NASA CR-132702

FEASIBILITY STUDY FOR THE IMPLEMENTATION OF NASTRAN ON THE ILLIAC IV PARALLEL PROCESSOR

by Eric I. Field

Prepared under Contract No. NAS1-12436 by
UNIVERSAL ANALYTICS, INC.
7740 West Manchester Boulevard
Playa Del Rey, California 90291

Subcontractor to
McDONNELL DOUGLAS ASTRONAUTICS COMPANY
Huntington Beach, California 92647

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

FOREWORD

This document presents the Final Report on the feasibility study of modifying the NASTRAN (NASA Structural Analysis) computer program to make it execute efficiently utilizing the new ILLIAC IV "parallel" processing computer. This effort was sponsored by the NASTRAN Systems Management Office (NSMO) of the National Aeronautics and Space Administration, Langley Research Center under contract with and monitored by McDonnell Douglas Astronautics Company - West (MDAC).

Universal Analytics, Inc. (UAI) performed this effort under MDAC Contract No. 6-74-489 H, supported by the MDAC prime contract with NASA, No. NAS1-12436. The principal investigation was performed by Dr. E. I. Field with the able technical support of Mr. R. L. Hoesly and Mr. S. E. Johnson and clerical support of Ms. C. Lassen. UAI extends its appreciation to MDAC's Dr. E. L. Stanton for his welcomed critique and to Mr. H. P. Adam for his support and contract monitoring efforts. Also, UAI thanks Mr. J. L. Rogers, Jr. of NSMO for his and the many helpful comments and suggestions offered by the NASA - LRC reviewing team.

This activity, however, could not have succeeded without the over two years of contract support provided UAI by the Advanced Research Projects Agency (ARPA) and supervised by the Defense Nuclear Agency (DNA001-72-C-0108) to develop ILSA, a general purpose finite element structural analysis program for the ILLIAC IV.

This report contains only unclassified material.

ABSTRACT

The ILLIAC IV, a fourth generation multiprocessor using "parallel" processing hardware concepts, is now operational at Moffett Field, California. Its capability to excell at matrix manipulation, makes the ILLIAC particularly well suited for performing structural analyses using the finite element displacement method. The NASTRAN Systems Management Office (NSMO), therefore, contracted with Universal Analytics, Inc. (UAI) to study the feasibility of modifying the NASTRAN (NASA Structural Analysis) computer program to make effective use of the ILLIAC IV.

This report summarizes the characteristics of the ILLIAC and the ARPANET, a telecommunications network which spans the continent making the ILLIAC accessible to nearly all major industrial centers in the United States. Two distinct approaches are studied: (1) Retaining NASTRAN as it now operates on many of the host computers of the ARPANET to process the input and output while using the ILLIAC only for the major computational tasks, and (2) Installing NASTRAN to operate entirely in the ILLIAC environment.

Though both alternatives offer similar and very significant increases in computational speed over modern third generation processors, the full installation of NASTRAN on the ILLIAC is recommended. Specifications are presented for performing that task with manpower estimates and schedules to correspond.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The potential speed and power of fourth generation computers is being realized. Though the electronics of computers has now reached a capacity to perform up to 1.6 millions of arithmetic operations per second, the architecture of existing computers has organized these electronics using multiprocessing techniques to overlap these operations and effectively magnify this capacity to provide up to 50 million such operations per second. These fourth generation computers, such as the ILLIAC IV with "parallel" processing and the STAR with "pipeline" processing, also introduce the opportunity and necessity for developing new computational algorithms. This opportunity has stimulated government agencies, such as NASA, to initiate studies to determine the feasibility of performing finite element structural analyses with these new fourth generation multiporcessor computers.

In response to this challenge, the NASTRAN Systems Management Office of NASA (NSMO) requested an evaluation study to determine the requirements for possibly installing NASTRAN (NASA Structural Analysis), a finite element analysis program, on the ILLIAC parallel processor. In response to this request, Universal Analytics, Inc. (UAI) has prepared the following feasibility study under the direction of McDonnell Douglas Astronautics Company - West (MDAC). This study has been performed under MDAC Subcontract No. 6-74-489 H as part of MDAC's overall contract NAS1-12436 with the NASA Langley Research Center.

The ILLIAC, which is now operational at Ames Research Center, Moffett Field, California, provides unique capabilities already demonstrated [Ref. 1,2,3,5] to be eminently suited to the matrix processing constructs of finite element analyses as currently executed with the NASTRAN program. The multiprocessing capabilities of the ILLIAC are achieved using parallel processing hardware concepts. Very simply, the ILLIAC physically consists of 64 small computers or Processing Elements (PE) directed by a single Control Unit (CU): (1) to execute a common instruction sequence (2) simultaneously (3) on independent sets of data. These three key phrases contain the essense of parallel processing. In other terms, related to structural analysis, these 64 PEs can be used to compute 64 beam element stiffness matrices simultaneously or they can be programmed to decompose a matrix by simultaneously processing 64 columns of that matrix. Thus, with parallel processing, the ILLIAC can provide 64 times the effective speed of a single series processing computer.

Installation of NASTRAN on the ILLIAC would provide a centralized, efficient, and sophisticated analytical tool to potential users in nearly every major industrial center in the United States. Not only does the ILLIAC provide its users with a significant increase in computer efficiency, it is now accessible to many government and university users throughout the nation via the high-speed communications ARPA Network (ARPANET) of small, medium, and large computers. In the future, the ILLIAC will be available to a broader spectrum of users for government-related and/or government sponsored projects. So far, however, specific provisions have not been made for a charging algorithm designed for individual job accounting.

