are associated with more than one file. For example, an EMPLOYEE NO. might be a key for an insurance file as well as a salary file. If one wanted to find the average insurance premium for persons of a certain salary, one would search the salary file for the salary range, extract the associated employee number, and then search the insurance file with the employee number to find the premium. Notice, that since the information is present in the symbol table, the system can make this cross file matching without the users

having to specify which files are needed. There was no need to mention an explicit sort and the related recopying which can monopolize time. This is because there is no need to order records on the disk.

Since a Symbol Table controls all structuring, easy and accurate file structure manipulations can be coded. Some changes require only the update of the pointers within each record and need no recopying. If a copy is required, it will be forced by a system call to the proper set of algorithms which, once debugged, will reduce risks of incorrect copying and losses of information for all users. Structuring of a new file will be done by indicating to the Symbol Table which data items are to be filled from old data and which from new data.

Since the Symbol Table is, itself, a managed file, it must also have a tree structure. Its structure is defined explicitly in the IMS Symbol Table. To change the structure of the Symbol Table, one only has to change the IMS Symbol Table. For example, perhaps a new subfield would be useful in the Symbol tables. It would be entered by name into the IMS Symbol Table which forces a change in the other Symbol Tables. Such a change in the Symbol Table may cause changes in the record structures. These changes should be carefully thought out, since every record in the IMS might be changed to carry out the restructuring. Although it's not a conceptually difficult problem, it could be very expensive in terms of time. The following is a list of the effects of changing various Symbol Tables.

- 1) The structure of the IMS Symbol Table never changes because it is fixed by the designer of the IMS.
- 2) Changes to, or additions of, entries to the IMS Symbol Table cause changes to the structure of the user Symbol Tables.
- 3) Changes or additions to entries of a user Symbol Table cause changes to the structures of the records in respective files of that user's data base.
- 4) If type (2) changes cause type (3) changes, then every record in every file of every data base in the IMS is changed.

The following is a description of each of the subfields in the IMS Symbol Table:

Name

This is a reference number for each table subfield which represents a column definition for user Symbol Tables or is the alphabetic name of a data base that exists in the IMS.

Entry Number

This is a reference number for each table entry which is actually the subscript displacement from the beginning of the table.

Size

This reserves a certain number of bytes for column definitions. If the name is a data base name, the current number of entries in that data base's Symbol Table are "sized". If the name is IMS, the size is the number of current data bases in the system. If the name is SYMBOL_TABLE, size is the number of columns in a Symbol Table. If the name is IMS_SYMBOL_TABLE, size is the number of entries in the IMS Symbol Table.

Parent, Level, Rank Among Sisters

All define the structural relationship of the column definition entries for the user Symbol Tables which then allow the SDSS routines to be used to restructure and make entries or deletions to the user Symbol Tables, as if they were files like any other files.

Base Address

If the name is a data base name, then this is the address of the base of the Symbol Table for this data base if it is in the ILLIAC IV memory, otherwise it is zero.

Figures 14 through 20 show examples of Symbol Tables, resulting key and record element formats, and associated data structures.

A packing scheme is used to reduce the number of pointers in a record format from having a pointer for every field. At each node of the tree the first entries are pointers to each variable length daughter node and the

beginning daughter node of each sequence of repetitive nodes; next follow all the fixed length items followed by the variable and repetitive items. Terminal nodes have no pointers since they would be null. In this scheme, there is one pointer for each occurrence of a variable or repetitive node.

The author understands that the packing scheme used in the examples as defined by the Symbol Table field definitions is not the most efficient -- with respect to the number of pointers required per record format. There exist trade-offs between adding more descriptor fields to the Symbol Tables, the number of pointers required in record formats, and the complexity of the tree climbing and record format packing algorithms.

The following is a description of each of the subfields in the Symbol Table that has been established by the IMS Symbol Table.

Name

The alphabetic name of the data item whose structural relationship with other data items is being described. In our examples, some of these have been PATIENT_NO, ILLNESS, and INSURANCE.

Entry Number

The reference number for each table entry is the actual subscript displacement from the beginning of the table.

Size

The length (in bits or bytes) for a fixed-length item. If the data items in question are variable length, such as a description, or if it is a repetitive data item, this entry is zero. Notice, any field which has a repetitive or variable length subfield is, therefore, itself, of variable length.

