AD NUMBER: AD0467356

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to US Government Agencies and their Contractors; Administrative/Operational Use; 20 Jul 1965. Other requests shall be referred to Office of Naval Research, Washington, DC, 20350.

AUTHORITY

Per ONR memo dtd 7 Jan 1966

UNIVERSITY of PENNSYLVANIA

The Moore School of Electrical Engineering

PHILADELPHIA, PENNSYLVANIA 19104

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

A PROBLEM SOLVING FACILITY

Prepared for
Department of the Navy
Office of Naval Research
Methodology Division
Washington 25, D.C.
Under
Contract NOnr 551(48)

20 July 1965

Noah S. Prywes
Principal Investigator

Approved

John G. Brainerd, Director
The Moore School of Electrical Engineering

Moore School Report No. 66-02

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This is a technical report on the research accomplished in developing general problem solving methodology during the period April 1963 to July 1965 under office of Nord Research sponsorship.

A major outcome of the research has been the development of a Problem Solving Facility at The Moore School of Electrical Engineering, University of Pennsylvania. This report contains a complete functional description of the facility except for the operation of the console. It is described in a separate report titled "Control, Input, Output and Editing Functions in the Problem Solving Facility", by M. Wolfberg, June 1965.

The support for the implementation and operation of the facility has been provided from two sources. The equipment and consoles and the PDP-5 programs have been prepared with the support of the Information Systems Branch, Office of Naval Research under Contract NOnr 551(40). The system programs for the IBM 7040, which incorporate the new problem solving methodology, were supported by the Methodology Division, Office of Naval Research under Contract NOnr 551(48).

The reader interested in specific aspects of operation on programming should refer to separate reports and appendices as follows:

(1)The Information Storage and Retrieval is the subject of a report titled "A Storage and Retrieval System for Real Time Problem Solving," by H. Freedman (May 1965).

(2)Another part of the system (the CONSAT program) is described in a report, "Verification of an Information Retrieval System," by D. W. Alexander (May 1965).

(3)The Problem Solver system programs in the IBM 7040 are described in a series of separately bound appendices to this report, as follows:

Appendix A1: Summary of System Programs

Appendix A2: Input and Output

Appendix A3: Storage Allocation and Retrieval in Core and Disk

Appendix A4: The IBM 7040 Executive Program

Appendix A5: The MULTILANG Assembler

Appendix A6: The MULTILANG Interpreter

Appendix A7: Programming in MAP

These reports and appendices are available in limited quantity on request, and will soon be available through the Defense Documentation Center.

The report was prepared primarily by Richard L. Wexelblat, who has a also done the major design of MULTILANG System. The design and programming team consisted of:

D. Alexander
D. Evans
H. Freedman
C. Kapps
I. Kelly
R. Morton
A. Noetzel
S. Peltsen
A. van Dam
R. L. Wexelblat
M. Wolfberg

Faculty responsible for various parts of Project Supervision were:

J. W. Carr, III
H. J. Gray
N. S. Prywes

Many others have contributed to the design of the facility in various ways. A great deal of valuable experience has been gained from a system at the Aviation Supply Office, which was a forerunner of the facility described here. The cooperation and improvement suggested by the officers and staff of the Naval Aviation Supply Office are gratefully acknowledged. Discussions with Scientific Officers at the Office of Naval Research have provided a test of the ideas against the background of various real life, military problem solving situations. The design of the facility has been the subject of many discussions among the faculty and students of The Moore School and the many criticisms, clarifications and stimuli received are gratefully acknowledged.

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

TECHNICAL REPORT

A PROBLEM SOLVING FACILITY

1. INTRODUCTION

1.1 Objectives

The primary objective of the work described in this report is to enhance a computer-memory complex as an extension of man's ability in general problem solving. The major concern of the work has been the "intellectual" problem of man-machine communication, particularly as related to the areas of human problem solving and information retrieval. It is hoped that the aspect of efficiency of man-machine tandems is advanced as well.

The practical objective of the work described in this report was to set up a computer with a large memory for on-line real-time use to provide a tool to aid human problem solving. The major contribution of the work is the combining of the computational abilities of the computer with its ability to store, retrieve, and manipulate large masses of data.

Unlike conventional systems, the general problem solving techniques and Problem Solving Facility are designed to meet the needs of the "non-programmer" in addition to the conventional user of computer systems. The "programmer" or "analyst" by contrast calls on a computer system to perform algorithms and use operands, being familiar with their function and structure. The "non-programmer," on the other hand, is defined as approaching the system without being assured a priori of the existence of related information or programs, and especially of their structure. Prior to trying out a solution approach, a

"non-programmer" must relate the problem to the relevant information in the system's possession. The Problem Solving Facility is designed to perform much of the latter step automatically.

1.2 The Concept of a Problem Solving Facility

While automation has taken over drudgery, man has retained almost exclusively the roles of interpretation and representation of various stimuli, structuring of problem solution and decision making. However, recent advances in the understanding of human thinking processes and in relevant technology have tended to enhance greatly the role of the machine in problem solving. These advances have been in a number of areas.

Foremost of the advances has been the insight into human problem solving and concept formation[1]. This has been applied to the area of problem solving, initially undertaken by Newell, and Simon[11,7].

Advances in information storage and retrieval have vast implications on machine problem solution capability. These consist of processing natural language text, automatic indexing[28] and the classification and organization of information[21].

The advances in mechanical languages, assemblers, interpreters and compilers[2,4,17] are very significant.

Still another area of great impact is the mechanization of real-time, man-machine systems providing computer power as a "public utility" leading to efficient computer use[2] by man people simultaneously.

---

[1] Dating back to Greek philosophers and more recently in the history and philosophy of science as well as in logic. See for example [14,31] (in Bibliography).

[2] Work to that end has been done at the University of Pennsylvania since 1959. A system of this type has been in operation since 1962 as described in [32]. See also [6].

A more recent development consists of using display consoles for easier man-machine communication through aid in editing and through graphic display[9,29].

The concept of the Facility, incorporating some of these advances, consists at three system components as follows:

(i) An Information Storage and Retrieval component which is capable of storing large amounts of information relating to resources, intelligence, etc., as well as to structures and algorithms for problem solving.

(ii) Another component, the Executive, retrieves the information and executes algorithms in response to the user's communication. The Executive functions in the context of a computer utility where computing power is accessible to a large number of users, each of whom may simultaneously conduct a dialogue with the system compatible with the speeds of human thinking, formulation and representation.

(iii) The man-machine communication language is that component which the user may call for information and execution of algorithms in his natural language, compatible with human thinking in problem solution. Intermediate results may then be displayed to the user to control the progress of problem solution through additional input, resulting in a well-integrated joint man-machine effort.

1.3 Approach to the Facility

The undertaking to complete a system such as envisioned would indeed be a monumental task. Fortunately, digital computer equipment and programs possess the property of "growth." Namely, with a nucleus capability, programs and equipment may be added and reassembled in various combinations, sometimes recursively, to add new functions to the system. This report is devoted to a description of a nucleus of the above-mentioned components.

The research described in this report is regarded as the first step of a two-step experiment for the testing of two hypotheses.

The first hypothesis is that the use of the system will be particularly natural to the user. This hypothesis is based on features of the MULTILANG man-machine language which facilitates the information recall functions beleived to be central to human problem solving.

The second hypothesis is that through accumulation of knowledge, the system will be able to cope with increasing classes of functions and needs.

The work here reported constitutes the first step towards the testing of the above hypotheses. In the second step the system will be improved through application to real-life problems and through development of better equipment (especially the man-machine communication consoles).

1.4 Initial Capabilities of the Facility - Summary

The nucleus of the system consists of three components.

The Executive provides the capability for the interpretation of input messages from remote consoles which call for storage and retrieval of information and for the execution of algorithms retrieved. Other tasks of the Executive include the scheduling and management of system resources and work load.

The Storage and Retrieval System, where the growth of the system starts immediately, is where information on available resources, or other data problem solution structures and algorithms, are stored for further use. Thus, the system as a whole "accumulates knowledge." The information must be indexed so that it can be retrieved in response to messages describing new problems. The sharing of the information in the common store is technically one of the most demanding and, intellectually, the most intricate functions

of the system. The plans call for continuous enhancement of this capability through reindexing, i.e., reclassifying the changing organization of the total information[1].

The third component of the system, the Communication language, is a subject occupying most of this report. The system is to include a hierarchy of computer languages. At the lowest level are machine language programs which may be assembled by the use of a machine oriented symbolic language. (such as MAP for the IBM 7040). A higher level compiler language, such as ALGOL, uses the symbolic language assembler. To meet the needs of the "non-programmer," and primarily to provide for the information storage and retrieval function not included in existing compiler languages, still a higher level language, MULTILANG, was added. To denote its highest position in the hierarchy it was termed "An Executive Language." It serves as a means for translation between the human representation of the problem (using keywords and index terms) and the relevant programs and data on resources, etc.

These languages are to be used for outlining problem solution as well as on-line programming. Through MULTILANG, the user may search for appropriate keywords to formulate his problem, and iteratively refine such formulation.

Though the class of problems considered have been aimed at a military problem solver[1], it was intended to make it completely general and equally applicable to problems in management, design, teaching, etc.

---

1 The latter processes are not included in this report. Research toward these objectives, however, is in progress at The Moore School of Electrical Engineeri University of Pennsylvania.

1.5 Suitability of the System for Problem Solving

The manner of man-machine communication described in this report, emanates from the present understanding of human problem solving as represented by the heuristic problem solving model[7,19]. It is believed that this model describes the frequently subconscious processes of the problem solver. The suitability of MULTILANG for problem solving can therefore be illustrated in this context. This process is briefly reviewed below.

Human problem solving processes, as found commonly in every day work are viewed here as the pursuit of goal-subgoal structures. The human problem solver first interprets stimuli to formulate the problem and then specifies the overall objective. This is termed the goal. The attainment of the goal may be a complex endeavor and may be reached with greater ease when broken into intermediate objectives to be pursued sequentially, and relatively independently. These objectives are the subgoals. A variety of such goal-subgoal sequences may be appropriate under certain conditions, depending on the availability of resources and other intelligence. The entire goal-subgoal strategy can be described by a tree structure where the statement of the goal is represented by the apex node in the tree and the subgoals are represented by the other various nodes. The branches in the tree emanating from each node may thus be taken to represent a variety of alternative steps toward a solution. Attaining the subgoals in any path between the apex node and an end point may represent one solution of the problem and attainment of the goal.

The branches emanating from each subgoal-node represent a range of alternatives acceptable under varying conditions. "Exploring in breadth" is referred to theinvestigation of these alternatives. MULTILANG statements are so designed, that each explores one alternative representing a branch in the tree structure. This is provided through the capability of describing (by a

combinational logic expression using natural language index terms) what would be appropriate procedure or method to be employed, as well as through the description (in similar expressions) of the relevant parameters, operands or variables applicable to the situation at hand. A number of statements can therefore explore the corresponding number of branches. Thereafter a test may be conducted of the results where they may be evaluated according to acceptable criteria.

A number of statements may be assembled to form a procedure which can be stored in the system library and recalled through the use of appropriate index terms. Procedures may then be applied sequentially to "search in depth" through pursuit of the subgoals in the path from the apex to the outside tree branches, thus attaining the overall goal. Also, where appropriate, specific subgoals may be replaced recursively by a tree network necessary for their attainment.

1.6 The Information Storage and Retrieval System

The representation of a problem and its structuring is aided through the use of index terms associated with records in the system library. Successfully explored problem structures may be stored within the system for future reference. The terms may be assembled in a thesaurus and thereafter automatically classified. The user in pursuing appropriate terms may proceed from the more generic to the more specific terms. The records in the library may also be reprocessed, using automatic indexing techniques, to respond to specific approaches.

Information retrieval is an integral part of the pursuit of problem solution and has been accordingly incorporated into the nucleus of the system. The information retrieval system consists of a store of the information utilizing the MULTILIST technique described in Sect. 3.3 to simulate an associative memory - both in the mass storage memory (the disk), as well as in the high-speed core memory - and is found useful for a variety of functions.

Most important is the representation of a variety of associations among records in storage. Each means of association is defined by a key with all the records associated with a key connected through a list. A record appears on as many lists as the relevant associations, thus, the name MULTILIST is derived. Another purpose is to retain hierarchical breakdown through associations from the records representing the more generic entities to records representing more specific entities. This is applicable to organizations of information, both conceptual and practical, as in a command hierarchy or in describing mechanical decomposition of complex equipment into smaller parts.

The MULTILIST structure is particularly suitable for representing patterns, those of trees or networks, which are frequently useful during the solution of some problems as can be seen through the use of list techniques in a variety of applications, especially in simulating human thinking processes.[7]

The technique is also quite efficient, especially when applied to a large storage, providing for the retrieval of lengthy records from disk at the rate exceeding five records per second in some mechanizations, depending on the application. A number of storage/retrievals, and execution of limited length algorithms might be performed in one or a few seconds, with the appropriate cost in the order of a telephone call, charged to the user. The approach thus appears economically feasible.