Based on the working knowledge gained by the UAI staff in developing structural analysis software for the ILLIAC under contract with the Defense Nuclear Agency

(DNA) and sponsored by the Advanced Research Projects Agency (ARPA), two basic approaches were studied for installing NASTRAN on the ILLIAC. One approach would be to utilize the large CDC, UNIVAC, and IBM computers of the ARPANET as host processors to perform selected NASTRAN input and output processing and to use the ILLIAC for the major computational tasks of NASTRAN. This "ILLIAC + Host" approach is compared to an "ILLIAC-Only" approach which assumes NASTRAN would be installed in its entirety on the ILLIAC. The results of this study provide the basis for the following conclusions:

1. Installation of NASTRAN using the "ILLIAC-Only" alternative is feasible and the preferred approach.

2. A complete NASTRAN system could be installed and available for execution in approximately 18 months. Full optimization would require an additional 18 to 24 months.

3. User access to NASTRAN on the ILLIAC would be possible via ARPANET from nearly every major industrial center throughout the United States.

The justification for expending the estimated 60 man-months for initial installation and an additional 50 to 75 man-months for full optimization of NASTRAN on the ILLIAC can be summarized as follows:

1. Major computational activity, such as matrix decomposition, can be performed 25 to 130 times faster on the ILLIAC as compared to current third generation serial processors.

2. The ILLIAC is ideally suited to processing large static, dynamic and nonlinear analyses involving highly iterative techniques which heretofore have not been practical on conventional serial processors.

3. Finally, the centralization of maintenance and development efforts on the ILLIAC would reduce future costs of maintenance and would provide ILLIAC users immediate access to all such improvements.

Also presented in this study are the justifications for selecting the ILLIAC over other multiprocessing computers. These may be summarized as follows:

1. The ILLIAC is now operational and has been demonstrated to be highly suited for the matrix processing operations of NASTRAN.

2. The ILLIAC can perform from 1 to 1.7 times more arithmetic operations per second compared to the most efficient alternative, the CDC STAR pipeline processor.

3. The ILLIAC is currently available via the ARPANET from nearly all parts of the nation.

To achieve these significant advantages, however, care must be exercised during program design in order to overcome certain limitations of the

ILLIAC System hardware. These limitations, which are fully described in Chapter II of this study, are as follows:

1. The ILLIAC System is dedicated to the execution of only one job at a time. Therefore, the effective total time of execution must include all phases of processing.

2. Data transfers from Central Memory to the ILLIAC disk is slow compared with the transfer rates from that disk to ILLIAC memory. This imposes practical limits on the size of problems to be solved.

3. The total space available on the ILLIAC disk is also limited, thereby accentuating the problems of slow data transfer to and from Central Memory which would be required during execution of very large problems.

These limitations, when properly accounted for, do not discount the impact the ILLIAC can have in providing the structural analysis community with a new increase in computer capability.

The background information for this study, from which each of the above conclusions was derived, is presented below. The current Level 15.7.7 [Ref. A, B, C], a precursor to the upcoming Level 16 release of NASTRAN, was assumed throughout. The following chapters are organized to present first the background information on the ILLIAC hardware and available software, and the design criteria for selecting the best approach to installing NASTRAN. Then, based on the selection of the "ILLIAC-Only" approach, the design specifications, the task identification, and the cost and schedule estimates for implementation are presented. The last chapter presents an overall summary and the conclusions derived from this feasibility study.

CHAPTER II

ILLIAC IV SYSTEM BACKGROUND

Background information relating to the ILLIAC computer, its supporting hardware and available software is presented here. Historical perspective is also provided to illustrate the magnitude and complexity of the tasks involved in developing a fourth generation computer system with the power and capacity of the ILLIAC [Ref. 6,7,9 - 13].

ILLIAC, which is now operational, offers an exceptionally high computational throughput compared to current third generation computers. This high rate is achieved principally through the parallel structure of its processor. This structure is pictured in Figure 2.1. It consists of 64 small computers called processing elements, or PEs. These 64 PEs are managed by a single control unit, or CU, which directs the execution of the same instruction sequence in each of the 64 PEs simultaneously. However, because the actual data in each PE may be different, the effect is to achieve a computational power as much as 64 times more effective than that of a conventional serial processor.

For example, in terms related to finite element analyses as performed by NASTRAN, the power of the ILLIAC can be achieved by:

1. Generating 64 beam element stiffness matrices at the same time.
2. Decomposing 64 columns of a stiffness matrix simultaneously.
3. Performing forward and backward substitution on 64 sets of right-hand sides simultaneously.

These examples of efficiency illustrate the optimum potential of the ILLIAC. The significance of the ILLIAC's parallel processing power is magnified when compared with other modern serial and multiprocessing computer systems such as the new CDC STAR pipeline processor, CDC 7600, IBM 360/95, and IBM 360/75* as shown below in Table 2.1 [Ref. 3]. This table compares the number of operations (in millions per second) that can be achieved by each of these computer systems. Two extremes are shown: one for serial operations (N = 1), and one for multiple processing with optimum efficiency (N = ∞), Assuming optimum efficiency, the ILLIAC would be

1. 180 to 300 times faster than the IBM 360/75
2. 10 times faster than the IBM 360/95
3. 8 to 9 times faster than the CDC 7600
4. 1 to 1.7 times faster than the CDC STAR

These same comparisons, assuming now a serial mode of operation (N = 1), show that the ILLIAC in its least efficient mode is still faster than all but the CDC 7600. Therefore, if only modest utilization of its parallel processing capability is achieved, the resulting increase in efficiency over that of current third generation computers would be quite significant.

*Of these, only the IBM 360/75 does not have some form of multiprocessing capability.