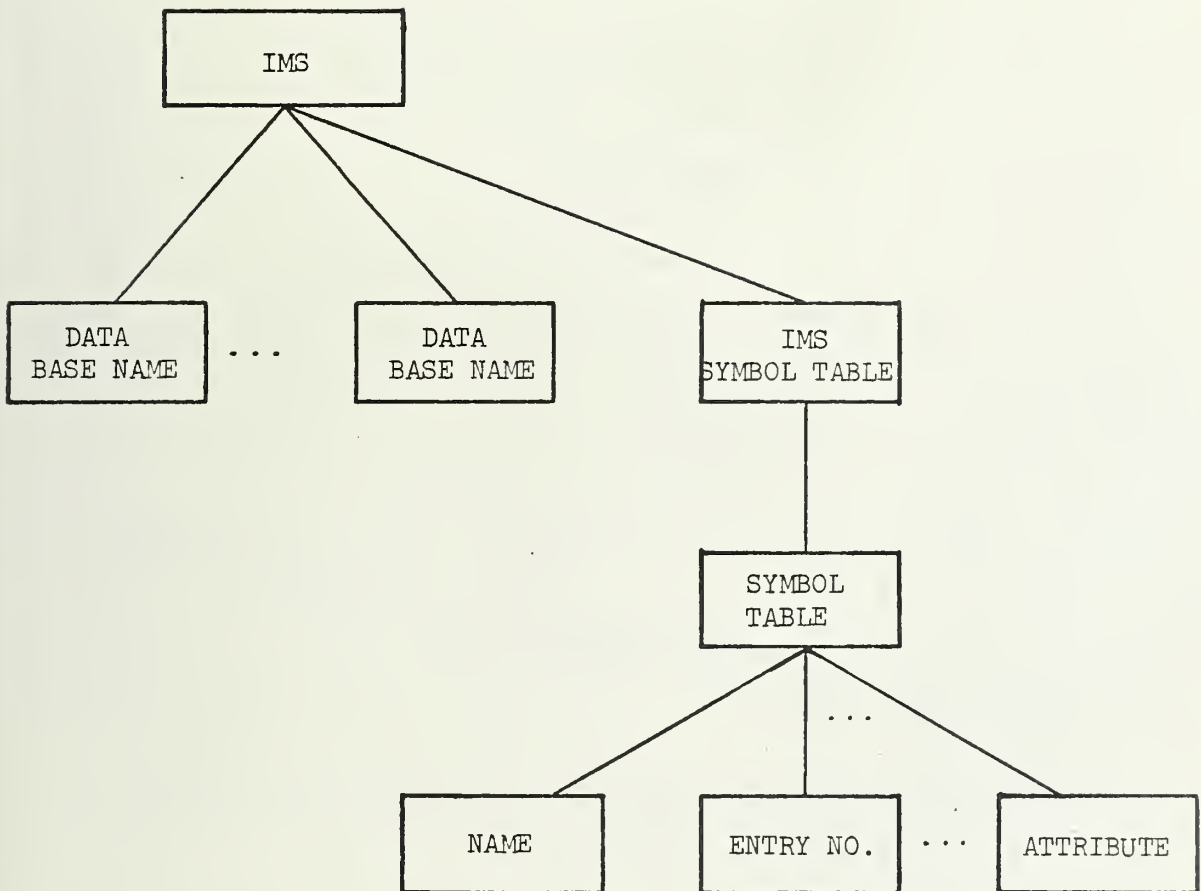
Repeat

A one bit indication as to whether the entry is a repetitive data item.

Name	Entry No.	Size	Parent	Level	Rank Among Sisters	Base Address
NAME	0		13	4	1	
SIZE	1		13	4	3	
REPEAT	2		13	4	4	
LEVEL	3		13	4	5	
PARENT	4		13	4	6	
ACCESS	5		13	4	7	
VARIABLE	6		13	4	8	
THREAD	7		13	4	9	
SORTED	8		13	4	10	
DISPLACEMENT	9		13	4	11	
NO_OF_POINTERS	10		13	4	12	
RANK_AMONG_SISTERS	11		13	4	13	
ENTRY_NO	12		13	4	2	
SYMBOL_TABLE	13	20	16	3	1	
X_HOSPITAL_DATA_BASE	14	17	17	2	1	
Y_EMPLOYEE_DATA_BASE	15		17	2	2	
IMS_SYMBOL_TABLE	16	25	17	2	3	
IMS	17	2	0	1	1	
PACKING_ORDER	18		13	4	14	
DISK_PRESENCE	19		13	4	15	
CROSS_REFERENCE	20		13	4	16	
UNITS	21		13	4	17	
KEY	22		13	4	18	
TYPE	23		13	4	19	
ATTRIBUTE	24		13	4	20	

Example of Entries in the IMS Symbol Table

Figure 14



The Data Structure Defined by the IMS Symbol Table

Figure 15

```
00 FILE_NAME IS PATIENT_FILE
  01 PATIENT_RECORD
    02 NAME
      03 LAST_NAME CHARACTER (15)
      03 FIRST_NAME CHARACTER (15)
    02 PATIENT_NO INTEGER (7) KEY
    02 ILLNESS REPETITIVE
      03 ILLNESS_NAME
      03 DRUG_NAME CHARACTER (15)
00 FILE_NAME IS INSURANCE_FILE
  01 PATIENT_RECORD
    02 PATIENT_NO INTEGER (7) KEY
    02 INSURANCE
      03 POLICIES REPETITIVE
        04 INSTITUTION_NAME CHARACTER (15)
        04 COVERAGE_TYPE INTEGER (2)
    02 TOTAL_PREMIUMS DECIMAL (7,2)
```

Example File and Record Format for the X_HOSPITAL_DATA_BASE

Figure 16

Name	Entry No.	Size	Repeat	Level	Parent	Access	Variable	Thread	Sorted	Displacement
PATIENT_FILE	0	0		0	14		1	0		0
PATIENT_RECORD	1	0	1	1	0		1	2		0
NAME	2	30		2	1			3		2
LAST_NAME	3	15		3	2			4		0
FIRST_NAME	4	15		3	2			5		15
PATIENT_NO	5	7		2	1			6		32
ILLNESS	6	30	1	2	1			7		0
ILLNESS_NAME	7	15		3	6			8		2
DRUG_NAME	8	15		3	6			1		17
INSURANCE_FILE	9	0		0	14		1	0		0
PATIENT_RECORD	10	0	1	1	9		1	11		0
PATIENT_NO	11	7		2	10			12		2
INSURANCE	12	0		2	10		1	13		2
POLICIES	13	17	1	3	12			14		0
INSTITUTION_NAME	14	15		4	13			15		2
COVERAGE_TYPE	15	2		4	13			16		17
TOTAL_PREMIUMS	16	7		2	12			10		9