1.7 Executive Functions

The MULTILIST structure is also used in the high speed core memory, which thereby becomes in fact, a simulated associative memory. This is based on a study[23] which showed how list techniques and indirect addressing can provide dynamic memory allocation and protection in a multi-user, multi-programming environment. Symbolic names are used exclusively in referring to program and

data segments. Within a segment, relative addresses are used. Segments are of variable length. Ultimately, a processor, incorporating these techniques in its design, will be necessary to fully realize the inherent potential. Basically, the techniques eliminate the need for protective keys and double address modification in executing each instruction, as is done in recent computers[3]. All the features of MULTILIST, as enumerated above in Sect. 1.7 also apply to core memory. In fact the disk storage is an extension of the associative memory in core.

1.8 On-line Programming

The system described herein was designed to be a novel combination of the information storage and retrieval described above and the on-line programming techniques. It incorporates the outline programming capabilities recently independently enumerated by Samuel[27] in defining a "General Purpose, Time-Sharing System" as follows:

> Two, or perhaps three, distinct types of time-sharing installations are already discernible. We can avoid misunderstandings if we draw a sharp distinction between these different kinds of systems. The first type, and the one meant more often when the term "Time-Sharing" is used, is the "General-Purpose System." A "General-Purpose, Time-Sharing System" attempts to provide each user with the full range of capabilities (except perhaps for speed) which he would have if he were the sole user of a general-purpose computer. The user should be able (1) to use all components of the computer system, (2) to work in any desired programming language, (3) to use programs and sub-routines which were written, originally, for other (presumably non-time-sharing) systems, with little or no change, (4) to wait for the results or to have the computation continue either in his absence or while he turns his attention to a quite different task using the same console, (5) to communicate freely with the computer both via a conventional typewriter and through some graphical device (a cathode ray tube with a light pen attachment, or the quivalent), (6) to store large amounts of data and program material within the system to which he can gain access as desired on a moment's notice, and finally, (7) to have the use of a large library of service routines and "debugging aids . . . There are in existence no systems which actually provide all of these services (the S.D.C. time-sharing system comes very close. . .), but this is the goal.

> A second type of system sometimes goes by the name of a "Dedicated-System," but a better name would be a "One-Language System." Here, no attempt is made to allow the user a variety of languages, and indeed he is constrained to use but a single language which may, for example, be simplified FORTRAN (as in IBM's QUIKTRAN . . .) or an even simpler language (JOSS, as used by the Rand Corporation . . .). In effect, the user is presented with a special computer which is literate in but a single language.
>
> Finally, we might distinguish a third class of systems in which the language of the computer is still further restricted and specialized to deal with a restricted set of problems, a common library facility, or a common data base. The SAGE air defense system was an early system of this type. The SABRE air-line reservation system is a more recent example, as are the many information retrieval systems that are being talked about.[1]

1.9 Report On the Use of the Problem Solving Facility

This report has been oriented to acquaint those personnel in a position to use the system to understand the capabilities and the overall aspects of the problem solving facility described herein. It consists of a description of the overall system and of the MULTILANG Executive language.

The overall system description is the subject of Sect. 2. It includes the organization of the equipment (Sect. 2.1) and the system programs. The use of the system and processing of messages composed at the consoles is described (Sect. 2.3) and illustrated through typical usage examples (Sect. 2.4).

The communication with the system is primarily through the use of descriptions, by means of which one may describe the programs and data that may be relevant to problem solution.

Descriptions, together with specification of arguments, may be assembled into MULTILANG statements. A statement consists of a variable number of descriptions, the first always referring to the operation to be executed.

---

1 Samuel, Arthur L. "Time-Sharing on a Multiconsole Computer," MIT, Project MAC Report MAC-TR-17, March 1965, pp. 2,4. References to the specific systems mentioned are S.D.C.[26], Quiktran [5], Joss [27].

A console message may consist of a single statement, or a number of statements assembled in a procedure.

The composition of MULTILANG descriptions, their purpose, syntax, and interpretation are the subject of Sect. 3. Section 4 describes the composition of MULTILANG statements and procedures.

More detailed system and programming information is provided in separate appendices and companion reports as enumerated in the Preface.

2. GENERAL DESCRIPTION OF THE PROBLEM SOLVING SYSTEM

2.1 Operation of the System

The Moore School Problem Solving System is composed of a large-scale digital computer, a disk file and a group of cathode ray tube consoles with keyboards. Figure 2.1 briefly illustrates the flow of information in the system.

user ↔ console ↔ local computer ↔ central computer ↔ disk file

Figure 2.1 Flow of Information

The system and its Executive language, MULTILANG, are designed for on-line use: a user sits before the console with a cathode ray tube for display, a keyboard for input and a teletype for hard copy; he types his calls for processes and specifications of data directly into the computer, which then executes retrieved programs upon retrieved data and communicates the results to him. Limited editing facilities are available at the console, allowing the user to modify recently typed lines or characters by deleting and adding on the cathode ray tube. Each complete page, as finished, is transmitted to a local computer which accumulates messages, edits and reformats them for the central computer. The central computer controls the solution of problems, stores information and programs the disk file while making use of programs and data previously on the disk. In a multi-user system, the local computer also has the duty of sequencing users and applying a priority control, interrupting the central computer whenever necessary. Computation results return from the central computer to the local computer and then to the console for display to the user.

A set of system programs controls operation of the system. These include an Executive Program for sequencing, execution, input and output, etc;

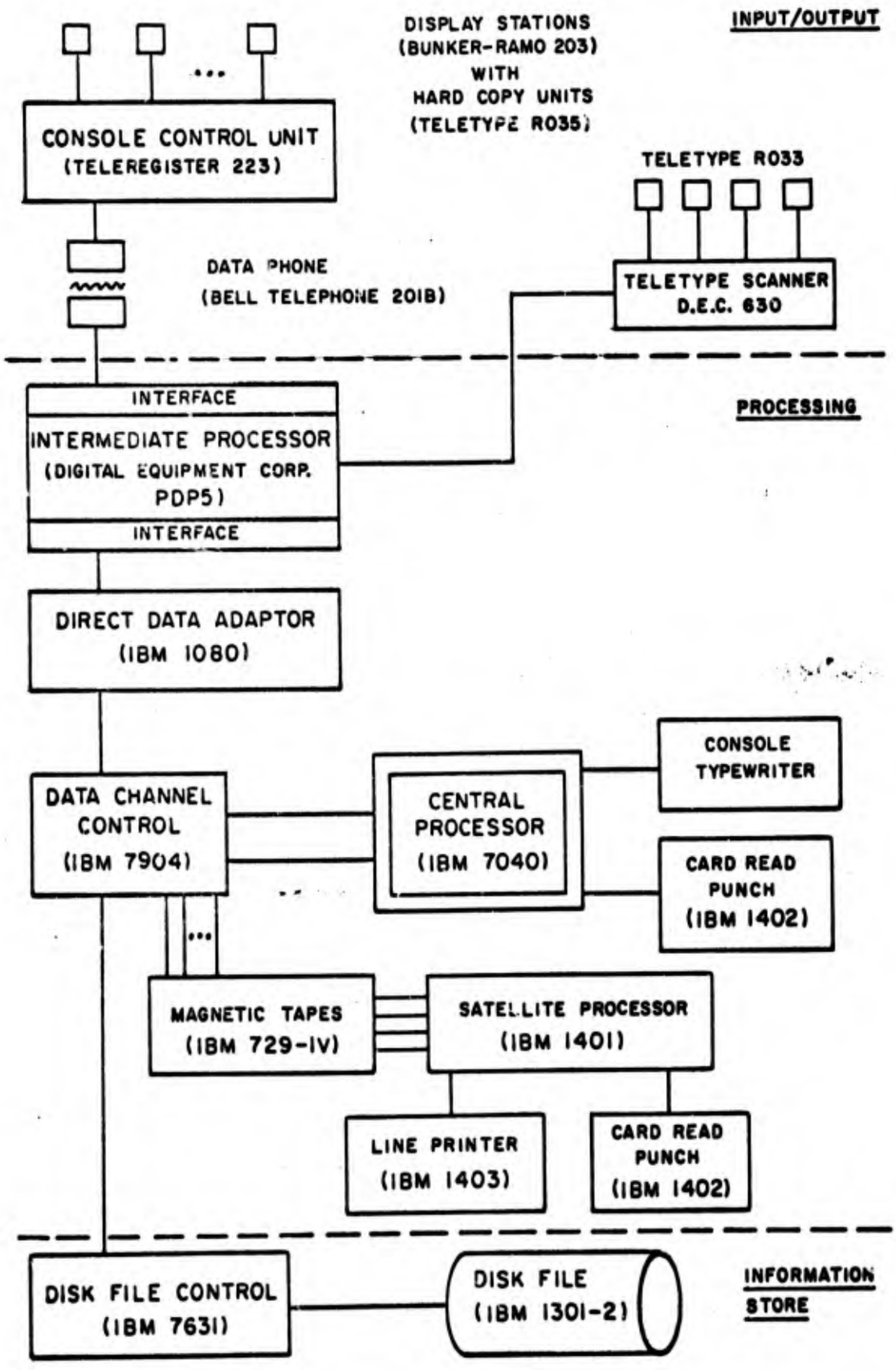


FIGURE 2.2 THE 7040/1301-PDP5 EQUIPMENT CONFIGURATION.

an Interpreter for executing queries and programs written in the Executive language by a user; and a Storage and Retrieval System[6] (SRS) for controlling a simulated associative memory in a disk and core storage. The SRS is based upon the MULTILIST techniques for simulating an associative memory in an addressable memory.[10,16,17] This technique is discussed in more detail in Sec. 3.1 and Appendix A3. Its main advantages are the rapid, random access storage and retrieval of information to and from disk storage; a user of the system need not be concerned about storage and retrieval of data or programs.

2.2 Equipment and Configuration

The specific equipment configuration used to implement the pilot version of the system is illustrated in Fig. 2.2. Bunker-Ramo Model 203 consoles are linked through a console control unit via data-phone to a PDP-5 computer. The PDP-5, acting as a local processor, performs the functions of editing input and output, reformatting input, scheduling and controlling priority for the various users on the system at any time. The local computer communicates through an adaptor with the data channel control of an IBM 7040 which acts as central processor. An IBM 1301 disk file is linked with the 7040's data channel control.

The disk file serves as the main mass storage device for the system. The programs and data for every user, as well as for many of the system programs, are kept on the disk. Information is kept on the disk until specifically deleted by a user. Both the PDP-5 and the 1301 are attached to channel B of the 7040. Other available equipment include magnetic tapes (on channels B and C), a console typewriter (output only) and card/reader/punch connected directly to the 7040 on channel A; and an IBM 1401 computer with line printer and card reader/punch which can use tape on channels B and C in

common with the 7040. All of this equipment is available to the user through the system.

Although the main processing path is Console to PDP 5 to 7040 to Disk File, the 1401 is useful for large scale input and output. The system is also programmed to accept input from card reader either directly or through the 1401 via magnetic tape and is able to give output to the line printer through 1401. (Much of the program testing was done in this manner even though on-line use is not possible this way. For this reason, many of the examples in later sections use the symbols that were defined for card input. Where symbols are defined that differ between card and console, both forms are given.)

2.3 The System Programs

The main language of the system, MULTILANG, serves both as a control language and as a programming language, allowing use of the system on three levels.

- The inexperienced programmer may make use only of programs already included in the system, although he may add new data and problem structures as needed.
- The experienced programmer may take fuller advantage of programs and data stored in the system by combining existing programs and data into new entities and adding new programs in any of various languages and data at will.
- The system programmer may modify the system itself.