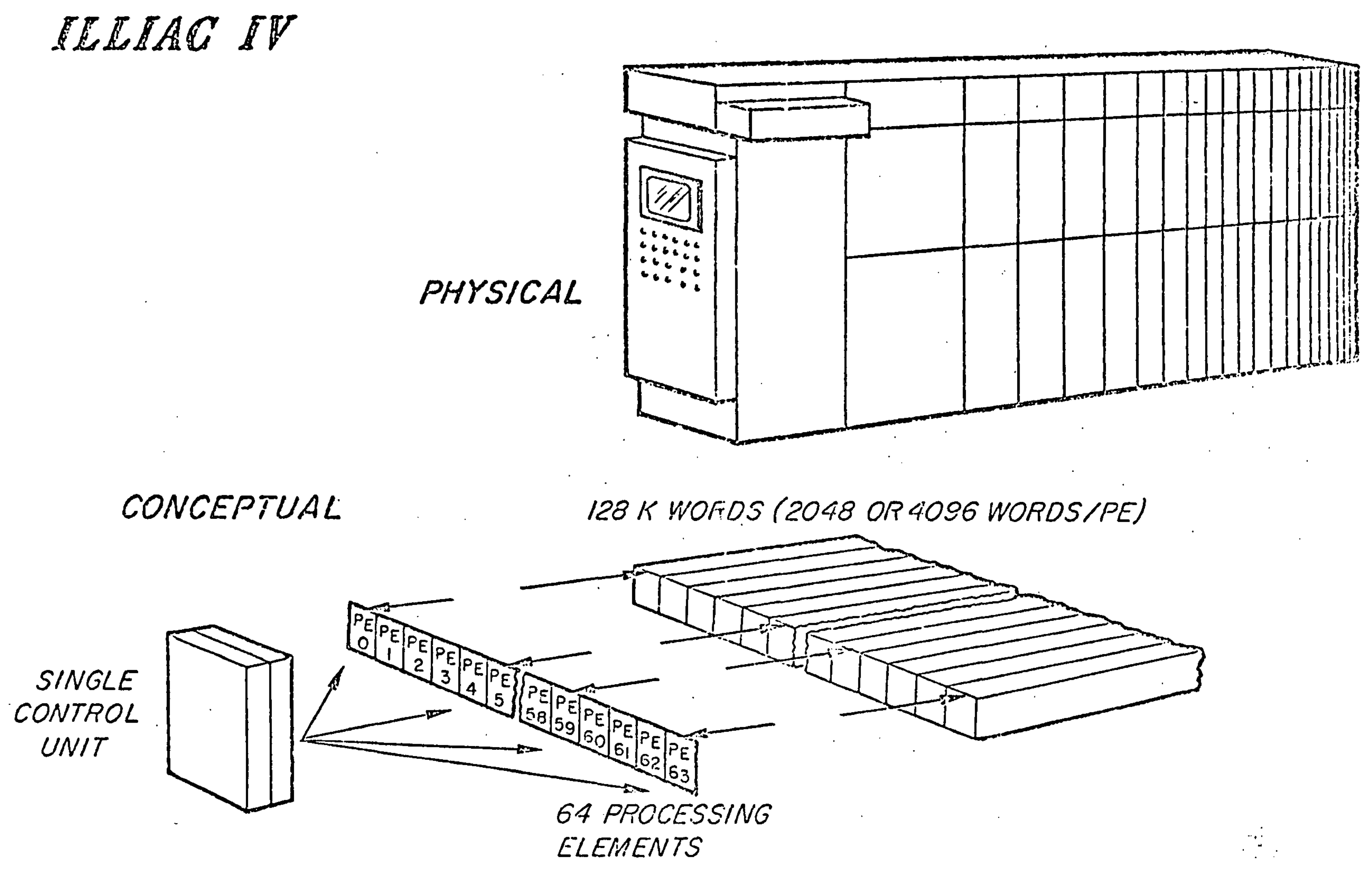

FIGURE 2.1. STRUCTURE OF THE ILLIAC IV HARDWARE

TABLE 2.1. COMPARATIVE SPEEDS OF EXECUTION

Operation	Steps per stage	IBM 360/75	IBM 360/95	CDC 7600	CDC STAR	ILLIAC IV
Addition	$N = \infty$	0.24	4.6	5.2	50	50
	$N = 1$	0.24	0.55	1.6	0.57	0.78
Multiplication	$N = \infty$	0.14	4.6	5.2	25	44
	$N = 1$	0.14	0.53	1.5	0.57	0.69
Division	$N = \infty$	0.096	1.7	2.0	12.5	17
	$N = 1$	0.096	0.43	0.93	0.56	0.27

64-bit precision computation speeds (memory to memory) in millions of operations per second.

For the ILLIAC IV and STAR, the figures are preliminary and for illustrative purposes only. For the 360/195 and 7600, the numbers are sensitive to the way the smaller, faster memories are used.

The ILLIAC processor, however, is only one component of an entire complex of computer systems accessible via the ARPA Communications Network (ARPANET) shown in Figure 2.2. As can be seen in Figure 2.2, the ARPANET offers the user access to the ILLIAC from many parts of the United States. This accessibility is one of the major justifications for implementing a nationally recognized applications package, such as NASTRAN, on the ILLIAC.

The first of the two major sections to follow includes a more detailed view of the ARPANET facilities followed by a detailed description of the ILLIAC and its supporting hardware. This discussion of the ILLIAC hardware focusses on the central system of PDP-10 computers that support and control the ILLIAC, the specific requirements and facilities for data storage, the data file transfers capabilities, and the major hardware constraints which would affect installation of NASTRAN on the ILLIAC. These constraints, which affect the specific design criteria and program specifications discussed later in Chapters 3 and 4 respectively, may be summarized as follows:

1. The ILLIAC is a system dedicated to processing of a single job at a time without overlap.

2. The rate of data transfer from the central system to the ILLIAC IV Disk Memory (I4DM) is from 10 to 50 times slower than from the I4DM to ILLIAC memory.

3. The rotational speed of the I4DM is relatively slow compared to execution time on the ILLIAC which tends to cause processing to be I/O bound.

4. Data transferred from one PE to disk may be read back only into the core memory of that same PE.

This section on the system hardware is then followed by a discussion of the software packages available for process control and applications program development. The best of the application languages are compared and a tabulation of their key features is presented. This is followed by a summary comparison and identifying the selection criteria. The major language section criteria involve:

1. Access to unique hardware features of the ILLIAC
2. Variety of I/O capability
3. Future of maintenance and support

The eventual choice of language to be used depends on which alternative approach is selected for implementing NASTRAN. Therefore, the final selection is made part of the specifications for implementation presented in Chapter 4.