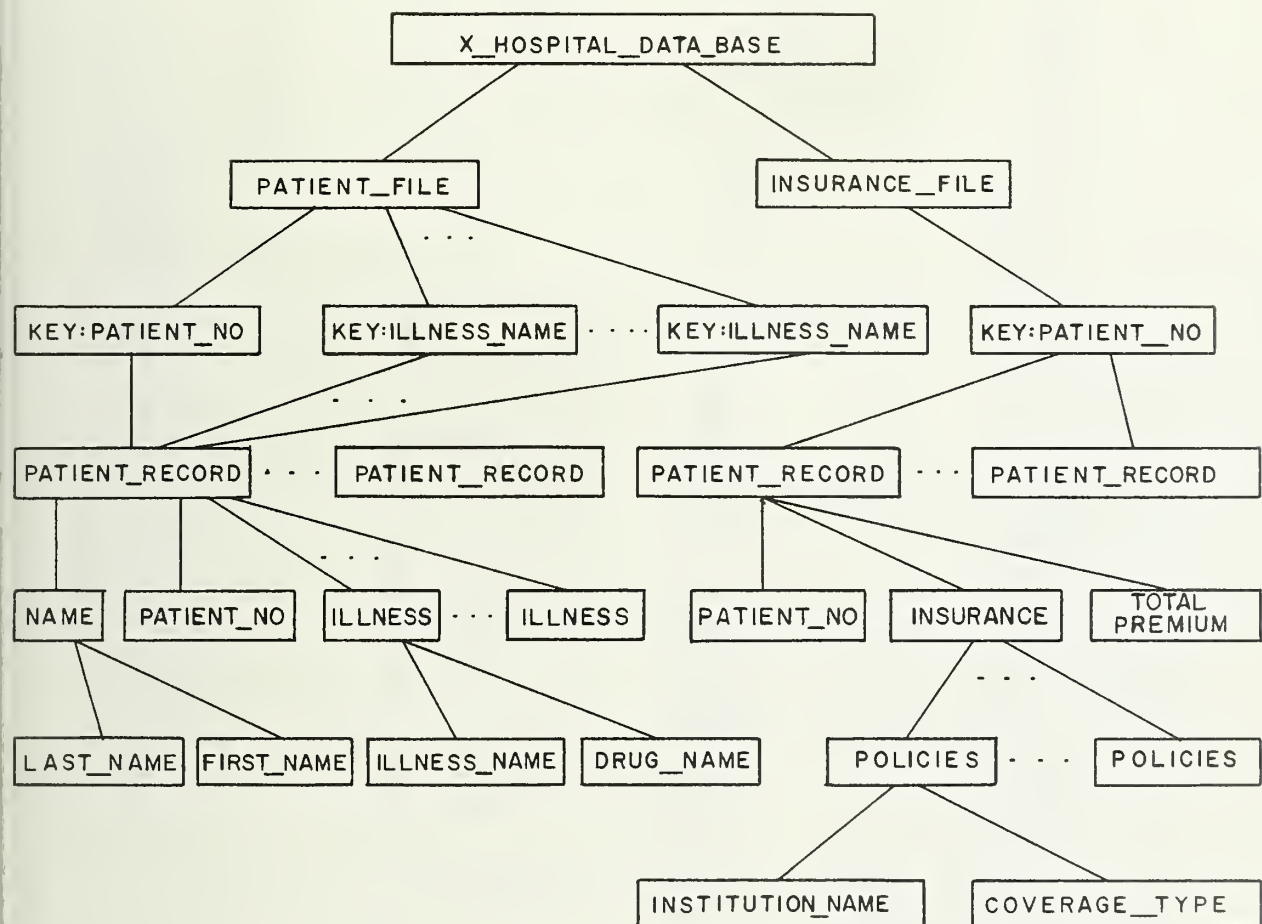
Example of the Symbol Table for the X_HOSPITAL_DATA_BASE

Figure 17a

Name	Entry No.	No. of Parts	Rank Among Sisters	Packing Order	Disk Presence	Cross Reference	Units	Key	Type Access
PATIENT_FILE	0	0	0	0					
PATIENT_RECORD	1	1	1	1		10			
NAME	2	0	0	0					
LAST_NAME	3	0	1	1					
FIRST_NAME	4	0	2	2					
PATIENT_NO	5	0	2	2		11		1	
ILLNESS	6	1	3	3					
ILLNESS_NAME	7	0	1	1				1	
DRUG_NAME	8	0	2	2					
INSURANCE_FILE	9	0	0	0					
PATIENT_RECORD	10	1	1	1		1			
PATIENT_NO	11	0	1	1		5		1	
INSURANCE	12	0	2	3					
POLICIES	13	1	1	1					
INSTITUTION_NAME	14	0	1	1					
COVERAGE_TYPE	15	0	2	2					
TOTAL PREMIUMS	16	0	3	2					

Example of the Symbol Table for the X_HOSPITAL_DATA_BASE

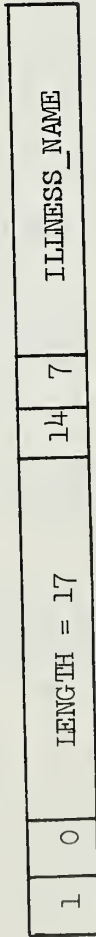
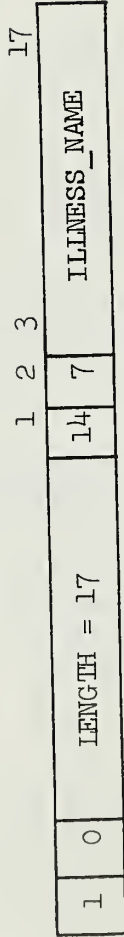
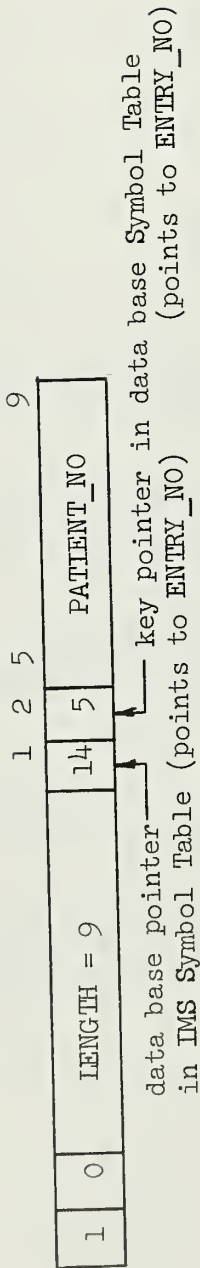
Figure 17b