In all cases, users can request data and programs by using a logical expression which specifies the required type of data or program by describing the desired item. (The term "item" is used to describe a block of information consisting of data and/or program. All information is organized into items,

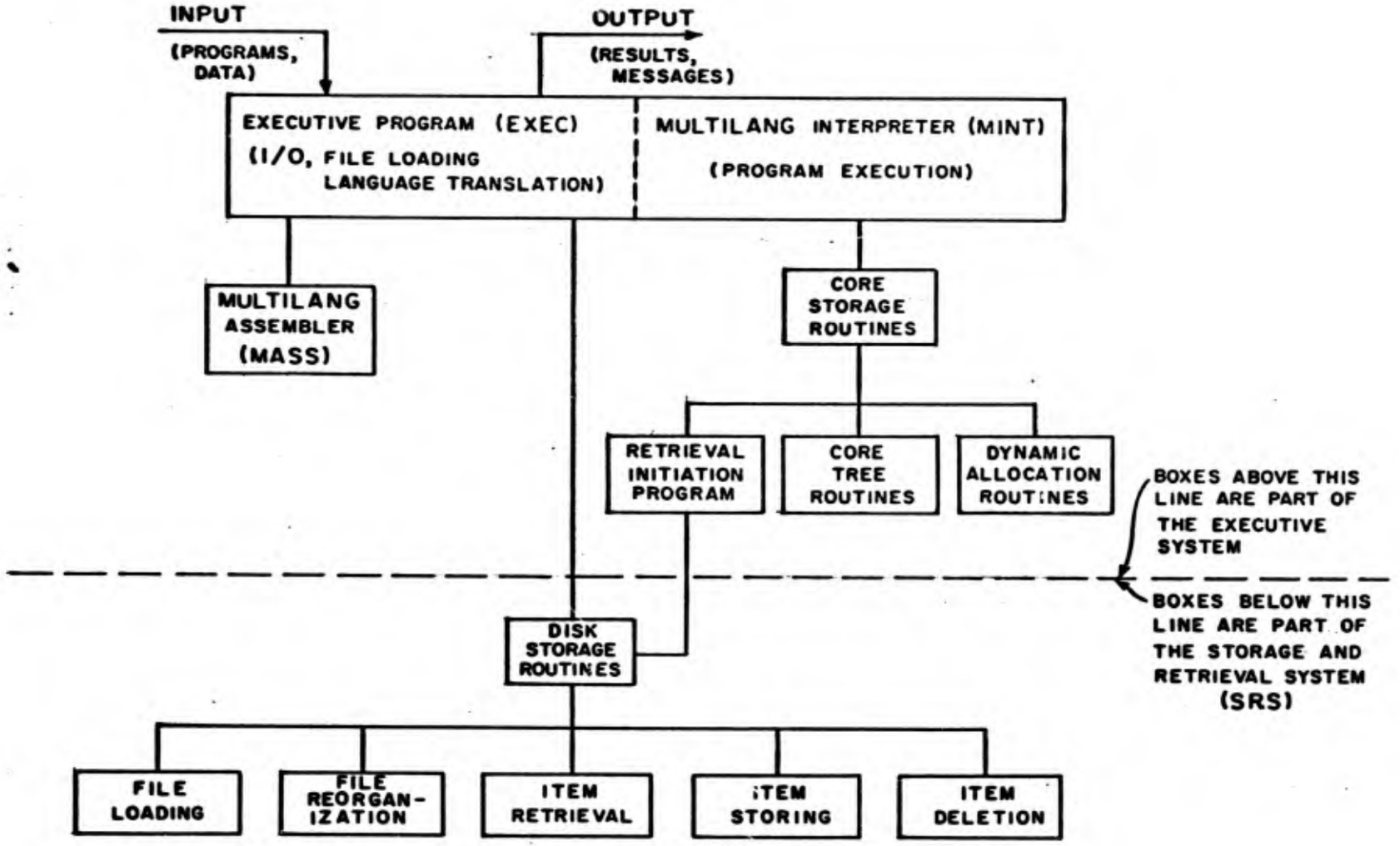


FIGURE 2.3 SYSTEM PROGRAMS

which can be decomposed into variable length records known as elements.) The description, a basic part of MULTILANG, is composed of (a) natural language, mnemonic or coded keywords describing the item; (b) conditions upon the values of data in the item; and (c) a listing of the specific parts of a multi-record item desired.

Since the system has been designed for mixed language use, user should be able to include programs in a number of problem-oriented languages, which will be added to the system. At present, users may define programs in MAP (the symbolic assembly language of the IBM 7040), and work is being done toward including ALGOL[2,4].

A block diagram of the System Programs is given as Fig. 2.3. The central block consists of two parts: an Executive Program which controls input and output, conversion of data from external to internal format, assembly of programs, and file/loading reorganization; and a MULTILANG Interpreter which controls execution of MULTILANG programs and directs retrieval of programs and data. The flow of control is from top to bottom, programs at one level controlling programs at lower levels. The Executive and Interpreter are fully discussed in separate appendices.

Briefly, the programming language is made up of lines of coding called "statements" which may be combined in groups into programs called "procedures." The input to the system consists of a series of messages which are programs in MULTILANG or other languages or data.

Below the Executive Program, at the left of Fig. 2.3 is shown the MULTILANG Assembler. This is an input routine used to convert MULTILANG programs from external to internal format and assemble them into packed blocks. In general, any assemblers, compilers or other special input conversion programs will be placed under the MULTILANG Assembler.

The programs simulating the associative memory on disk are in the lower part of Fig. 2.3. File loading and file reorganization, coming under the Disk Storage Routines, are considered to be Executive processes, usually occurring before, between and after the execution of programs. Retrieval, restoring, and deletion of specific items of information usually occur during the execution of a user's program. The Disk Storage Routines, control the sequencing and allocation of disk and call upon the five blocks as needed. These six blocks make up the SRS mentioned above. In addition to much of the disk, the associative memory also includes a large segment of core storage and a set of Core Storage Routines control the dynamic allocation of storage for program and data. The Retrieval Initiation Program is used by the interpreter to initiate retrieval (and restoring) from disk and provide communication between the interpreter and the disk memory.

2.4 Use of the System

MULTILANG is primarily an Executive Language, not a programming language. As will be seen in the next two sections, MULTILANG statements and procedures are used to specify calls on machine language programs stored in the file and to provide to these programs linkages to data or other necessary information. The formal specification of the syntax of a statement does not define the meaning actual executable statements in the language; rather, a structure or schematic form of a valid statement is given. This is analogous to the syntax definition of for example "< arithmetic expression >" in ALGOL-60[2] in which no actual arithmetic expressions are specified but a structure for a valid arithmetic expression is given.

A statement which purports to specify program and data is meaningless unless the program and data have been entered into the file. Initially, the system file is empty. As the system is used, programs and data will

be added to provide various capabilities. A user (or group of users) may enter any programs desired, either sophisticated or primitive. In the former case, each program might be sufficient to provide the solution of a program. In the latter case, statements calling upon the primitive programs may be combined into procedures, and the procedures may be stored upon the disk for future reference.

Appendix A7 describes a method of writing programs to enter into the system. The parts of a statement are interpreted partly by the MULTILANG Interpreter and partly by the worker program itself as explained in Sec. 4 and Appendix A7.

2.5 Examples of System Usage

The function of the various system program blocks shown in Fig. 2.3 can be further described by illustrating their function in specific system usage examples.

Figure 2.4 shows the general flow of man-machine functions.

The user may compose messages at remote consoles composed of:

1. MULTILANG statements.
2. MULTILANG procedures, composed of a number of MULTILANG statements and possibly followed by other information elements or items.
3. Data input to programs.

These messages are interpersed with Executive commands which identify the type of message and the desired system functions. The Executive commands are listed in Table 2.I.

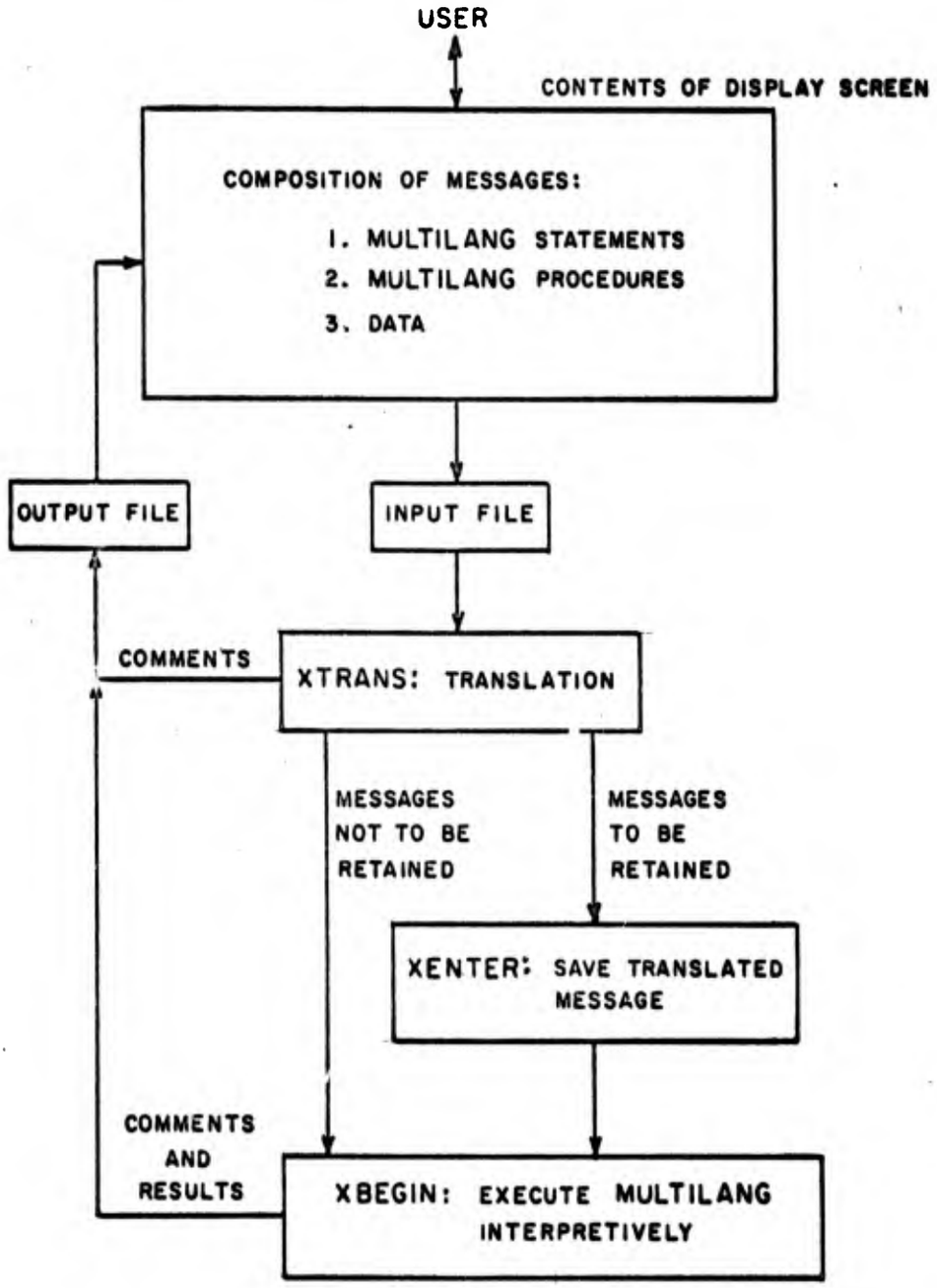


FIGURE 2.4 SYSTEM FUNCTIONS WHICH ARE CALLED FOR BY COMMANDS IN TABLE 2.1.

The function of the system in response to such input may consist of one or any combination of the following:

1. Translation into machine-item formats.
2. Compiling or assembling with communication of comments back to console.
3. Entering into the associative mass memory. This is if message is to be retained, even if only temporarily, for referral or modification.
4. Execution and communication of intermediate and final results to console.

Table 2.I Brief explanation of Symbols Used in the Text

(Full explanation is given in following Appendix A5)

Symbol	Meaning
XTRANS	An input translation operation to follow. XTRANS is always followed by either PROC or STAT.
PROC	A MULTILANG procedure follows.
STAT	A MULTILANG Statement.
END	The end of a MULTILANG input.
XENTER	Enter the previously input item into the simulated associative memory.
XBEGIN	Transfer control to the MULTILANG Interpreter to execute a procedure or statement.

As indicated in Fig. 2.4, single MULTILANG statements (indicated by XTRAN STAT at the beginning of message) can only be executed and cannot be entered into permanent storage. These are used to call for the execution of programs and MULTILANG procedures previously entered. MULTILANG procedures (identified by XTRAN PROC followed by name of procedure at the beginning of message) may be either entered or executed or entered and executed as specified by the input. XBEGIN following a statement or procedure indicates that immediate execution of the preceding is to occur. XBEGIN followed by a procedure name indicates that

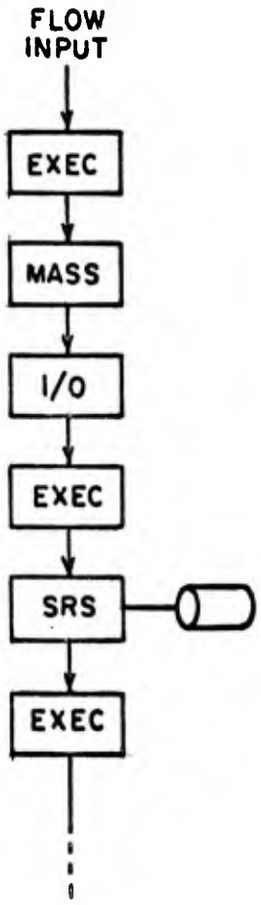
INPUT

XTRANS PROC "NAME"

: } MULTILANG PROCEDURE
: } (MAY HAVE DATA)

END

XENTER



KEY: EXEC : EXECUTIVE PROGRAM
MASS: MULTILANG ASSEMBLER
I/O : INPUT AND OUTPUT
SRS : STORAGE AND RETRIEVAL SYSTEM

FIGURE 2.5 FLOW DIAGRAM SHOWING THE SEQUENCE OF PROGRAMS DURING ENTERING OF A MULTILANG PROCEDURE.

the procedure named, which must have been previously input, is to be executed.

This illustrates the idea that MULTILANG procedures and statements are the means of communicating execution requests. Programming, however, will Eventually be able to be done in a number of existing computer programming languages, followed by compilation and/or assembly. Execution or test may follow immediately or be called upon at a later time by MULTILANG statements. Degugging and changes are communicated by MULTILANG statements calling by name previously entered programs.

The entry or revision of data elements or items requires translation from a variety of formats of incoming information and a variety of updating processes may be required. Entry of such data is called for by a MULTILANG procedure, which can use previously stored programs and tables to perform decomposition of input formation, translation and assembly to update the associative memory. The incoming data may be included in the message or may appear on another input device such as cards, magnetic or paper tape, etc.