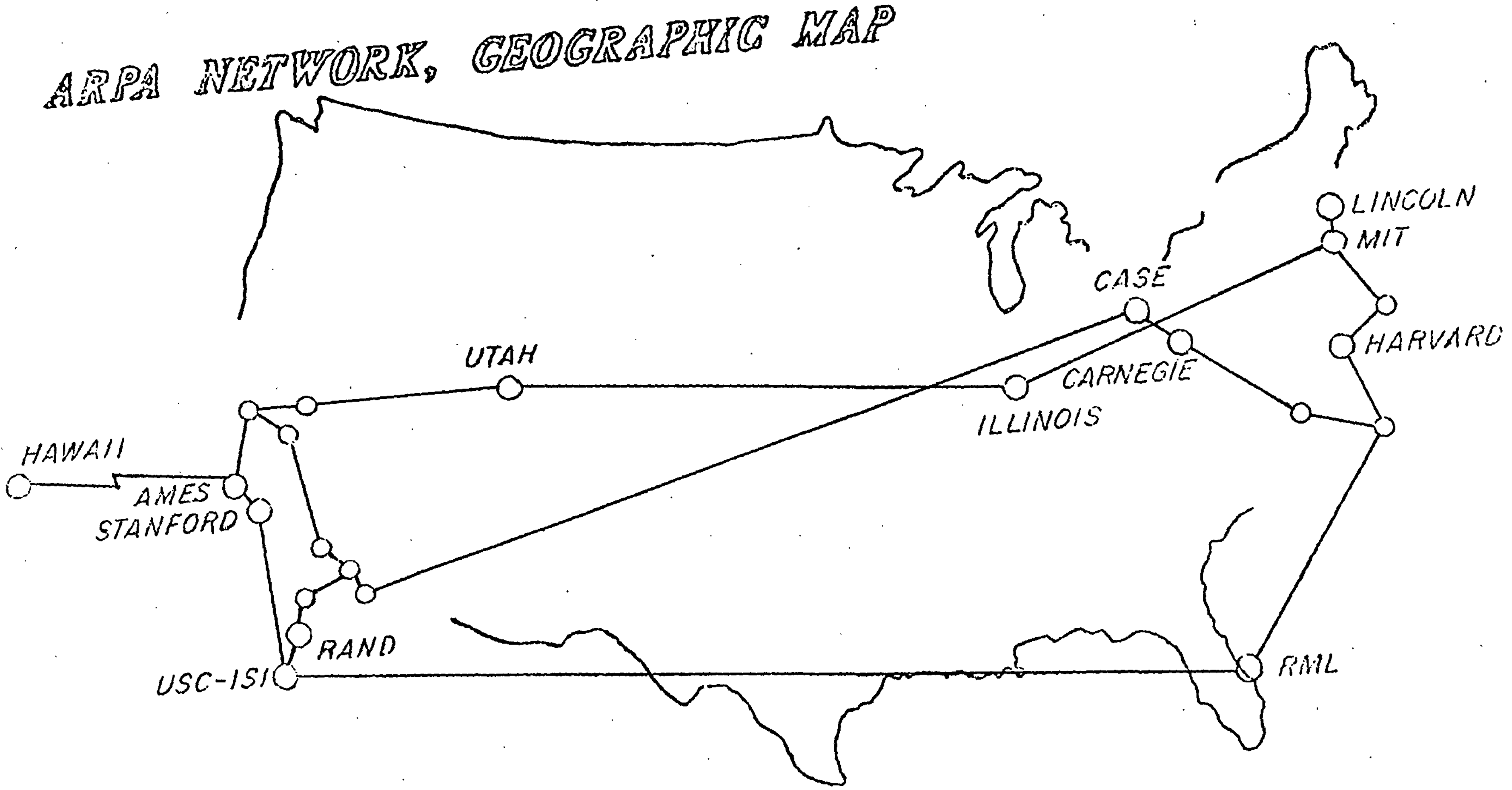

FIGURE 2.2. GEOGRAPHIC MAP OF ARPA NETWORK (1974)

COMPUTER AND NETWORK HARDWARE

The system that includes the ILLIAC IV processor is an integrated system of data processing, information storage, and communications equipment located at NASA Ames Research Center, Moffett Field, California. The system is accessed, as shown in Figure 2.3, via a high-speed communications network (ARPANET) which provides large-scale computational and file management services to a growing community of users located throughout the nation. Through ARPANET, specialized analysis and research groups seeking solutions to a range of contemporary problems have access to required computational power not otherwise available to them, and not economically feasible on an individual or regional basis.

The ILLIAC system is being developed at Ames Research Center by the NASA Institute for Advanced Computation (IAC). The Advanced Research Projects Agency, ARPA, of the Department of Defense underwrote the research and production that were required in the early phases of development. Since 1971, the system at Ames has been sponsored and managed by a multi-agency board of owners, and today IAC receives its technical direction from this board. The board currently includes representatives of NASA, ARPA, and IAC.