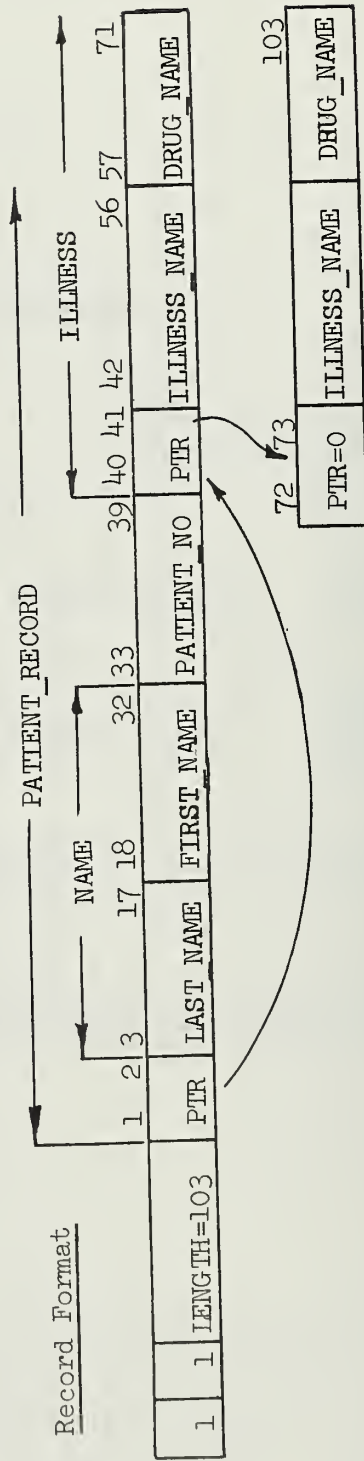
The Data Structure Defined by the Symbol Table

Figure 18

Key Formats



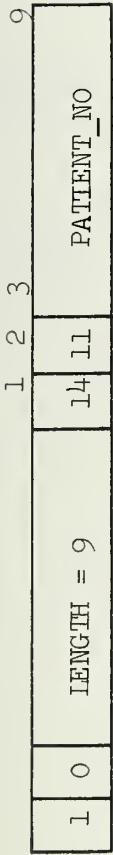
one of these formats exists for each ILLNESS_NAME per record



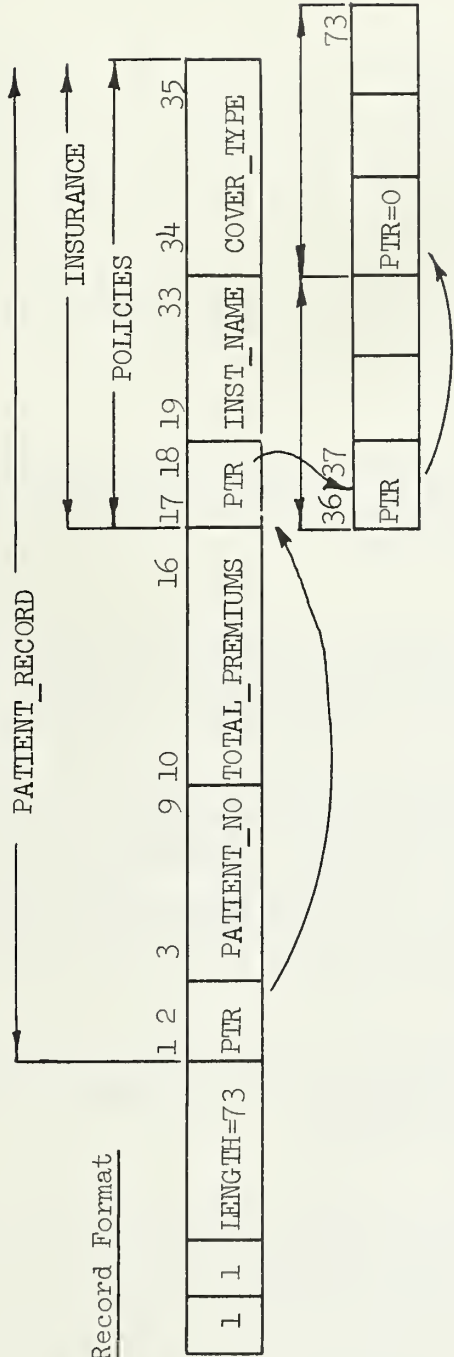
Key and Record Element Formats for the PATIENT_FILE

Figure 19

Key Format



Record Format



Key and Record Element Formats for the INSURANCE_FILE

Figure 20

Level

The level of this data item in the tree structure.

Parent

The entry number in the table of the parent data item for this entry. If the entry is a file name, this points to his parent (the data base name in the IMS Symbol Table).

Access

Contains security code information to protect the data from misuse by unauthorized users.

Variable

A one-bit indication that tell whether the entry is a variable data item.

Thread

The entry number in the table of the first daughter of the data item. If there are no daughters for this entry, the pointer will point to the next sister at the same level. If there are no sisters, it points to the next sister of the parent. If there is no such item, it will point to the next sister up another level. This entry allows simple traversal of the data tree.

Sorted

A one bit indication as to whether or not the repetitive data item is sorted.

Displacement

The byte displacement from the base of the parent field in the record format for a fixed-length entry or the displacement from the base of the parent field for the entry if this is a variable-length or repetitive entry.

Number of Pointers

The number of pointers at the beginning of this entry's record format field. There are pointers to each variable or repetitive subfields. The list of pointers is the first thing to appear in the field. All pointers are fixed-length.

Rank Among Sisters

A number which indicates how many sisters are to the left of this data item. The purpose again is for touring the tree.

Packing Order

The rank among sisters as they appear in the second formats, i.e., fixed-length items are shifted forward in their fields.

Disk Presence

A one-bit indication that tells, if the item is a file, whether it is now, presently on the ILLIAC IV disk.

Cross Reference

A circular linked list by entry number if this data item has the same name as another. These data items may be distinguished by their parents.

Units

A code representing the physical units of the stored data, e.g., feet, miles, or years.

Key

A one bit code that indicates if a data item is being used as a key.