Figure 2.5 illustrates the sequence of steps in entering a MULTILANG procedure. The message starts with the input symbol XTRANS (translate) which is read first and causes the Executive to transfer control to the MULTILANG Assembler (MASS). From then on the Assembler controls input processing up to an END symbol. The input symbol PROC (procedure), which appears in the same line as XTRANS, informs the MULTILANG assembler that a MULTILANG procedure follows. The procedure is then read by the assembler, line by line, and translated into internal machine format up to the symbol END which denotes the end of the procedure. Errors found in the translation are communicated to the console and control returns to the Executive. Data may be included in the procedure before the END or be entered separately as in the example of Fig. 2.7.

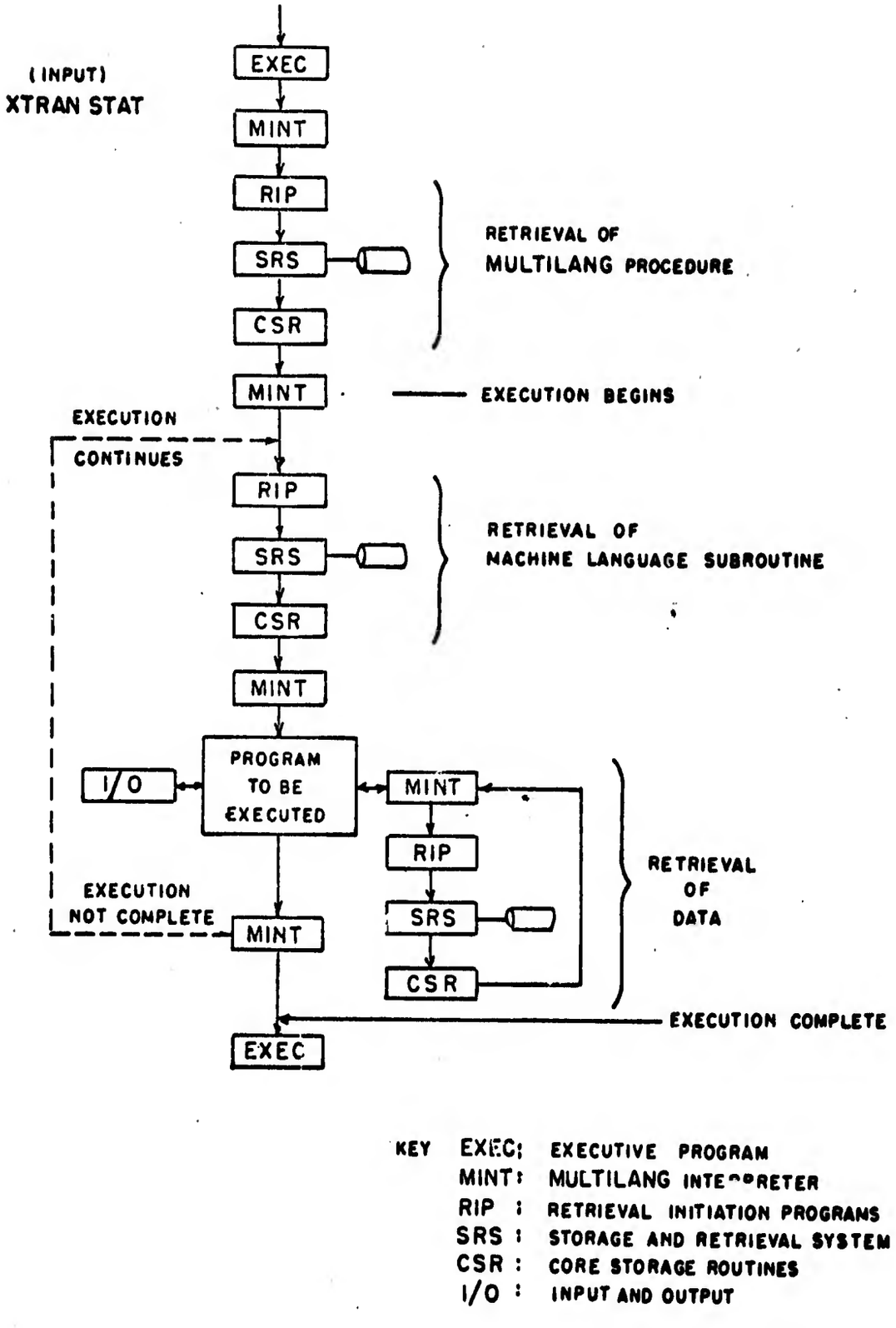


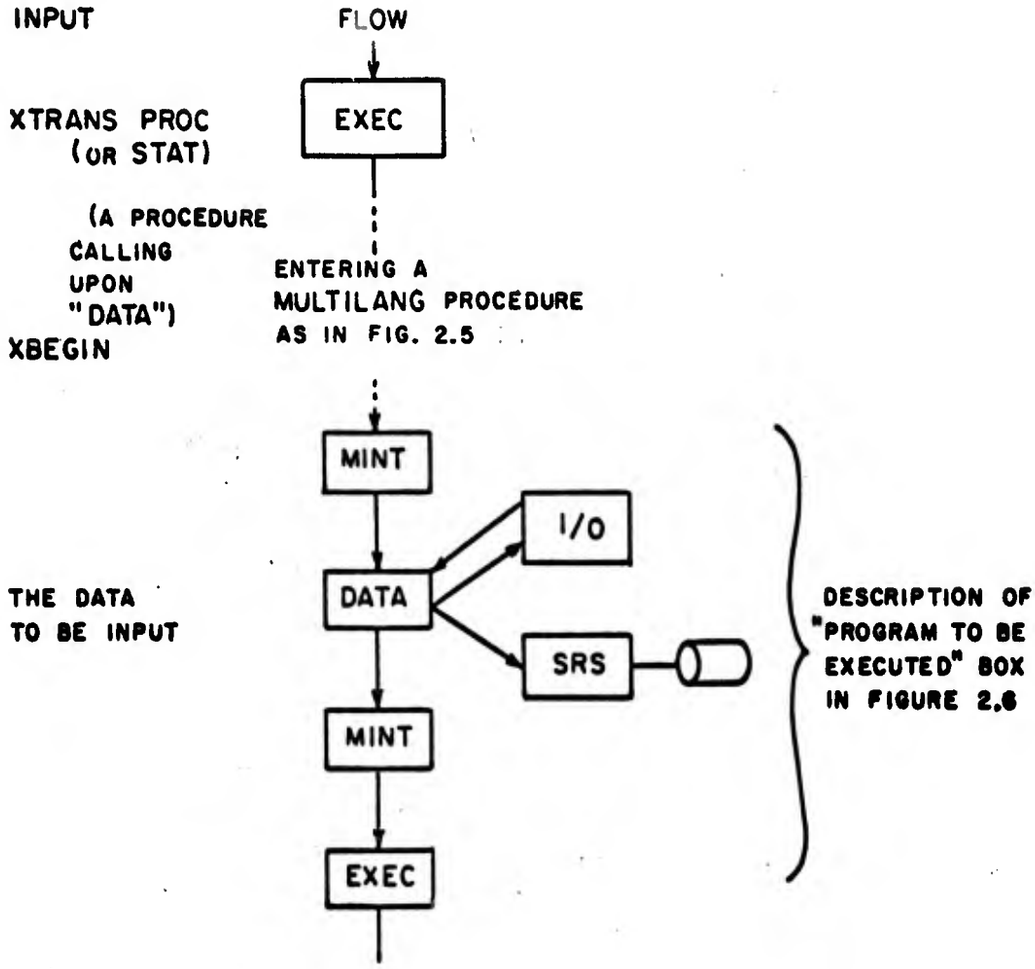
FIGURE 2.6 FLOW DIAGRAM ILLUSTRATING A SINGLE MULTILANG STATEMENT WHEN IT CALLS FOR EXECUTION OF A MULTILANG PROCEDURE.

In Fig. 2.5 the next input is the symbol XENTER which instructs the Executive the enter the procedure item into the associative memory. The Executive then calls upon the SRS which stores the item on the disk, returning control to the Executive. (In this case, the item will remain in the system until changed or deleted by the user who defined it.) The use of XENTER is optional, a user may define and use a procedure without entering it into the associative memory on disk.

Figure 2.6 illustrates the sequence of programs in response to an input message which contains a single MULTILANG statement. In this example the statement requests the execution of a MULTILANG procedure. A MULTILANG statement may include description of operand programs to be retrieved and executed as well as data to be retrieved. A MULTILANG procedure may be composed of one or several statements, data, parameters, variables and names. The message begins with XTRAN STAT followed by the statement. An XBEGIN symbol which follows instructs the Executive to transfer to the MULTILANG Interpreter. A retrival request is then transmitted to the SRS, in order to retrieve a MULTILANG procedure or worker program to be executed.

The diagram in Fig. 2.6 shows one sequence of three boxes several times. This is Retrieval Initiation Program to Storage and Retrieval System and back to Core Storage Routines and is the sequence used for retrieval of information from disk to core storage.

First the MULTILANG procedure of one or more statements is retrieved from disk to core and the Interpreter begins executing the procedure. Upon interpretation, each statement of the MULTILANG procedure specifies machine language routines. The Interpreter then transmits a request to the SRS to retrieve each routine from disk to core for execution. These programs may request retrievals of data via the SRS and may also call upon the Input and



KEY - EXEC: EXECUTIVE PROGRAM
MINT: MULTILANG INTERPRETER
DATA: DATA INPUT WORKER PROGRAM
I/O : INPUT AND OUTPUT
SRS : STORAGE AND RETRIEVAL SYSTEM

FIGURE 2.7 FLOW DIAGRAM SHOWING THE SEQUENCE OF PROGRAMS DURING THE ENTERING OF A DATA ITEM IN A MULTILANG STATEMENT OR PROCEDURE.

Output programs. As each program is completed, the interpreter seeks another program to execute, going from statement to statement in sequence or jumping, as instructed by the statements. When the last program is done, control returns to the Exective Program.

The problems involved in entering data items into the system are different from those of program entry. The internal format of a MULTILANG procedure item and the internal format of a worker program are specially designed for manipulation by the system programs. Data may originate and be assembled into items in almost any fashion convenient to a user (or to the programs he is using). For this reason, data input programs are to be entered as worker programs that are called upon by MULTILANG statements in a procedure. For the purpose of the example such an input program, called "DATA," is assumed.

As illustrated in Fig. 2.7, the user precedes the data by a MULTILANG procedure. A statement in the procedure calls upon the DATA routine, specifying all parameters needed (such as decimal data, octal data, the lines following the procedures which contain the data or the device the data is to be entered from: console, cards, etc.). If the procedure calling upon DATA is already in the file, this step may be omitted, and the procedure may be called by a statement as in Fig. 2.6. The execution of this procedure is initiated by an XBEGIN and the steps during execution are as given in Fig. 2.6.

During execution of the program "DATA" will call upon an input routine and upon the SRS as needed. The DATA program accepts input, assembles this information into an item and the SRS stores this item on the disk, as indicated in Fig. 2.7. The DATA program could input several sets of data during one execution; or it could be executed several times for several sets of data.

3. RETRIEVAL AND STORAGE: INFORMATION STRUCTURE AND DESCRIPTIONS

3.1 Information Structure

The basic unit of information in the system is the "item" - a block of characters that may contain program, data, or both. Items are stored in the simulated associative memory, on disk, and may be retrieved from disk to core by the Storage and Retrieval System for use either as programs or as data. Generally, the interpretation of a description initiates retrieval of programs to be executed. These programs may then request the interpretation of other descriptions and retrieval of items to be used as operands.

The general structure of an item is shown in Fig. 8 to consist of a Header, a set of key elements, a Linkword, a set of data elements, and a Table of Contents. The exact structure used and interpretation of these five parts are given in Appendix A2.

Figure 3.2 illustrates a specific item, corresponding to a "library catalog item." The information in this example is from an entry in an annotated bibliography on Artificial Intelligence.[16] Table 3.I shows the original entry and its associated subject code descriptors.

In brief, the Header contains code words for the use of the retrieval programs (Fig. 9, words 0 and 1). The second part of the item contains a set of keys (keywords, descriptors, attributes or names containing up to five characters) which classify an item, representing associations, characteristics or identifications of the information within the item (Fig. 3.2, words 2 through 31). The keys may be grouped into elements if the specific applications require it. In the example of Fig. 3.2, the key elements correspond to Date of Publication (Element 1, words 4 and 5), Subject Code Descriptors (Element 4, words 6 to 27) and Author(s) (Element 5, words 28 to 31). A Linkword containing information for the system follows the keys (Fig. 3.2, word 32).