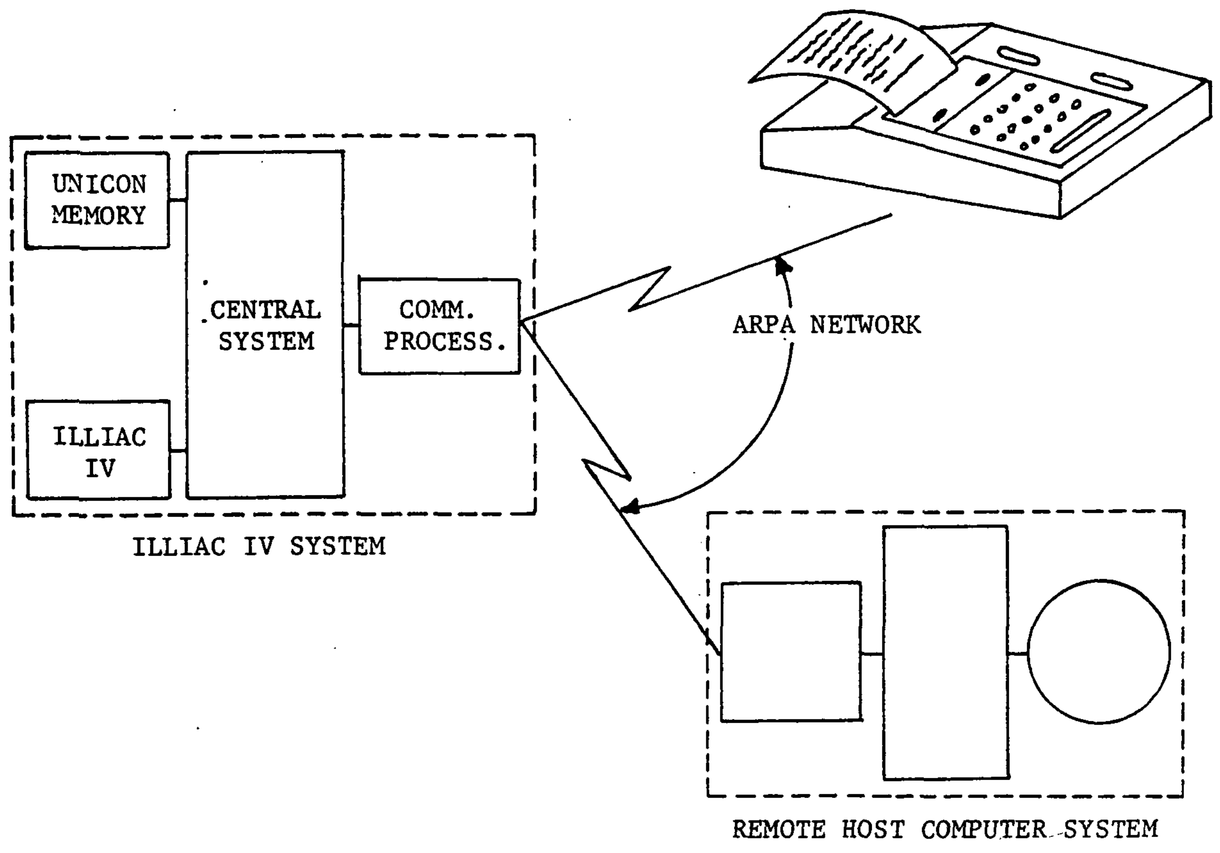

FIGURE 2.3. SYSTEM ACCESS VIA ARPANET

The system being developed by IAC is the largest of several systems of computing resources available to users through ARPANET. Other resources, see Figure 2.4, vary from small interactive systems to large conventional processors. This network approach to the distribution of computer system services is one of the principal economic justifications for the development and operation of large systems - on the scale of the IAC System - for users whose applications require such systems but who cannot economically justify a dedicated system. The small systems on the network are used for communications and for program and data preparation tasks, allowing larger resources to be applied to tasks commensurate with their special capabilities.

The potential value of the system at Ames is greatly enhanced by its accessibility to a large number of disparate and geographically remote users through the ARPA communications network. All users, wherever located, access the system remotely via ARPANET for both data transfer and interactive submission of processing requests. Therefore, the ARPANET is described first, followed by a summary of the ILLIAC System design specifications and a summary of the critical hardware features which will affect the implementation of NASTRAN.

The ARPA Network

ARPANET is a wide-band (currently 50,000 bits per second) [Ref. 8] communications network linking together computing centers and terminal access points throughout the country. The network has been constantly growing, both in the number of computing facilities available through it and in its geographic extent, since its inception in 1969. It should be noted that the participating institutions are primarily universities, with a sprinkling of public and private research organizations.

A primary purpose underlying ARPA's sponsorship of the network is to establish the feasibility and to develop the technical and management basis for computer system networks on a large scale. In the future, data rates in excess of one million bits per second are expected over the ARPANET.

ARPANET not only represents a linking together of equipment, but also includes a standard set of management procedures and communications protocol. For example, network procedures include a file transfer facility that allows the easy transfer of files to and from virtually any computing center in the network. There is also a high degree of standardization in the control languages of the various network systems.

Network users of any of the available computing and data storage services interact with the resources of their choice, either directly through data terminal devices tied into the network, or indirectly through interaction with local computer centers which are in turn tied into the network. A set of geographically dispersed network resources can be readily combined, and each applied to an appropriate part of the solution of a single problem. Regardless of how the resources of the system are intended to be used, the procedures for access and interaction are generally the same. The potential of the network as a resource for servicing and making available the multitude of current and yet-to-be-developed structural mechanics programs is obvious.