Type

A code representing the data type, e.g., integer, real, symbolic, link, vector, matrix, or contextual.

Attribute

A code representing more information about the data type, e.g., matrix may be diagonal or real. Decimals may have the decimal point in a specific location.

D.4 Data Types and Attributes and Characteristic Definitions

The system will be able to process any form of data. Along with the usual numeric and character types, the system will provide for the following data types of data items:

- 1) vectors
- 2) matrices
- 3) symbolic coded
- 4) English text
- 5) relational links

Those data items requiring further explanations will be used as keys to reference a descriptor file. This file will contain detailed English text descriptions of the data item's semantic definition. Also entered in the descriptor file are the descriptions of various attributes of the data item. For example, in the NARIS system, the data item SOIL_TYPE may have the value, A, which certainly requires further explanation. There may exist several levels of detail for these descriptions since the descriptor file is a hierarchical file like any other. The report generator may use this file to generate its reports at several levels of detail. Explanations of data items may conditionally be put out with the report. New users of a data base may interactively use and learn the system while at the same time becoming familiar with the various data items.

The relational link data type is used to cross-reference different nodes within a tree, between trees of the same file or across files. This allows complete flexibility for performing complicated searches on a data base. For instance, in a natural resource data base, such as NARIS, which maintains data on a 40 acre tract basis, it would greatly aid search and trace routines to link together all the tracts through which a river flows. Tag bits can be

declared to identify the type of link being made -- to differentiate it from other links -- and the descriptor file can maintain an English explanation of the relation associated with a particular link.

It has been found that ILLIAC IV is an efficient tool for contextual pattern matching. This may allow simple content analysis to be performed on English textual data for limited document retrieval purposes, although this system is not explicitly designed for such activities. Pattern matching also allows the use of a thesaurus file so that equivalent data item names and values may be used in place of each other. For example, WHITE could be used equivalently as CAUCASIAN for the value of a RACE data item. A search on either key value would return the same records.

Another use of efficient pattern matching is the ability to allow partial pattern matches as a valid data item value identification. For example, suppose an inquirer of a mental health data base wants to ask a question about the distribution of patients among ethnic groups. The ethnic group of a patient may be stored as SPANISH AMERICAN. Suppose the researcher wanted to ask a question about SPANISH patients. He could then use a pattern matching operator if he wanted to admit the information contained in records that had the word SPANISH as any part of the ethnic group. He would not have to know all the different combinations of SPANISH heritage to specify his query.

In order to avoid writing the same specification for a frequently used boolean combination of relations on data, the notion of a data characteristic will be incorporated. A characteristic is a name associated with a specific boolean combination of data items and other characteristics whose values have been restricted. For example, the characteristics FERTILE and CORN-LAND may be defined as follows:

```
DEFINE THE CHARACTERISTIC FERTILE TO BE SOIL_TYPE A AND SOIL_DEPTH
    10 INCHES OR SOIL_TYPE B AND SOIL SLOPE LESS THAN 10 DEGREES.
DEFINE THE CHARACTERISTIC CORN_LAND TO BE FERTILE AND SOIL_ACREAGE
    GREATER THAN OR EQUAL TO 10 ACRES.
```

Researchers may build up very complicated characteristics and never have to be concerned with the details of the definitions again. The library file will maintain these definitions so that they can be used wherever data item names are employed. Further information on these characteristics can be found by name in the descriptor file.

The user should also be allowed to match names to questions that he would like to ask again -- but would not like to remember the entire language body of the request. These named routines may also be kept in the library file.

Any generalized information management system cannot foresee all the idiosyncracies of several users and their different data bases. It would be very valuable to provide the ability for users to program, in a high-level programming language (e.g., GLYPNIR or FORTRAN), routines tailored for their own purposes to operate on retrieved data. These routines can be named and stored in the library file for future reference.

Figure 21 is a diagram of the control and information flow for the SDSS.

Appendix E: A Suggested Design for the Information Retrieval System (IRS)