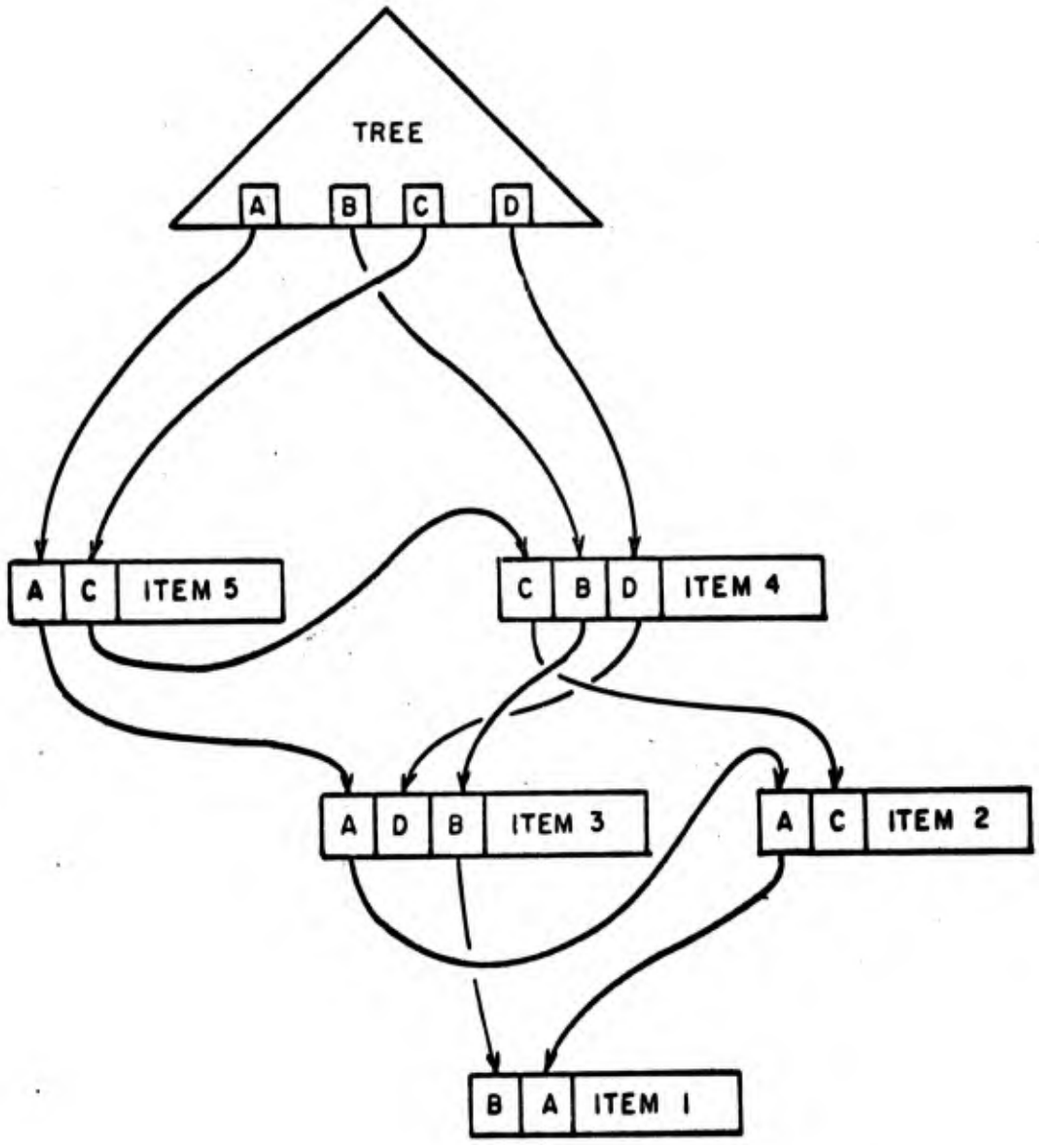


FIGURE 3.3 SAMPLE MULTILIST FILE

Table 3.I The Bibliography Entry Used in the Example of Fig. 3.2

Newell, A., and Simon, H. A., "The Logic Theory Machine," IRE Trans. on Information Theory, 1956 IT-2(3): 61-79.

This document is identified by the descriptors

J_1 - Discussion of Heuristics for Machine Solution of Problems

J_2 - Discussion of Human Problem-solving Heuristics

C_2 - Imperfect Searches Involving Failure, as Opposed to Decision Procedures

G_7 - Theorem Proving by Machine

G_8 - Use of Deductive Logic in Problem-Solving

I_2 - Grammatical Induction

H_4 - Symbol Manipulation Programming Systems

H_2 - Formal Languages

14 - Uses Special Programming System

34 - Report of Experiment

36 - General Discussion

Certain data elements are said to be "associated with" certain keys of the item: The element with number i + 64 is associated with the i^{th} key of the item. Various types of applications could use this method of associating data elements with keys. For instance, the 10^{th} key (14 - Uses a special programming language) has associated with it a name of a programming language that used - IPL. Another purpose is to effectively have variable length keys.

Data elements 79 and 8C contain the full names of the authors; these names were limited to five characters when used as keys.

Element 128 contains the full title, publisher, etc. The final part of an item is a table of contents which lists the element numbers and the

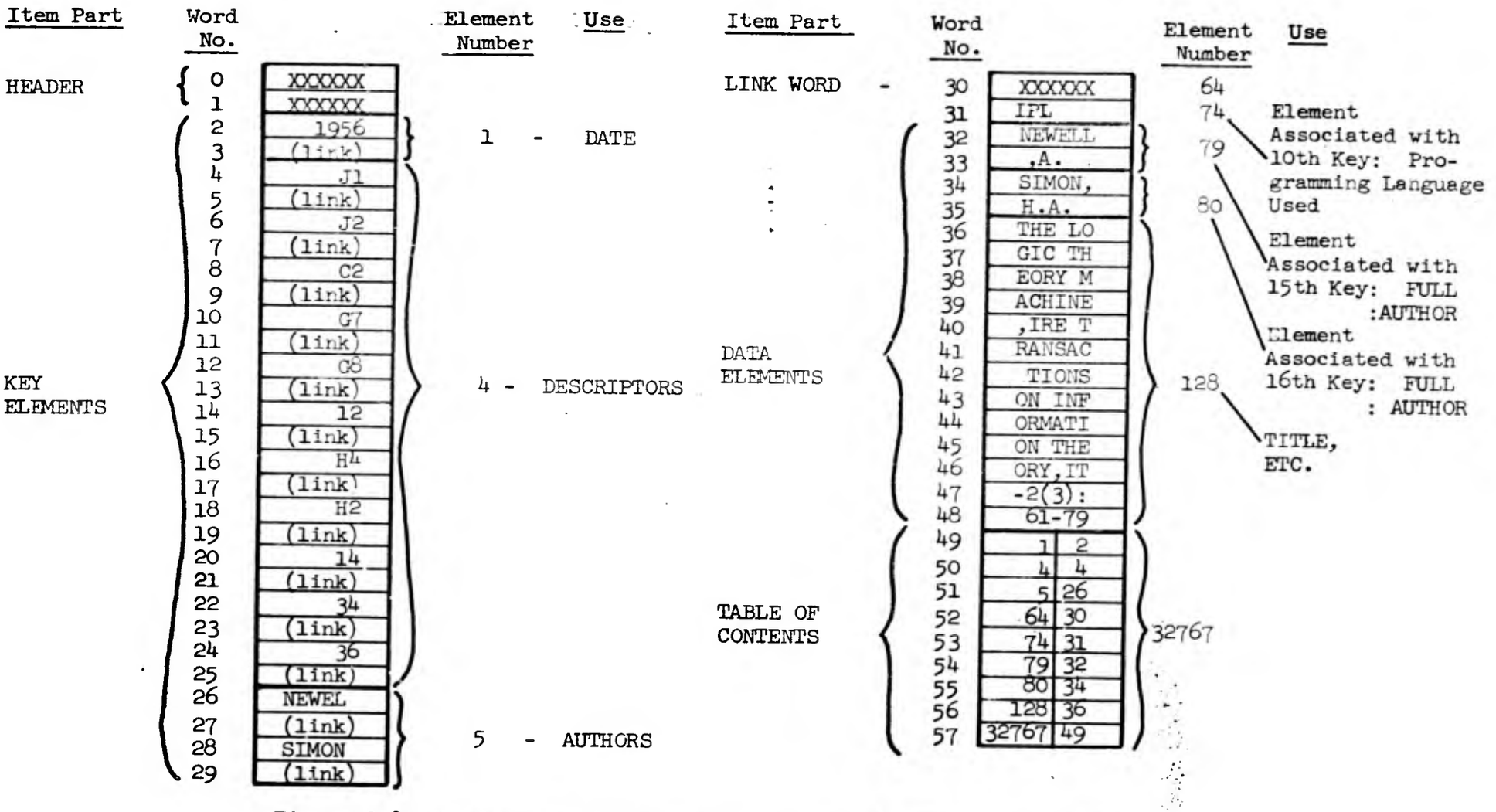


Figure 3.2 Sample Item Corresponding to the Information in Table 3.I

corresponding locations.

For purposes of the example in Fig. 3.2, the Linkword was assigned element number 64 and the table of contents was assigned the number 32767 (which, in octal, is 77777 - the largest possible element number). Also for purposes of the example, element numbers 1 through 63 were reserved for key elements, and element numbers 65 through 32766 were reserved for data elements.

3.2 Descriptions

An item may be recognized by the keys contained within it, the elements of information within it and the numeric value of the data within the elements of the item. In order to retrieve items stored in the simulated associative memory, an expression combining keys, element identifying numbers and conditions upon the values of elements is presented to the retrieval programs. The Storage and Retrieval System and the Core Storage Routines then retrieve all the desired items, extract the requested information and place it in the high speed memory. This information may be program and/or data.

This expression, made up of keys and element numbers, is known as a "description" and a formal Backus Normal Form Specification of "description" is given in Table 3.II.

A description is made up of three parts - the "primary" and "secondary" parts constitute a set of conditions which specify the items to be retrieved, and a list of numbers, called a "format," specifying the elements of a retrieved item that are to be kept in core memory. At least one condition must be present in every description but the format may be omitted. Fig. 3.3 illustrates the parts of a description.

Table 3.II - Backus Normal Form Specification of the Syntax of "Description"

Terms not defined here are explained in the text.

```
< description >:: = < primary part > < secondary part > < format part >

< primary part >:: = < key part > | (< key part list >)

< key part list >:: = < key part > 'OR' < key part > | < key part list >
                      'OR' < key part >

< key part >:: = < key > | < key > 'TO' < key > | < key value condition >

< key value condition >:: = < key > < relation > < value condition part >

< value condition part >:: = < key > | < element number > | < constant >

< secondary part >:: = < condition list > | < empty >

< condition list >:: = 'AND' < condition > | < condition list > 'AND'
                       < condition >

< condition >:: = < simple condition part > | (< simple condition part
                  list >)

< simple condition part list >::= < simple condition part > 'OR'
                                  < simple condition part > |
                                  < simple condition part list >
                                  'OR' < simple condition part >

< simple condition part >::= < existence condition > | 'NOT'
                             < existence condition > | < value condition >

< existence condition >::= < key > |< element number > |< key >
                           'TO' < key > | < element number > 'TO'
                           < element number >

< value condition >::= < key value condition > |< element number >
                       < relation > < value condition part >

< relation >::= 'LS' | 'LQ' | 'EQ' | 'GQ' | 'GR' | 'NQ'

< format part >::= < empty > |,F (< element number list >)

< element number list >::= < element number > | < element number list >
                           < element number >

< empty >::=
```

Table 3.III Formats of Simple Conditions and Rules for Truth and Falsity of a Condition with Respect to an Item

Name of Condition	Format of Condition	TRUE, when item under consideration	FALSE, when item under consideration
Existence Conditions			
1. Key	a) K	contains key K	does not contain key K
	b) 'NOT'K	does not contain key K	contains key K
2. Element	a) E(N)	contains an element with number N	contains no element with number N
	b) 'NOT'E(N)	contains no element with number N	contains an element with number N
3. Key Range (see note 1 below for def. of I)	a) K1'TO'K2	contains at least one key K such that $I(K1) \leq I(K) \leq I(K2)$	contains no key K such that $I(K1) \leq I(K) \leq I(K2)$
	b) 'NOT'K1 'TO'K2	contains no key K such that $I(K1) \leq I(K) \leq I(K2)$	contains at least one key K such that $I(K1) \leq I(K) \leq I(K2)$
4. Element Range	a) E(N1)'TO' E(N2)	contains at least one element with number N such that $N1 \leq N \leq N2$	contains no element with number N such that $N1 \leq N \leq N2$
	b) 'NOT'E(N1) 'TO'E(N2)	contains no element with number N such that $N1 \leq N \leq N2$	contains at least one element with number N such that $N1 \leq N \leq N2$

Table 3.III (Continued)

Name of Condition	Format of Condition	TRUE, when item under consideration	FALSE, when item under consideration
Value Conditions			
5. Key Value (see note 2 below for def. of m)	a) K'GR'C	contains key K as the j th key in the item; has an element E with number j+m; the first word of E is greater than the constant C	either: does not contain key K; or, if K is the j th key then no element is numbered j+m; or, if both K and E(j+m) are present, the first word of E(j+m) is less than the constant C
	b) K1'GR'K2	contains key K1 as the j th key in the item, key K2 as the i th key in the item; contains elements with numbers j+m and i+m; the first word of E(j+m) is greater than the first word of E(i+m)	either: does not contain both K1 and K2; or, if K1 is the j th key and K2 is the i th key, there are not elements numbered j+m and i+m; or, if both elements are present, the first word of E(j+m) is not greater than the first word of E(i+m)
6. Element value	a) E(N)'GR'C	contains an element with number N and the first word of this element is greater than the constant C	contains no element with number N; or, if E(N) is present, the first word of E(N) is not greater than The constant C
	b) E(N1)'GR' E(N2)	contains elements with numbers N1 and N2 and the first word of E(N1) is greater than the first word of E(N2)	either: does not contain an element with number N1 and an element with number N2; or, if both are present, the first word of E(N1) is not greater than the first word of E(N2)

Table 3.III (Continued)

Name of Condition	Format of Condition	TRUE, when item under consideration	FALSE, when item under consideration
7. Mixed Value	a) K'GR'E(N)	contains key K as the j<u>th</u> key in the item and contains an element with number N and an element with number j+m; the first word of element E(j+m) is greater than the first word of element E(N)	either: does not contain key K; or, if key K is present as the j<u>th</u> key, does not contain both an element numbered j+m and an element numbered N; or, if both are present, the first word of E(j+m) is not greater than the first word of E(N)
	b) E(N)'GR'K	(same as in 7a, mutatis mutandis)	

Note 1: I(K) means the value of the key when the five 6-bit characters are taken as a 30-bit integer.