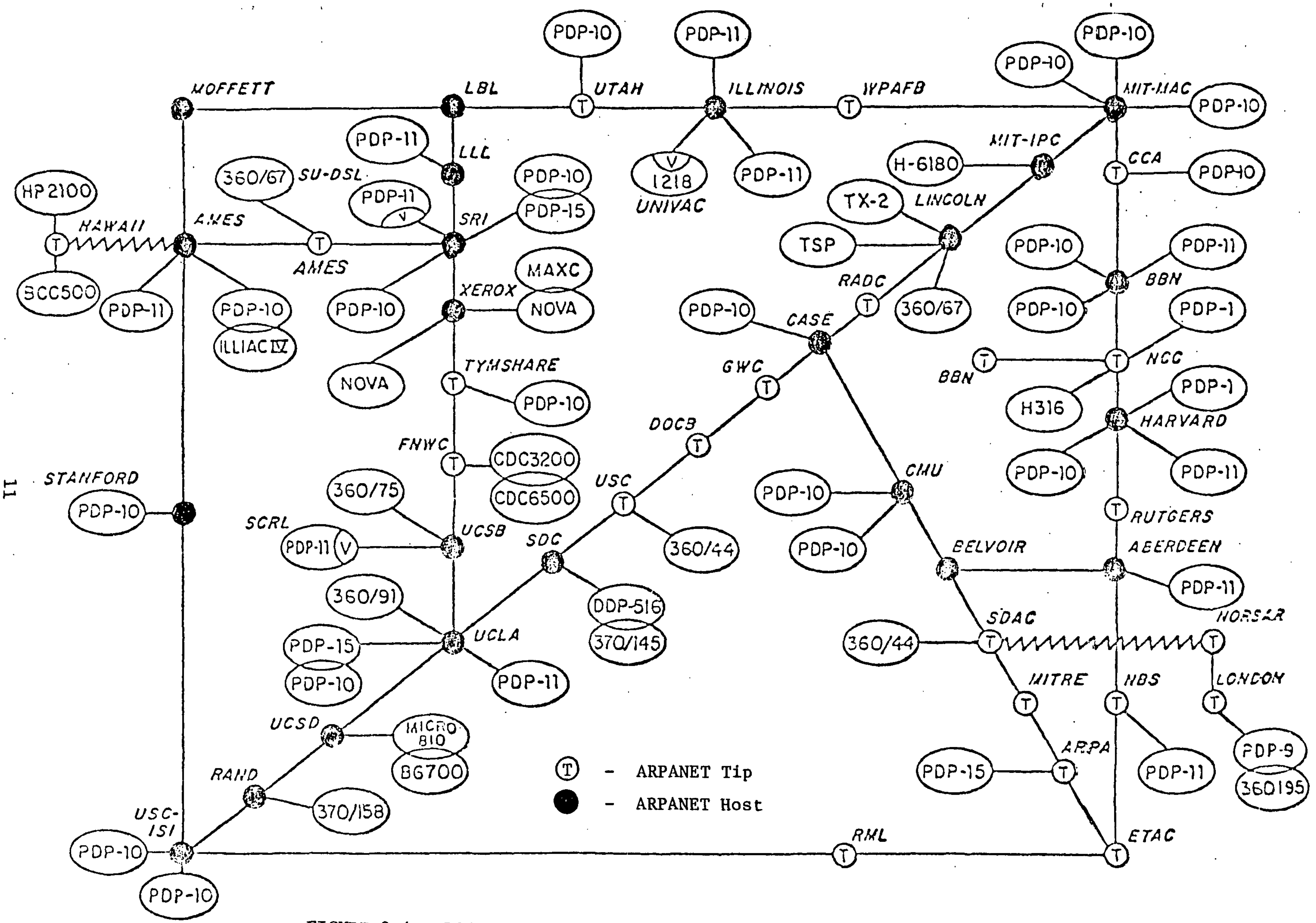

FIGURE 2.4. LOGICAL MAP OF ARPA NETWORK RESOURCES, JANUARY 1974

The major centers available to the ARPANET, which are shown in Figure 2.4, illustrate the generality of inter-system communication capabilities. For example, the current development efforts at UAI have utilized several components of this system. These include the IBM 360/91 at the University of California at Los Angeles (UCLA), the Burroughs B6700 at the UC San Diego (UCSD), and of course, the ILLIAC IV System at Ames, Moffett Field, California. The IBM 360/91 at UCLA provides the host computer facilities on which NASTRAN has been installed and modified to transmit basic structural data and the control files for execution on the ILLIAC. Once a solution is obtained, the results are communicated back to the IBM 360/91 host for the selective output processing as requested by the user. Also, the B6700 at UCSD was used via ARPANET during initial code development and testing via the SSK ILLIAC System simulator. This effort, however, is no longer required as the ILLIAC is now available directly for code development and testing.

The TIP (Terminal Interface Processor) at USC is used as the local resource for interaction with the ARPANET. Any other TIP (marked by Ⓣ in Figure 2.4) could be used interchangeably for dial-up typewriter terminal activities. Via the TIP, a user can remotely control files and execution on any host to which he is given access on the ARPANET. This feature allows a user in Boston, for example, to initiate computations on the UCLA IBM 360/91 for creation of his input files for the ILLIAC, to transfer these files to Ames and to process them on the ILLIAC, to return the results to UCLA for output processing, and finally to interrogate these files selectively or have them shipped to a local site in Boston for volume printout.

The hardware facilities of the ILLIAC IV System at Ames are described next.

The ILLIAC IV System Hardware

The ILLIAC IV System at Ames, which consists of a number of computing elements, is a remotely accessible facility available to a variety of user groups as described above. Figure 2.5 is a high-level block diagram showing the major functional elements of the system as it is now in operation. Looking at these elements first from a user's point of view, the communications processor ties the system to ARPANET. It provides data error checking and performs the necessary communications protocol.

The central processors (PDP 10 and 11, and one Burroughs 6700) control the user job sequences and execute all of the utility programs provided to users by the system. These processors interpret service requests from users and either execute the necessary utility programs or pass on the requests when they are for services provided by other resources in the system. The resource management processors control and allocate the major resources in the system; they do not directly execute user programs. The resources they control include the data storage devices and the ILLIAC processor itself which was pictured earlier in Figure 2.1.

All of the system processors communicate with each other through central memory. Central memory also functions both as a program and data store for the processors, and as an intermediate buffer for data transfers between any two storage devices in the system and between the system and ARPANET. Central memory is thus pictured as the central device in the system in Figure 2.5.

The major devices in the system are the UNICON laser memory and the ILLIAC processor. The UNICON laser memory, with an on-line capacity of 700 billion bits, is the largest storage device in a central file system which also includes a large buffer disk and magnetic tapes. Users do not directly address or specify the UNICON memory for data storage; rather, the location of all data files in the system is controlled by the central file system.

The block labeled "ILLIAC IV" in Figure 2.5 represents both the ILLIAC processor and the large ILLIAC disk memory system, which has a capacity of over 32 million 32-bit words. The ILLIAC processor, which is the major processor in the system, is totally dedicated to the execution of user code. It is seen by the central system as a large peripheral device providing a special user service, namely, very high-speed parallel processing on large data volumes. These services can be used independently or in combination in the solution of a single problem. The separate nature of these services is a result of the architecture of the system. Although fully integrated, it can be functionally utilized as three independent subsystems, namely, the ILLIAC, the information storage subsystem, and the central system.