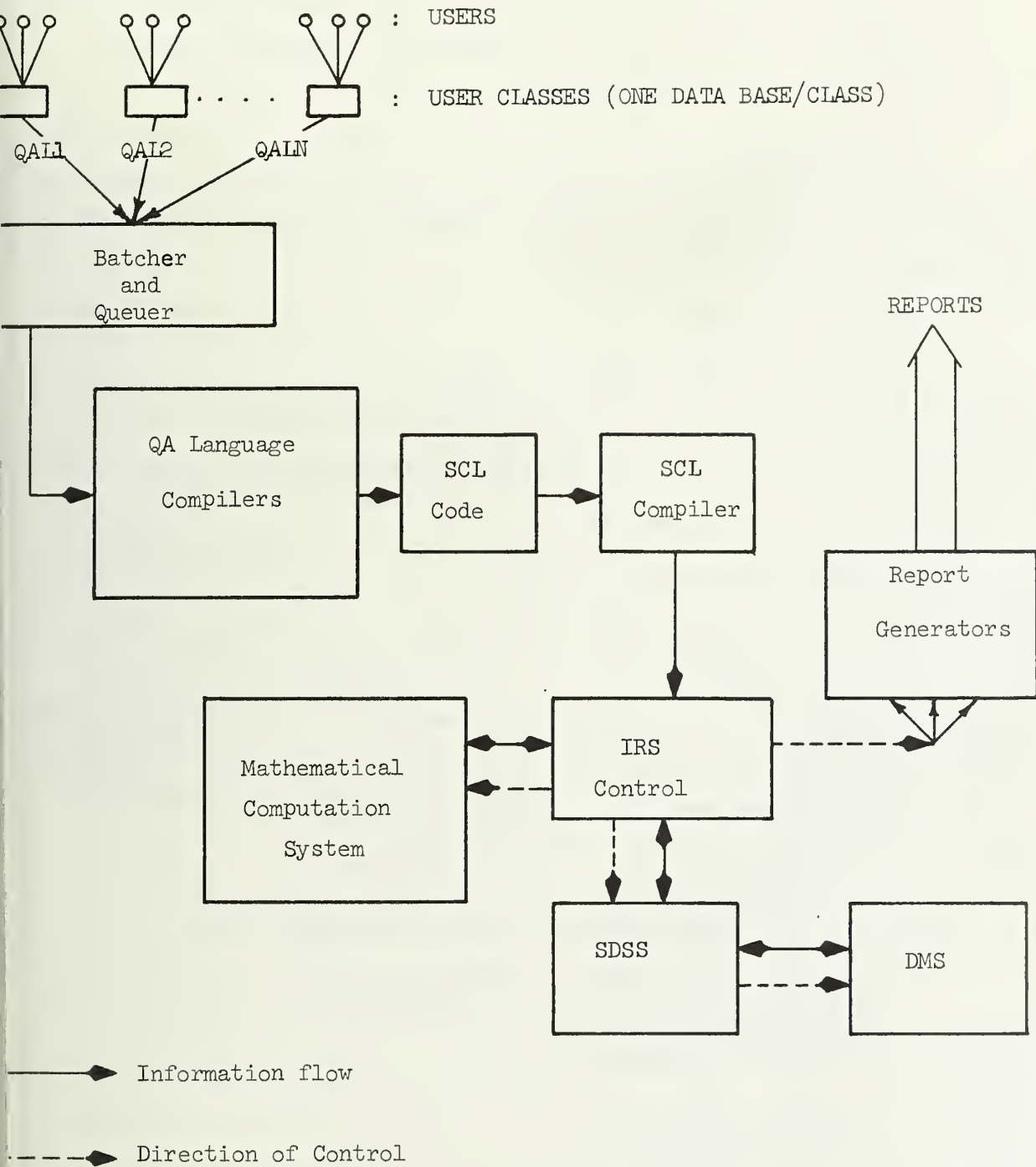
E.1 Responsibilities of the IRS

- 1) Provide a general search and control language (SCL) into which a set of data-base-dependent question-answering languages (QAL) can be compiled.
- 2) Provide, as part of each data-base-dependent QAL, a report generation facility.
- 3) Allow SCL to communicate with and control the Mathematical Computation System (MCS).
- 4) Provide for multi-user simultaneous query processing allowing an interactive mode.

E.2 The IRS Description

Figure 22 is a diagram of the control and information flow for the IRS. Several discrete languages have been suggested for conceptual purposes. The main language will be the SCL language; each QAL could be procedure names and associated parameters attached to segments of SCL code. Each class of users could then build his own language. The reason for the separation is that question-answering languages for different data bases would inherently be different. A QAL for a natural resource inventory system would be geographically oriented; questions would have a two-dimensional quality. However, the QAL for a mental patient system would be inclined to look for statistical relationships between the effects of several data items (variables) on other data items. The same philosophy is employed for the implementation of user oriented report generators. However, the emphasis would be upon the subset of report structures that would exist for all users.

The language must allow for conditional forms of instructions so that different actions may be taken, depending on the outcome of some mathematical routine or the value of a data item. Boolean combinations of characteristics could be used to define the scope (i.e., the subset of records) over which verb declarations can act. Verb forms would exist in classes that coincide with their subsystem purposes. For example, GRAPH pertains to the repor



The Control and Information Flow for the IRS

Figure 22

generator; FIND relates to the IRS; CORRELATE is associated with the Statistical System. The syntax for the languages in under study. Our direct link to ARPA-network members with experience in this area, such as Stanford University, MIT, and BB&N, should be of great help. We can experiment with their languages to determine a useful syntax.

The time-sharing and interactive capabilities are limited by the first version of the operating system. The first version of the ILLIAC IV IMS will have to provide these facilities in a round-about way. The sequence of events will be as follows:

- 1) Load the IMS;
- 2) Batch and queue several independent queries;
- 3) Process each query to their first report stage;
- 4) Report results, but save all intermediate files and variable values;
- 5) Release the IMS for other uses;
- 6) After a short time (on the order of a few seconds) return to step 1.

The batching facility and re-initiation of the system on a cyclic basis will simulate a time sharing system; the retention of all intermediate results will provide an interactive capability. The entire sequence, (1) through (6), should only be on the order of a few seconds.

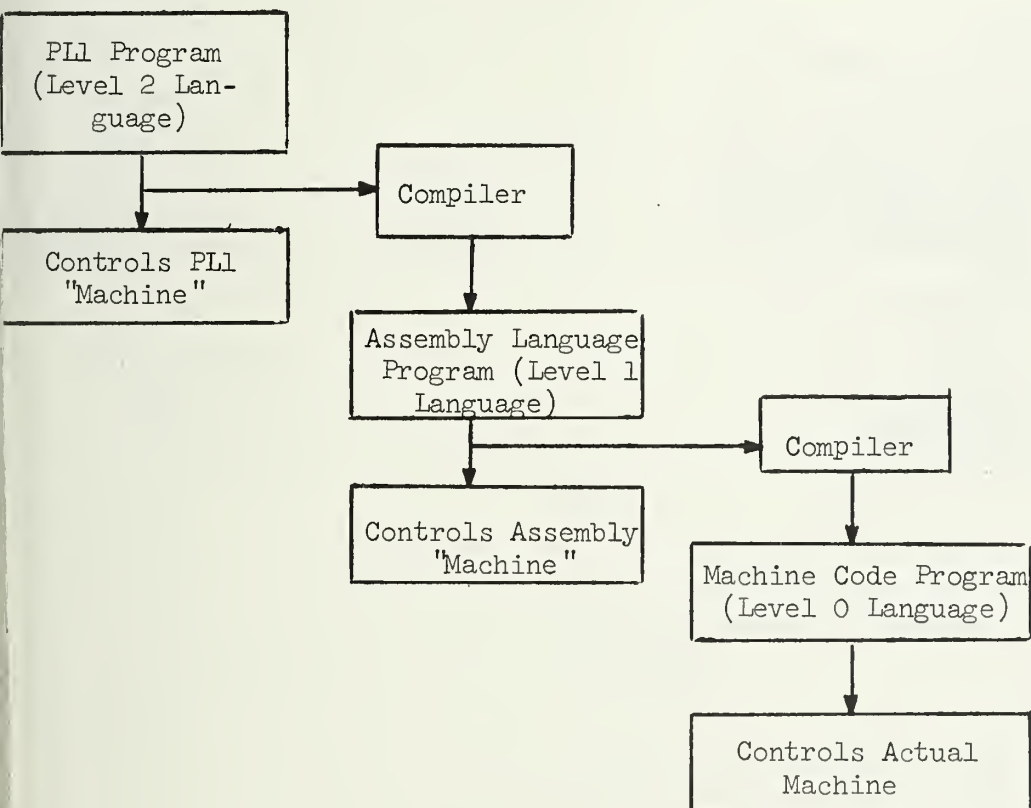
E.3 Interaction with the Mathematical Computation System (MCS)