Note 2: <u>m</u> is used to represent an arbitrary integer which is constant for any item but may vary among items. In the example of Fig. 3.2, m=64. In general, m is equal to the element number of an item's link word.

Note 3: The symbol " 'GR' " meaning "greater than" is used in this table. The other permitted relation symbols are given in Table 3.IV.

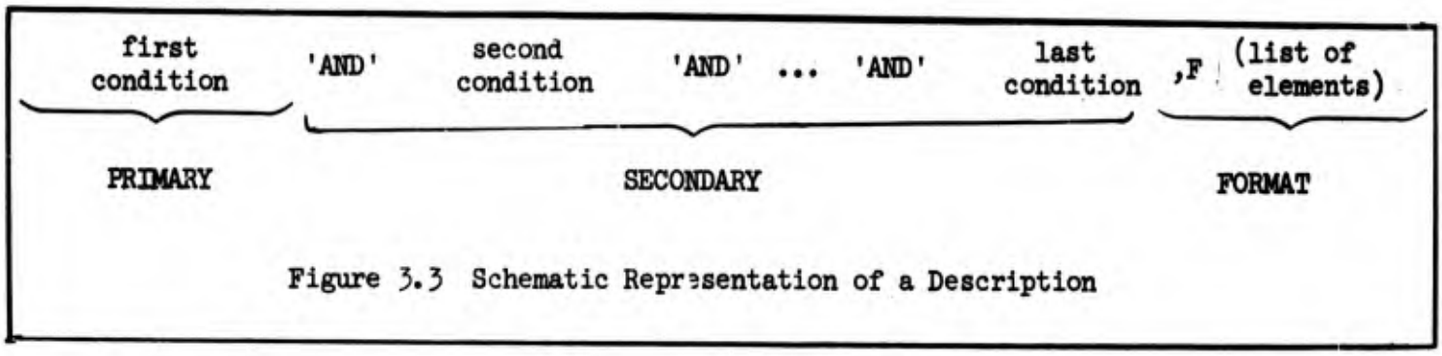


Figure 3.3 Schematic Representation of a Description

Each condition may be an expression containing the logical connectives 'OR' and 'NOT' so that the condition part forms a formula in Conjunctive Normal Form in which the "terms" are keys, element numbers or certain compounds containing keys and element numbers. The terms of this formula are known as "simple conditions" and the structure of the various legitimate simple conditions is given in Table 3.III. Every simple condition is either TRUE or FALSE with respect to a given item, and rules for determining truth or falsity of a simple condition may be found in Table 3.III.

To effect retrieval, a description is presented to the retrieval programs. The items retrieved will be those for which the formula in the condition part of the description is TRUE.

Conditions of the types numbered 1 through 4 in Table 3.III are known as "existence conditions"; that is, their truth or falsity depends upon the existence of an element or key in a given item. Conditions of types 5, 6 or 7 are known as "value conditions" since their validity depends upon the value of the data in an element of a given item. The value conditions are illustrated using the symbol 'GR', standing for the "greater than" relation, as an example. Table 3.IV gives the other acceptable relations and their symbolic representation

Table 3.IV Relation Symbols

Relation	Card Input Symbol	Console Symbol
greater than	'GR'	>
less than	'LS'	<
greater than or equal to	'GQ'	>= (representing $\geq$)
less than or equal to	'LQ'	<= (representing $\leq$)
equal to	'EQ'	=
not equal to	'NQ'	=/ (representing $\neq$)
and	'AND'	$\wedge$
or	'OR'	$\vee$
to	'TO'	$\rightarrow$ (representing arrow)
not	'NOT'	- (representing $\neg$ or $\sim$)

The first condition in the description, known as the primary part of the description is restricted to simple conditions of the types 1a, 3a, 6 and 7a of Table 3.III (i.e., non-negated key conditions).

The system is being programmed to accept input from card reader as well as from consoles, and preliminary testing is being done using card input. This report uses symbols in descriptions that were defined for use on punched card input. The console input symbols are more concise but less meaningful to the reader unfamiliar with the symbolic logic.

3.3 Interpretation of a Description

In order to discuss the method used to translate a description into a retrieval order, it is necessary to give some background information concerning the MULTILIST technique for information storage and retrieval. The MULTILIST

file organization uses list techniques to simulate an associative memory in which each item is stored with an arbitrary number of names (keys) and any item may be retrieved by any of its names. There is one list for each key in the file and all terms containing a given key are on a unique list associated with that key. A mapping function upon an index tableknown as a "tree" provides a unique translation from a key to the address of the head of that key's list. An item appears only once in the file but is linked to as many lists as there are keys in the item. Figure 3.4 illustrates the linkage for a set of fictitious items containing keys from the set [A,B,C,D]. The items are numbered in order of their supposed entry into the file.

The list of items containing key A is 5,3,2,1; while those with key C are 5,4 and 2. Since retrievals follow along lists, in order to retrieve those items with both A and C, corresponding to the description

A'AND'C

it is necessary either to follow the A list and then look for the presence of key C, or else to follow the C list and seek the presence of key A. In this case, the latter would appear preferable since the C list is the shorter. The details of the actual retrieval mechanisms, as well as the storage of items and exact file organization, are discussed at length in the SRS report previously mentioned.[8]

In Fig. 3.3 the form of a description was illustrated and shown to have three parts. Each part has a particular role in the retrieval of items from disk to core:

The primary part determines the key lists which control the retrieval.

The secondary part specifies which items having keys in the primary part are to be selected.

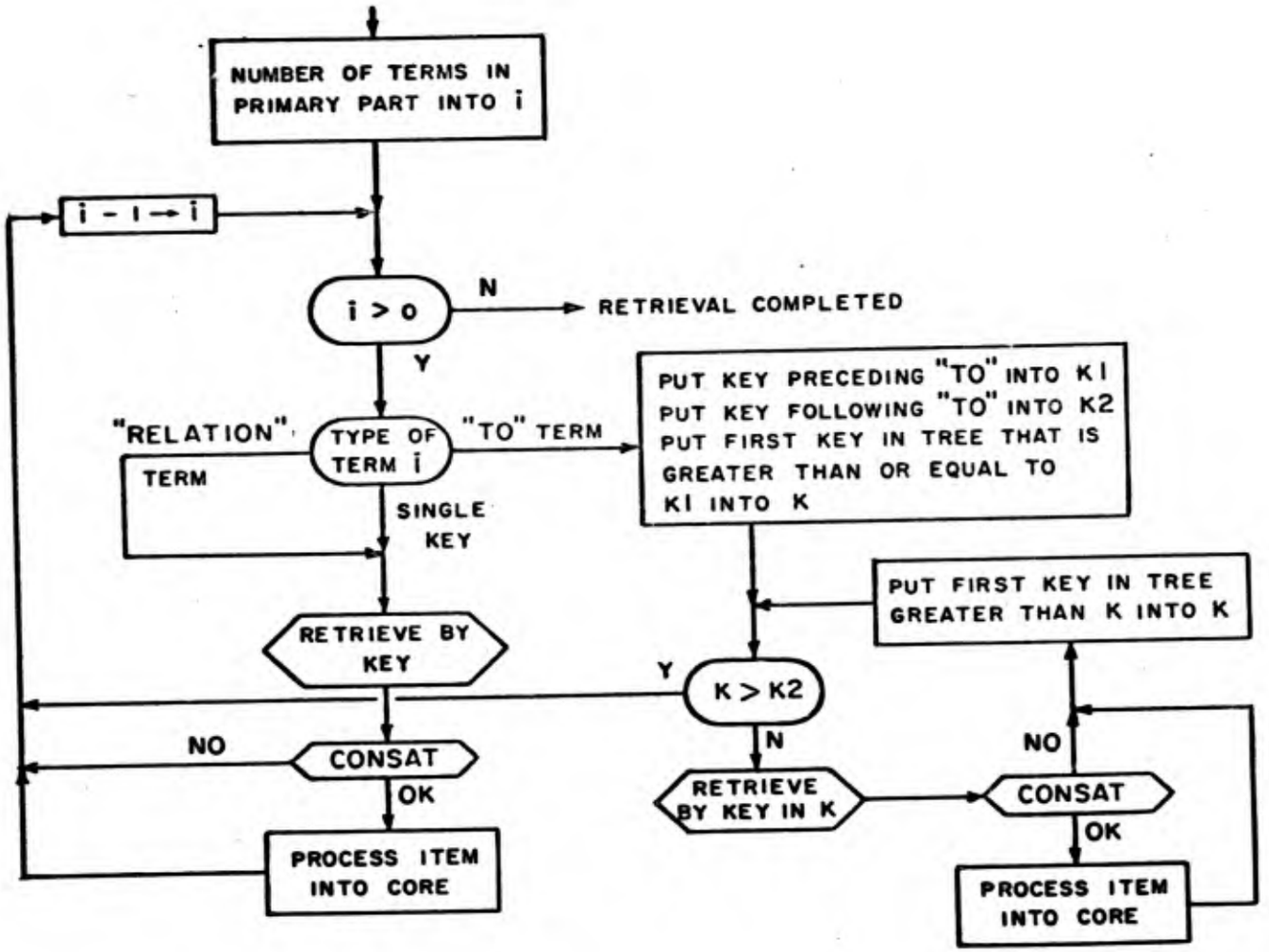


FIGURE 3.4 FLOW CHART OF PRIMARY PART DECODING AND RETRIEVAL.

The format part identifies the specific elements desired by the user. These elements are extracted from the selected items and are stored in high speed core memory.

The primary part consists of one or more "terms" (separated by 'OR' when more than one) where each term is of the form: KEY, or KEY1'TO'KEY2, or KEY'RELATION'SOMETHING. Fig. 3.5 contains a flow chart illustrating the method of retrieval in a somewhat idealized form. For a key alone or appearing first in a relation term, the retrieval is along that key's list. For a KEY1'TO'KEY2 term, the retrieval is along the list corresponding to every key present in the system whose integer image is in the range [I(Key 1), I(Key 2)].

Once an item has been retrieved it is necessary to compare the keys and elements in this item to the conditions specified in the description. A subroutine known as the "conditions satisfied?" (CONSAT) subroutine takes an item and a description and determines whether the item satisfies the description or not according to the definitions of Table 3.III.

The retrieval programs provide a requesting program with all the items satisfying a given description one at a time. This may in some cases by many items, in some cases only one item, and in other cases no items at all.

The worker program which is performing the retrieval decides what to do in case of too many or too few items.

The material in this section is described in full detail in Appendix A3, where a complete expansion of the flow chart is given.

3.4 Examples of the Use of Descriptions

The use of descriptions for retrieval may be illustrated with sample descriptions referring to the bibliography entry used in Fig. 3.2. Table 3.V restates the elements used in the example.

Table 3.V Meanings of Elements of Fig. 3.2

Element	Contents	Key
1	Date (year of publication)	Yes
4	Subject Codes	Yes
5	Authors' names (abbr. to 5 char.)	Yes
79,80	Authors' Full names	No
128	Full title, publisher, etc.	No

For the first description of Table 3.VI, every item in the file for which "SIMON" is a key would be retrieved; in this case, every document with Simon as an author, including the document of Fig. 3.2.

The second description is somewhat more restrictive and will retrieve all documents written by Simon, published in 1956 and identified by key G7 (Theorem Proving by Machine). The format specifies that only Element 128 (title, publisher, etc.) is to be kept.

Table 3.VI Some Sample Descriptions

```
SIMON                                                          (1)
SIMON'AND'1956'AND'G7,F(128)                                   (2)
NEWEL'AND'SIMON'AND''NOT'SHAW                                  (3)
(NEWEL'OR'SIMON)'AND'H2'AND''NOT'H3,F(4,5,1,128)               (4)
32'AND'(P4'OR'I)'AND'A1'TO'A6                                  (5)
1956'AND'E(5)'EQ'=HOSIMON                                      (6)
```

(=H Prefixes a 6 character constant)

The third description asks for documents by both Newell and Simon but not also by Shaw.