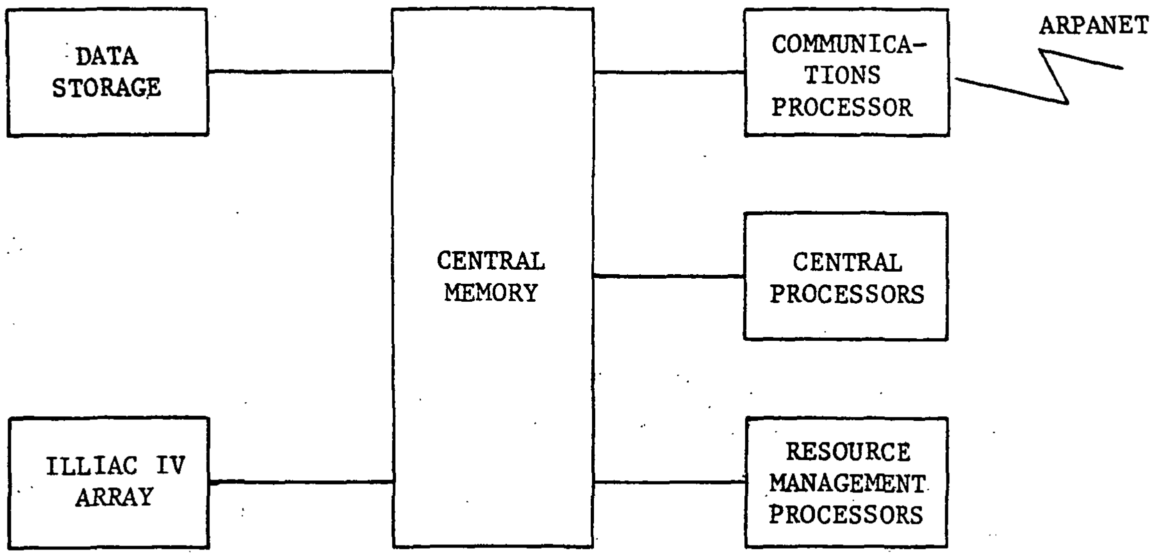

FIGURE 2.5. MAJOR FUNCTIONAL ELEMENTS OF THE SYSTEM

Once the program and data have been stored on the system at Ames, the user defines the general processing tasks to be performed, including those steps to be done on the ILLIAC. This definition can be done interactively or in a batch sequence. The central system then initiates the movement of the data files from the UNICON memory to the ILLIAC memory system. The transfer takes place in a series of steps under control of resource management processors. Once a sufficient quantity of data has been transferred to the ILLIAC disk memory, the program itself is moved from the UNICON to the processor memory, which functions as working storage for the ILLIAC. Execution then begins. Refer to Figure 2.6 for an illustration of data movement within the ILLIAC IV subsystem.

The ILLIAC subsystem, which is distinct from the central system but controlled by it, includes the ILLIAC processor, a 256K (32-bit word) processor memory, and a 32-million-word main disk memory device (see Figure 2.6). The high execution rate of the ILLIAC is achieved principally through the parallel structure of its processor, presented here in general terms.

The ILLIAC processor executes a common instruction sequence simultaneously on a large number of otherwise independent sets of data. This simple description contains the three key phrases in understanding the parallelism implemented in the ILLIAC: (1) "a common instruction sequence", (2) "simultaneously", and (3) "independent sets of data".

The instruction sequence is similar to that of any modern large-scale processor. "Simultaneously" means literally that each instruction in the program sequence executes on every one of the independent data sets at the same time. The number of independent data sets can vary from one up to 512, depending on the required word size of each data item and the skill of the implementing programmer. With a single data stream, the ILLIAC is functionally identical to a conventional, nonparallel processor. The machine architecture, as seen in Figure 2.6, facilitates 64, 128, or 512 parallel data streams which are 64, 32, or eight bits wide, respectively. Both 64-bit and 32-bit words are standard ILLIAC word sizes and are fully implemented in parallel in the hardware and in the programming languages. Eight-bit wide parallelism may be thought of as a byte mode with more limited implementation in the hardware and software.

To achieve this high degree of parallelism, the processor structure consists of a single Control Unit (CU) that performs instruction decoding and program control and 64 arithmetic and logic units referred to as Processing Elements (PEs). The CU reads the instruction sequence stored in processor memory, decodes each instruction, and generates identical control signals for each of the PEs. The entire set of PEs (or any subset within it) executes the same instruction simultaneously, under CU control, each on different sets of data. Under program control, any subset of PEs can be selected <u>not</u> to execute the current instruction.

The instruction set and instruction execution times of each individual PE in the ILLIAC processor equal or exceed those of existing conventional processors. For example, a full 64-bit, floating point, normalized ADD takes approximately 300 nanoseconds. A MULTIPLY under the same conditions executes in about 600 nanoseconds.

There is a single instruction sequence, stored in processor memory. Interleaved in this instruction sequence, normally, are computational steps