The essence of the IMS is its ability to mathematically operate on several large independent data bases. The ILLIAC IV computer provides the capability for direct high speed processing of several files located simultaneously on its disk. Very intricate cross file searches can retrieve data and pass it to extensive computation systems residing in the MCS. In a matter of seconds, significant results can be reported to researchers where such operations would normally take hours (if they were economically feasible at all!)

References to these computational systems will be embedded within the SCL language, providing consistent accessibility to widely different systems. An hierarchical "language-machine" approach for complex subsystems

design is suggested to provide this systematic interactive ability. A by-product of this design is the ability for the computation systems to use the facilities of the IMS to maintain intermediate files. Thus, these systems may be developed without being concerned with details of complex data management. This considerably lessens development costs for the individual subsystems.

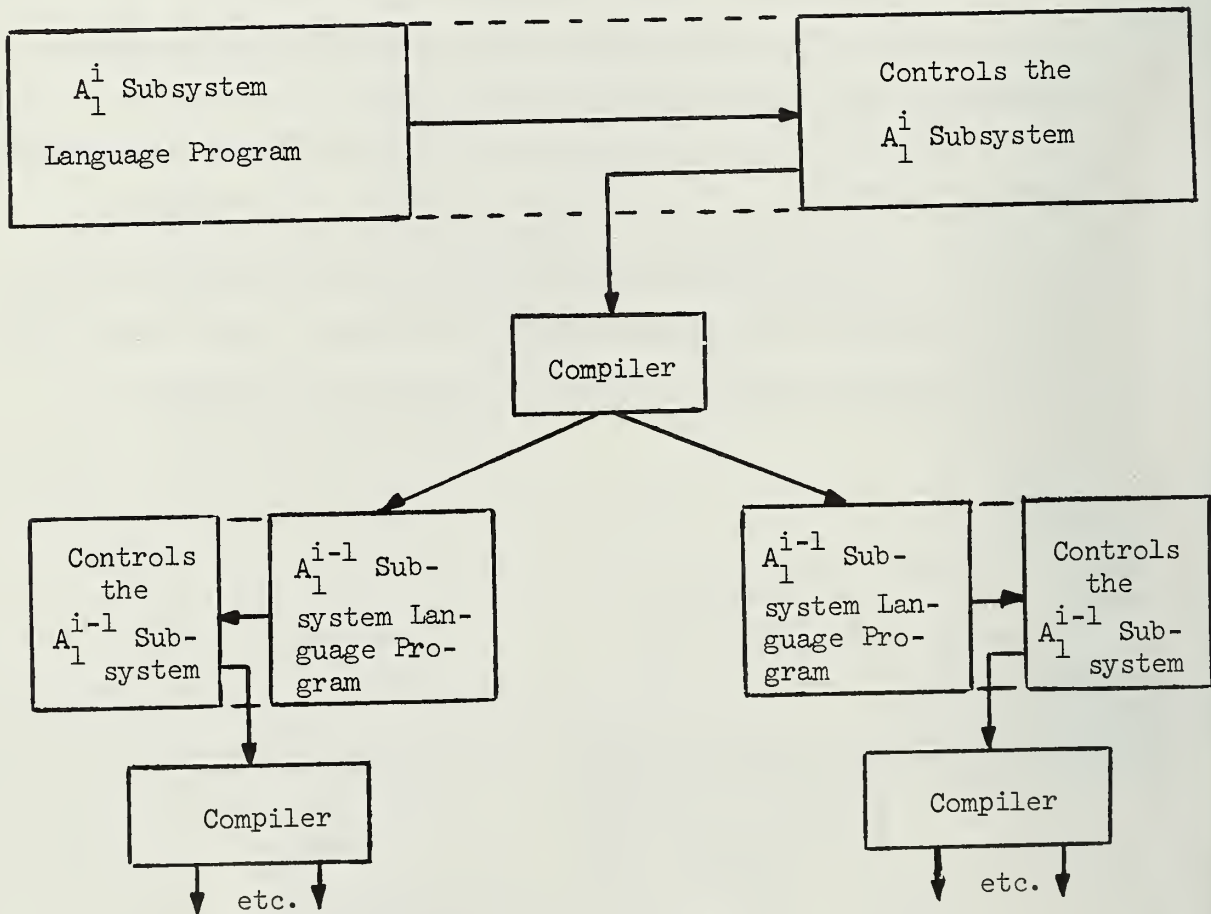
This approach is just an extension of the methods behind the philosophy of designing high level languages to run "high level language machines". These "machine" are then simulated by machine instructions on the computer doing the simulation. This allows programmers, for example, to view a PLI program as code written for a "PLI machine". An hierarchy is established because the PLI program may be compiled into an assembly language which views its machine as an "assembly language machine". This code is then assembled into hard machine code. Figure 23 illustrates the process.



A Language Hierarchy Established for
High Level Language Compilation

Figure 23

This structure applied to subsystem design is shown in the next diagram, Figure 24. Essentially it is exactly the same as the previous example except that "machines" are considered to be relevant subsystems and that any subsystem can generate programs for other subsystems.