The fourth description specifies documents by Newell and/or Simon, containing key H2 (Formal Languages) but not Key H3 (Programming Language Systems). (To exclude jointly written documents 'AND'('NOT'NEWEL'OR''NOT'SIMON) could be added anywhere after the first term and before the format.) The format states that elements 4, 5, 1 and 128 are to be kept.

The fifth description concerns purely subject codes, identifying documents that contain

32 (Extensive Bibliographies)

and either P4 (Self-Reproducing Machines)

or I (Inductive Inference Machines)

and also one or more of

A1 (Finite State Machines, Mathematical Theory)

A2 (Logical Network Theory)

A3 (Switching Theory)

A4 (Turing Machines)

A5 (Growing Machines)

A6 (Probabilistic Machines)

The preceding examples have accepted authors whether first or not. It is, however, possible to take advantage of the element value conditions (which consider only the first word of a multiword item) to seek first authors. The sixth description of Table 3.VI asks for all documents published in 1956 with (the first word of) element 5 equal to the constant OSIMON. Since element 5 is the author element, this is equivalent to specifying that the first author key be SIMON.

4. THE MULTILIST EXECUTIVE LANGUAGE--MULTILANG, STATEMENTS AND PROCEDURES

4.1 Introduction

MULTILANG consists primarily of descriptions of programs and data associated with problem solution. Statements in the language are interpreted and corresponding programs are executed. "Description" has been defined and explained in Sec. 3. This section describes how to assemble these descriptions, together with other information, into MULTILANG Statements (Sec. 4.2) and Procedures (Sec. 4.3).

As illustrated by the examples in Sec. 2.3, messages composed at the consoles consist of:

1. MULTILANG Statements and Procedures which may be entered into mass storage and/or executed interpretively;
2. calls for Executive functions, as enumerated in Table 2.I (and explained in Sec. 2.3);
3. data

Though capability for describing complex structures is inherent in MULTILANG, the user may use a very simple subset of the available capability. For instance, the complexity of the description is entirely up to the user. Similarly, the system will accept single statements or sequences of statements in a procedure. Existing procedures stored in the system may be called by a single name. Natural language or mnemonic words may be used as keys in forming descriptions and as procedure names.

4.2 The MULTILANG Statement

A line of coding in MULTILANG is called a statement. A statement may appear alone in a message or with other statements in a MULTILANG procedure. Each statement corresponds roughly to a subroutine call with an arbitrary number of parameters. Upon interpretation, the Interpreter program initiates

the retrieval of the "described" routines and data.

A schematic representation of a statement is given in Fig. 4.1.

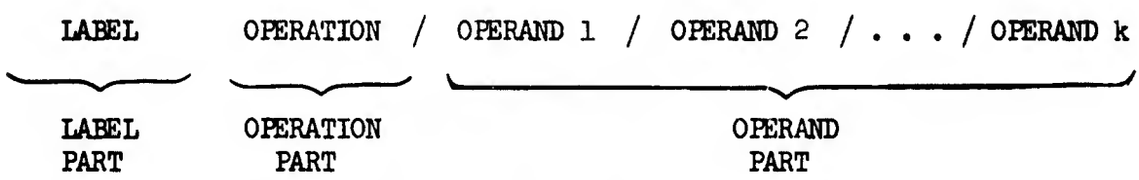


Figure 4.1 The Format of a Statement

A statement may begin with a label - an octal integer of one to five digits - or it may have a blank first field. When present, the label serves to name or identify the statement so that the interpreter may call on it as a result of execution of another statement.

The operation part of the statement, which must always be present, is a description interpreted as describing the program to be executed. The Interpreter initiates retrieval using this description. Any retrieved items are considered programs. A description used for retrieval may identify no, one, or several of the items. If the SRS finds no items in response to the description of the operation, the respective statement is treated as a "no-operation" and is omitted (the user being so notified). If the operation part description identifies one item and this item is a program item, the program will be retrieved and executed. If more than one program item is identified by the description, the programs are retrieved and executed one at a time in order of retrieval.

The function of the operand part is to describe or provide the operands, parameters, variables, etc., necessary for execution of the operation. Following the operation part of a statement are operands, each preceded by a solidus (/) and each may be any one of the following:

a) a description

b) a local name - defined in Sec. 4.3

c) a label or element number
d) a constant or block of constants } - defined in Table 4.II

A sample statement:

3776 TAN-1'AND'E(21)'GR'=.0005/=.3321

The above statement consists of a label (3776), an operation description and one operand (.3321). Upon interpretation of this statement, the Interpreter will retrieve and execute all programs having the keys TAN-1 and an element numbered 21 with the value of element 21 greater than 0.0005. The decimal constant .3321 is the sole argument.

The following statement illustrates other possible cominations:

12321 GAME'AND'E(17)'GR'=.07/(BOARD'OR'OLD)'AND'MOVE'EQ'=17
/B34/L(139)/E(3327)/=.0427,=HHELPME

The first part is the label: 12321

The second is the operation part, a description identifying all programs with the key GAME and element 17 greater than .07.

The third part is the first operand. It is another description where the keys are BOARD or OLD and where the key MOVE has an element associated with it containing the decimal number 17.

The fourth part, the secone operand is a single key which may be a description or a local name depending upon the context (as will be explained in Sec. 4.3).

Table 4.I Backus Normal Form Specification of Procedure
(continued in Table 4.II)

(1) < procedure >::=< proc part > < syn list > < locnam group >
< statement group > < endstat symbol > < data group >
< end symbol >

(2) < proc part >::=PROC < procedure name >
< procedure name >::=< key >

(3) < syn list >::=< empty > | SYN < synonym list >
< synonym list >::=< synonym > | < synonym list > , < synonym >
< synonym >::=< key >

(4) <locnam group >::=< empty > | < locnam list >
<locnam list >::=< local name spec > | < locnam list >
< local name spec >
<local name spec >::=LOCNAM < local name > / < description >
<local name >::=< key >

(5) < statement group >::=< labelled statement > < statement list >
< labelled statement >::=< label > < unlabelled statement >
< label >::=< element number >
< unlabelled statement >::=< operation > < operand list > |
< operation >
< operation >::< description > / < local name
< operand list >::= / < operand > | < operand list > / < operand >
< operand >::=< description > | < local name part > | F(< label >) |
L(< label >) | < constant list >
< local name part >::= < local name > < format >
< format >::= < empty > | , F (< element list >)
< constant list >::= < constant > | < constant list > , < constant >
< statement list >::= < statement > | < statement list > < statement >
< statement >::= < labelled statement > | < unlabelled statement >

(6) < endstat symbol >::= ENDSTAT

(7) < data group >::= < empty > | < constant group >
< constant group >::= < constant block > | < constant group >
< constant block >
< constant block >::= < label > : < constant list >

(8) < end symbol >::= END

Table 4.II Backus Normal Form Specification of Label and Constant

< constant >::= =< decimal constant >|=0< octal constant >|
= H< hollerith constant >

< decimal constant >::=< sign > < decimal string >

< decimal string >::=< digit string >|< digit string > <digit string >

< digit string >::=< digit >|< digit string > < digit >

< digit >::= 0|1|2|3|4|5|6|7|8|9

< sign >::= < empty > | + | -

< octal constant >::= < octal string >

< octal string >::= < octal digit >|< octal string > < octal digit >

< octal digit >::= 0|1|2|3|4|5|6|7

< hollerith constant >::= < character string >

< character string >::= < character >|< character string > < character >

< character >::= < digit >|< letter >|< special >

< letter >::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

< special >::= '|=|:|>|+|∧|.|)|[|<|-|∨|*|]|;|/|,|b

< label >::= < octal constant >

Note: A label is restricted to 5 characters
A decimal constant is limited to 16 characters
An octal constant is limited to 12 characters
A Hollerith constant is limited to 6 characters

The fifth part consists of a label.

The sixth part contains an element number.

The final part is a block of two constants.

The second is the operation part, a description identifying all programs with the key GAME and element 17 greater than .07.

The third part is the first operand. It is another description where the keys are BOARD or OLD and where the key MOVE has an element associated with it containing the decimal number 17.

The fourth part, the second operand is a single key which may be a description or a local name depending upon the context (as will be explained in Sec. 4.3).

The fifth part consists of a label.

The sixth part contain an element number.

The final part is a block of two constants.

4.3 The MULTILANG Procedure

Messages from consoles may contain a single statement - for ease in usage or a procedure, representing more complex instructions to the system. A procedure consists of one or more statements in MULTILANG together with certain identifying parts. The MULTILANG Assembler converts MULTILANG procedures from the console message format to a compact internal format, creating an item that may be stored in the simulated associative memory.

A MULTILANG procedure is nominally composed of seven parts, some of which may be omitted. The Backus Normal Form specification of the syntax of a procedure is given in Tables 4.I and 4.II.

1) PROC (Procedure Name) Part:

Following the Executive symbol XTRANS (translate) each procedure begins with the symbol "PROC" followed by the name of the procedure, a key made up

of one to five characters.

example: XTRANS PROC SAMPL

The key following PROC will be the first key of the item formed from the procedure.

2) SYN (Synonym) Part:

Additional keys are entered as alternate names for the procedure following the symbol "SYN" on the next line.

example: SYN STUFF,RLW,234,5XBC

Thus keys STUFF, RLW, 234 or 5XBC are other names of the procedures by which it may be retrieved. This line is optional and may be omitted if no additional names are desired.

3) LOCNAM (Local Name) Part:

Descriptions may be quite lengthy and it may be necessary for more than one statement in a procedure to use the same description. To avoid having to repeat the entire description, a "local name" may be defined to stand for and be used in a statement in place of the description. An arbitrary number of local names may be defined, each definition consisting of the symbol "LOCNAM" followed by the local name (a key) then the description. A solidus (/) separates name from description.

example: LOCNAM SEMI/QRP'AND'E(39)'GQ'E(97)

The Local Name "SEMI" is defined as equivalent to the description "QRP ...".

A set of characters defined as a local name is so interpreted whenever it stands alone as an operand. Should the same set of characters be used as a key in a description, the second use will be independent of the first and no ambiguity will arise. Keys defined in PROC or SYN fields may also be used in descriptions or as local names.

4) BODY Part:

The next section of a procedure, present in every procedure, is the body consisting of one or more statements. Normal execution of statements is first, second, third, etc. in succession, unless one of the subroutines executed specifies a "jump" to an out-of-sequence statement.

5) ENDSTA (End of Statements) Part:

Following the last executable statement is the symbol "ENDSTA."

6) DATA Part:

An arbitrary number of data elements may be assembled into a procedure item in the form of pseudo-statements - that is, sets of characters that look like statements but actually contain only data and are not executable. The format of a pseudo-statement is given in Fig. 4.2.

Label Constant 1, Constant 2, ..., Constant k

Figure 4.2 Format of a Data Generating Pseudo-Statement

Each pseudo-statement generates a single data element and must contain at least one constant. Pseudo-statements must be labelled.

7. END Part:

The end of a procedure is signalled by the symbol "END" and if there are no pseudo-statements the ENDSTA may be omitted.

The minimum number of parts in a procedure is three: PROC followed by a key, one or more statements and END. Figure 4.3 shows a simple procedure.

```
PROC SAMPL
 362 PROG
END
```

Figure 4.3 A Simple Procedure

This procedure called SAMPL has a single statement labelled 362 which is a call upon the routine PROG. (XTRANS would precede the PROC in an actual application).

A procedure may then be followed by executive symbols XENTER and/or XBEGIN as shown in Sec. 2.3.

Figure 4.4 illustrates a complete procedure.

The procedure of Fig. 4.4 is called GAME with RENJU and MOKU as alternate names. VECTR is a local name corresponding to the description TERM'AND'E(21)'GQ' = 123. The first statement is labelled "1" and calls for execution of all programs COMPV on the operand BOARD. There follow several other statements, some labelled; following the ENDSTA, is one pseudo-statement labelled "7" which defines a data element containing the constants 300,300,200,400,0,0, and 100.

Just as the example of Fig. 4.4 tacitly assumed the existence of a program in the file known as PROG, the example of Fig. 4.5 assumes the existence of "worker" programs named COMPV, MNMX, TEST, OUTMV, INMV, STORE, and QUIT. The procedure in Fig. 4.5 is meant roughly to indicate the sequence of execution of subroutines in a simple game-playing program.[25] Without going into the smantics of the statements the interpretation of a few will be considered.