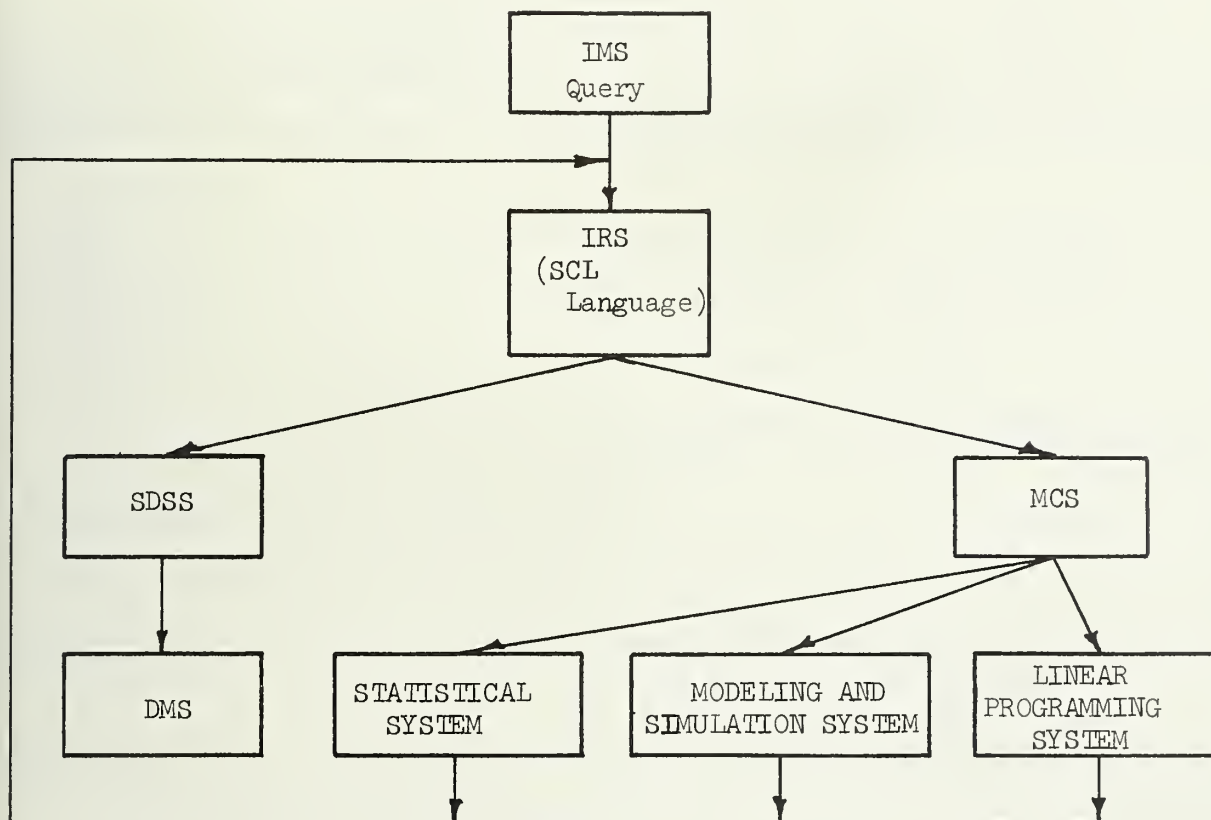


Hierarchy Language Control of Subsystems

Figure 24

Essentially, each program-system is a collection of subroutines with in a driver subroutine which accepts long sequences of parameters (i.e., a language), executes them, and can generate requests to other subsystems. There is no reason why a lower level system cannot call a higher system as long as the action does not cause an infinite loop.

The following diagram, Figure 25, shows how the language-subsystem approach applies to the IMS and MCS subsystems.



The Language Subsystem Approach Applied to the IMS and MCS Systems

Figure 25

An arrow implies a compilation step which is simply a conditional call of the subroutines in the next system.

Reiterating, this modular approach allows simple coordination of the total system with a uniform language development. It provides for independent (as far as the user is concerned) use of the different subsystems. Changes may be made to a subsystem without necessarily causing code changes in other systems. There is a disadvantage of having several compiler interpretation steps, but this is offset by the processing power of the ILLIAC IV system. Besides, the design is conceptually very simple, allowing easier development, implementation, and debugging.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author) Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
REPORT TITLE Information Management and Analysis System for ILLIAC IV		2b. GROUP	
DESCRIPTIVE NOTES (Type of report and inclusive dates) Research Report			
AUTHOR(S) (First name, middle initial, last name) Stewart A. Schuster			
REPORT DATE December 11, 1970	7a. TOTAL NO. OF PAGES 76	7b. NO. OF REFS 6	
CONTRACT OR GRANT NO. SAF 30-(602)-4144	8a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 1		
PROJECT NO. RPA Order 788	8b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)		
DISTRIBUTION STATEMENT Copies may be requested from the address given in (1) above.			
SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Rome Air Development Center Griffiss Air Force Base Rome, New York 13440		
ABSTRACT This document presents the preliminary specifications for the design and implementation of an Information Management System, capable of being integrated with an extensive Mathematical Analysis System to be implemented on the ILLIAC IV computer. The system would provide multiple keyed access to several large data bases and the capability of receiving instructions and questions expressed in a simple inquiry language. Mathematical and statistical routines will be interfaced through the Analysis System. There will also be a report generator which displays the responses to questions in forms easily interpretable by researchers.			

14.	KEY WORDS	LINK A		LINK B		LINK
		ROLE	WT	ROLE	WT	ROLE
	Information Retrieval (General) File Maintenance Searching Data Structures					

14.	KEY WORDS	LINK A		LINK B		LINK C	
		ROLE	WT	ROLE	WT	ROLE	WT
	Information Retrieval (General) File Maintenance Searching Data Structures						



UNIVERSITY OF ILLINOIS-URBANA

510.841L63C C001
CAC DOCUMENT\$URBANA
1-10 1970



3 0112 007263764