In the first statement (labelled "1") the program (or programs) labelled COMU will be retrieved and executed. (This might be the program that computed a vector of possible moves,) Upon the request of the program, retrieval will

```
PROC    GAME
SYN     RENJU,MOKU
LOCNAM  VECTR/TERM'AND'E(21)'GQ'=123
 1      COMPV/BOARD
        MNMX/VECTR/BOARD/L(1)
 331    (TEST'OR'MATCH)/STORE'AND'(E(23)'EQ'=WHITE'OR'
        E(23)'EQ'=HOODRAW)/BOARD
        MOVE/L(1)/L(777)
        OUTMV
        INMV/L(1)/LEARN/BOARD/L(331)/E(7)
 777    STORE
        QUIT
ENDSTA
 7      =300,=300,=200,=400,=0,=0,=100
END
```

Figure 4.4 A Procedure

```
PROC      GAME
SYN       RENJU, MOKU
LOCNAM    VECTR/TERM'AND'E(21)'GQ'=123
 1        COMPV/BOARD
          MNMX/VECTR/BOARD/L(1)
 331      (TEST' OR'MATCH)/STORE'AND'(E(23)'EQ'=HWHITE'OR'
          E(23)'EQ'=HOODRAW)/BOARD
          MOVE/L(1)/L(777)
          OUTMV
          INMV/L(1)/LEARN/BOARD/L(331)/E(7)
 777      STORE
          QUIT
ENDSTA
 7        =300,=300,=200,=400,=0,=0,=100
END
```

Figure 4.5 A Complete Procedure

be performed according to the description "BOARD" (which might identify an item containing the board representation) and the address, in core, of an item corresponding to the description (if such there be) will be given to the COMPU program. The program does not actually "know" what the description is; requesting retrieval according to its first operand. The actual decoding of the description and its translation into retrieval requests is done by the system programs (cf. Appendix A5). In general, a worker program and the system programs do the rest. If this operand is a description (or local name) retrieval is done. If the operand is a label or element number, the label or element number is given to the program. Should the operand consist of constants, the address of the first constant is given to the worker program. This, also, is described in Appendix 3.

A worker program has the option of informing the interpreter what to do next: continue to the next statement, terminate the procedure or skip to another statement. The method for doing this is described in Appendix 6. The statement above the one labelled 777 (which begins with INMU) illustrates these points. The first and fourth operands are labels, the second and third are descriptions. The last operand is an element number. An INMU program may request "retrieval" of operands one, two,... or five. In each case, the result will be a code indentifying the type of operand and the resulting data according to the rules given above. (Requests for operands greater than five will yield an error return). The program, upon completion, and depending upon the results of its computation, may have the interpreter continue or go to statements 1 or 31. An attempt to execute any data element or non-existent statement (undefined label) will cause termination of the procedure. Needless to say, the programs and data specified must be in the file before the procedure can be executed.

5. CONCLUSIONS

5.1 Conclusions From System Implementation

This report has described the pilot version of a real-time man-machine computer system, the Moore School Problem Solving Facility.

As stated in Sect. 1.4 the basic hypothesis were that through accumulating knowledge the system would be able to cope with increasing classes of functions and that an executive language will make the system more compatible with user. The mechanization of this facility has achieved these goals:

1. An Information Storage and Retrieval System sufficiently versatile to relieve a user of the need to worry about where his data and programs are at any given time. Application of the MULTILIST indexing technique to a dynamic storage allocation environment provides a quick and efficient means of storing, retrieving, modifying, and deleting information.
2. An Executive Language which provides the ability to specify programs for execution and data for these programs either directly or indirectly by means of descriptions to be interpreted by the retrieval programs and also the programs necessary to interpret the language and translate it into execution and retrieval calls.

The system is able to "grow" in two senses, by adding, deleting, or modifying data or program and by defining new associations and relations between existing information as it is used.

The facility makes use of a central computer with a large memory, a smaller computer for input and output, and display consoles for man-machine communications. The system consists of (a) a storage and retrieval component capable of handling large amounts of data and programs, (b) an executive language for communicating requests for information and program execution, and (c) an executive

program that permits several users to communicate with the computer at the same time to provide real-time processing.

The system is open-ended, requiring a minimum amount of program permanently kept in the central computer. Almost all functions and operations are peformed by machine language "worker" programs stored in the simulated associative memory, retrieved when needed by a user. Any user may add worker programs to the system to perform special functions and any user may define procedures using sequences of statements calling on worker programs.

In Sect. 1.8, the capabilities that make a programming system general-purpose and time-sharing were quoted. How well does the Problem Solving Facility fit that description?

(1) Availability of components of the system to the user.

Any user may make use of any part of the system either directly through statements in the Executive Language or indirectly by calling upon programs that use the system.

(2) Programming in various languages.

Although not currently available, programming language processors such as assemblers, compilers and interpreters could be entered as worker programs, called just as any other program, thereby permitting programming from the console in languages other than the Executive Language.[1]

(3) Use of programs formulated outside of the system.

MAP programs can be entered into the system off-line. Very few modifications would be needed in any previously written MAP program.

---

[1] As of this writing, work is in progress to add MAP, the Symbolic Assembly Program of the IBM 7040 to the system.[2] Preliminary studies have begun toward adding an ALGOL compiler written at Grenoble University.[4]

(4) Real-time operation.

At present, consoles work on a first-in-first-out basis. When several users transmit information in sequence on a single console, the response is directly in the order of input. There is time-sharing among consoles.

(5) Ease of communication.

Use of display consoles for input and output permits quick transfer from user to machine. Versatile editing features allow for modification of information on the console screen[1]. Output may be edited and then re-entered as input.

(6) Convenient mass data storage and retrieval.

Information storage and retrieval is a fundamental component of the system, so planned that information can be added or deleted whenever required.

5.2 Future Applications and Extensions of the Problem Solving Facility

These capabilities will have a large impact on Military Command and Control, Data Files, strategic, tactical and management information systems providing for efficient integration of large operation, real time query and updating and machine assistance in decision making. The system described is an example of one of several nuclei which may be started simultaneously in various military application areas eventually to be integrated through on-line communication and information exchange.

The growth of the system is anticipated as a by product of its use. To this end it is planned that it will be applied at the University of Pennsylvania to a variety of problem areas. One application, which is planned as extension

---

1 See "Control, Input-Output and Editing Functions in the Problem Solving Facility," by Michael S. Wolfberg, Report No. MS-65-31, prepared under Contract NOnr, 551(40) for Office of Naval Research, Information Systems Branch by the University of Pennsylvania, June, 1965.

of work under contract NOnr 541(48) involves military decisions.

The faculty and student have diverse interests ranging from improving the facility itself and changing it to reflect the demand on the system to the use of the facility for advancing basic research in Information Sciences. Other projects in planning range from automated library retrieval in Law to codification and classification of X-ray pictures of interest in anthropology. Three areas of planned research are briefly outlined below.

Addition of Programming Languages and Translators to the Facility: Limited on-line programming exists at the present, where the user may use the MULTILANG Executive Language, which allows interpretive execution of programs called by the user, and a MAP assembler which allows symbolic programming of the IBM 7040. We plan to use the capability of programming from the CRT consoles in ALGOL and FORTRAN. In addition we plan to undertake the programming of translators, which will allow the user to address the system in a restricted class of English sentences to be translated into MULTILANG statements and thereafter executed.

Use of the Facility as a Teaching Machine: It is important that the facility itself should aid in instructing the new user in its method of operation. This would consist of addition of information storage and retrieval programs for automatic indexing, automatic classification and display of the organization of the information in the system in terms of hierarchical diagrams.

The facility may be used in conjunction with one course in the Computer Information Sciences curriculum. For example, theorems in one area, such as Graph Theory, Combinational Logic or Sequential Systems may be stored in the Associative memory and explored with a combined graph and formula manipulation language. This combination would allow experimentation in analysis and in synthesis. In this type of investigation pictorial displays of algebraic

structures would prove useful.

Systems Programming: Enhancement of the system's programs should continue concurently with research in the above areas would include addition of display devices, addition of compiling languages, experiments in multi-programming and implementation of multi-computer supervisory systems to communicate with other computers in the University community.

REFERENCES

1. "A Concept for Real Time Strategic Navy Command and Control System," (Prepared under Contract NOnr 4366(00)), Computer Command Control Co., Philadelphia, Pa., April, 1965.
2. IBM 7040/7044 Operating System, Macro Assembly Program (MAP) Language, IBM, 1963.
3. IBM System/360 Principles of Operation, IBM, 1965 (Form 822-6821-0).
4. Bauman, R., et al., Introduction to ALGOL, Prentice Hall, N.Y., 1964.
5. Dunn, T.M. and Morrissey, J.H., Remote Computing - An Experimental System, Part I, Proc. S.J.C.C., 1964, pp. 413-443.
6. Fano, R.M., "The MAC System: A Progress Report," in Computer Augmentation of Human Reasoning, M. Sass and Wm. Wilkinson, ed., Spartan Books, N.Y., 1965.
7. Feigenbaum, E.A. and J. Feldman, Computers and Thought, McGraw-Hill, N.Y., 1964.
8. Freedman, Harry A., "A Storage and Retrieval System for Real-Time Problem Solving," University of Pennsylvania, The Moore School of Electrical Engineering, Technical Report on Contract NOnr 551(48), May, 1965.
9. Fried, G., et al., "The TRW 2 Station On-Line Scientific Computer," in Computer Augmentation of Human Reading, M. Sass and Wm. Wilkinson, ed., Spartan Books, 1965.
10. Gelernter, Herbert, "Machine Generated Problem-Solving Graphs" in Mathematical Theory of Automata, J. Fox, ed., Polytechnic Press, Brooklyn, N.Y., 1962.
11. Gorn, S., "Common Programming Language Task, Final Report," The Moore School of Electrical Engineering, University of Pennsylvania, (Signal Corps. Contract DA-36-039-sc-75047), 1959.
12. IBM 7040/7044, Programmer's Guide, IBM, 1963.
13. ....., Systems Programmer's Guide, IBM, 1963.
14. Hempel, Carl G., "Fundamentals of Concept Formation," International Encyclopedia of Unified Science, Vol. II, No. 7, University of Chicago Press, 1952.
15. Landauer, Walter I., "The Balanced Tree and Its Utilization in Information Retrieval," TRANS IEEE (PTGEC), December, 1963, pp. 863-871.

16. Minsky, M., "A Selected Descriptor Indexed Bibliography to the Literature on Artificial Intelligence," in Reference 7.

17. Newell, A., et al., Information Processing Language V Manual, Prentice Hall, N.Y., 1964.

18. Newell, A., Shaw, J.C., and Simon, H.A. "A Variety of Intelligent Learning in a General Problem Solver" in Self-Organizing Systems, Yovits and Cameron, ed., Pergamon Press, N.Y., 1960.

19. Newell, A. and H.A. Simon, "The Logic Theory Machine," IRE Trans. on Information Theory, Vol. IT-2, No. 3, 1956, pp. 61-79.

20. Polya, George, Patterns of Plausible Inference, Princeton University Press, Princeton, N.J., 1954.

21. Prywes, N.S., "Browsing in an Automated Library Through Remote Access," in Computer Augmentation of Human Reasoning, M. Sass and Wm. Wilkinson, ed. Spartan Books, N.Y., 1965.

22. Prywes, N.S., H.J. Gray, et al., "The MULTILIST System," (2 Vols.), University of Pennsylvania, The Moore School of Electrical Engineering, Technical Report on Contract NOnr 551(40), November 1961 (AD 270573, 270572).

23. Prywes, N.S. and S. Litwin, "The MULTILIST Central Processor," Proc. of the Workshop Symposium on New Concepts for Computer Organization. Held at Westinghouse Air Arm Division, Baltimore, Md., October 2, 1962. (Proceedings pub. by Spartan Press, 1963).

24. Samuel, A.L., "Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, Vol. 3, July, 1959, pp. 211, ff. (Reprinted in (I)).

25. Samuel, A.L., "Time Sharing on a Multiconsole Computer," Mass. Institute of Technical Project MAC Report MAC-TR-17, March, 1965.

26. Schwartz, J.I., "The SDC Time Sharing System," Determination Magazine, Part I, November, 1964, Part II, December, 1964.

27. Shaw, J.C., "The JOSS System, Finie Sharing at Rand," Automation Magazine November, 1964, pp. 28-31.

28. Stevens, Mary E., "Automatic Indexing: A State of the Art Report," National Bureau of Standards, MONOGRAPH 91, March 30, 1965.

29. van Dam, A., "A Survey of Pictorial Data Processing Techniques and Equipments," (Prepared under Contract NOnr 551(40)), The Moore School of Electrical Engineering, University of Pennsylvania, January, 1965.

30. Wexelblat, R.L., An Experiment in Computer Learning Based on the Game of Renjya. Unpublished Master's Thesis, 1961.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

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

1. ORIGINATING ACTIVITY *(Corporate author)*
University of Pennsylvania
The Moore School of Electrical Engineering

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
A Problem Solving Facility

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report

5. AUTHOR(S) *(Last name, first name, initial)*
Moore School, Research Staff

6. REPORT DATE
20 July 1965

7a. TOTAL NO. OF PAGES
68

7b. NO. OF REFS
32

8a. CONTRACT OR GRANT NO.
Nonr 551(48)

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Moore School Report No. 66-02

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES
Qualified requesters may obtain copies of this report from DDC

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Department of the Navy
Office of Naval Research
Methodology Division

13. ABSTRACT

The objective of the reported work is to set up a computer with a large memory for on-line, real time use to aid in human problem solving, combining the computational abilities of the computer and its ability to store, retrieve and manipulate large masses of data. Information retrieval programs use Multilist techniques to simulate an associative memory. Multilang, the Executive language, serves both as a control language and as a programming language.

DD FORM 1 JAN 64 1473

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
General Problem Solving Real-Time Time Sharing Information Storage and Retrieval Man/Machine Communication Multilist Multilang Executive System Executive Language Associative Memory						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.