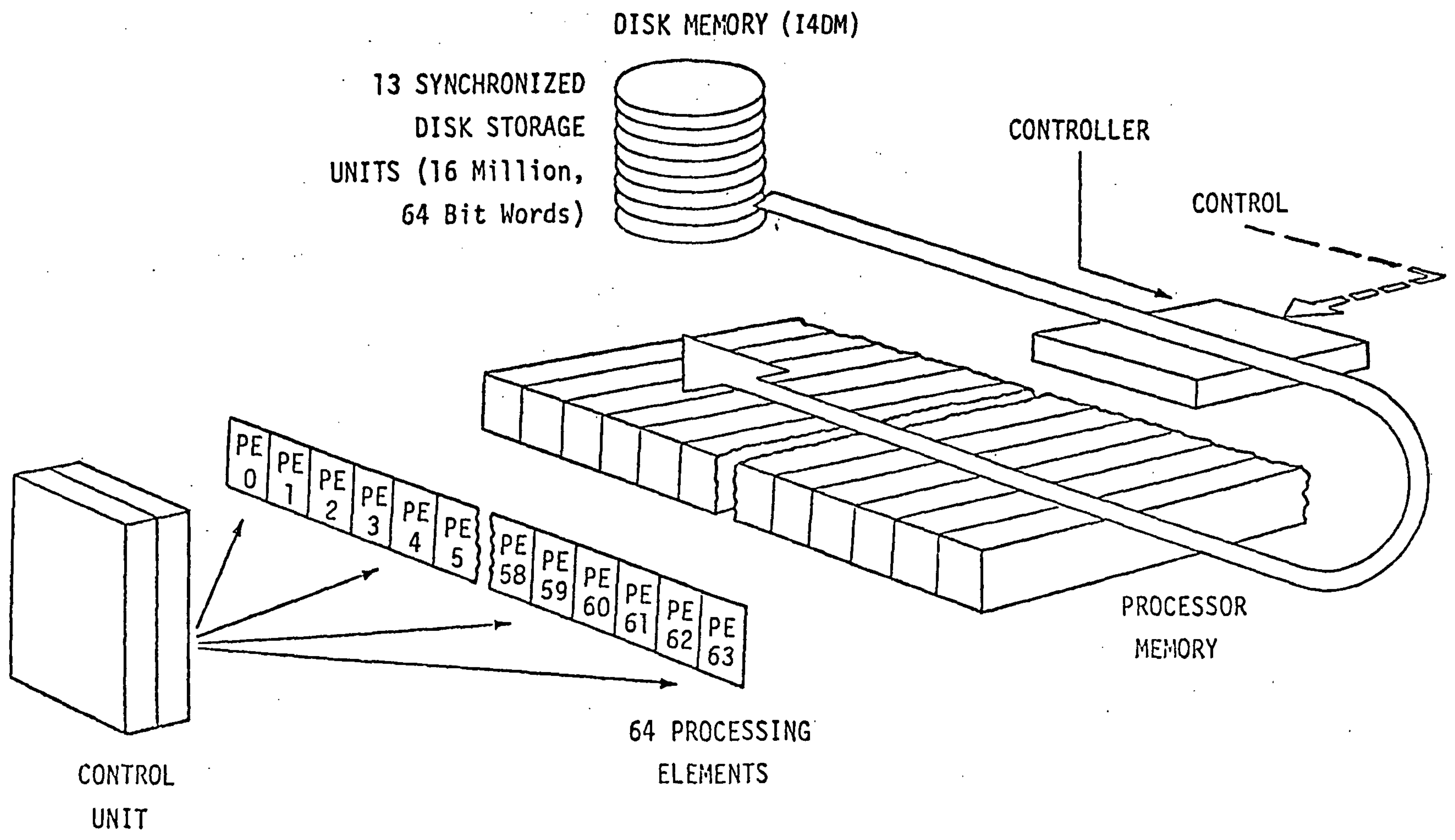

FIGURE 2.6. DATA MOVEMENT WITHIN THE ILLIAC IV SUBSYSTEM

executed by the PEs, and program control instructions executed by the CU itself. Thus the CU, in addition to controlling the PEs, executes instructions such as branching, loop counting, and the generation of external system calls. In order to provide a sophisticated control capacity (for example, complex loop indexing), the CU has a complete instruction set including arithmetic and logic, byte and bit, and special control instructions. The CU is, if viewed as a stand-alone device, a full-scale processor.

Another vital function of the CU is to direct the transfer of data from one PE to another. Each PE memory segment is hard-wired to four neighboring PEs to provide high-speed, core-to-core transfer rates. Two methods of transfer are available to the program user. One is broadcasting, which takes a single word from one PE and broadcasts it to all PEs. The second provides for routing of one word from each PE(i) the same distance j to PE(i + j) for all PEs simultaneously.

The high execution rate of the ILLIAC is achieved not only through the parallel structure of the PEs but also through execution overlap. The execution of instructions within the CU is overlapped in time with the execution of instructions by the PEs. The accessing of instructions from processor memory is overlapped with the execution of these instructions. Finally, PE operand fetches are overlapped with PE instruction execution.

The processor memory is working storage for both instructions and data for the ILLIAC processor. This memory may be thought of as an array of 64 columns and 2048 rows. Each column (which is either two 32-bit words or one 64-bit word in width) is associated with an individual PE. The CU can access the entire processor memory, while each PE accesses only its associated column. As described earlier, however, data can also be moved from one PE to another under control of the CU.

The main memory storage for the ILLIAC is the disk memory, the I4DM. This device is a fixed-head rotating disk system with a capacity of about 32 million 32-bit words. The system is composed physically of 13 disks which rotate synchronously with a 40-millisecond rotation period and which provide a maximum data transfer rate of about 10^9 bits per second. The entire processor memory can be written out to disk or loaded from disk in just one revolution. That is, 256K 32-bit words can be transferred in 40 milliseconds.

System Hardware Constraints

Though the hardware facilities described above offer significant computational speed, massive storage capabilities and unique inter-computer communication features, certain of these facilities impose constraints which must be mentioned. These constraints may be divided into four basic categories which effect the design of any applications program to be implemented. These are:

1. Data transferred from one PE to disk may be read back only into the core memory of that same PE.
2. The rotational speed of the disk is relatively slow compared to the speed of execution so that the system tends to be I/Ø bound for programs that require significant temporary data storage on disk.

3. The relatively slow transfer rates from central memory to the I4DM and the restricted number of I4DM disk drives (compared with current IBM 3330 capacities) place limits on the size of problems to be solved.

4. The ILLIAC is a dedicated machine restricted to the processing of a single job at a time without overlap.

Specifically, the first constraint requires that primary consideration must be given to design of the data structures to be implemented. Files must be organized to avoid as much as possible the necessity to transfer data from one PE memory to another. For example, matrix generation and matrix assembly modules need to be coordinated so that data output from one module is input to the other module in such a manner as to minimize the time required to route the data for assembly. Also, the storage of assembled matrix data should be designed to anticipate the specific PE into which it must be read for subsequent processing. In anticipation of these objectives, attention must be given to the selection of internal numbering schemes, to the design of solution algorithms, to take advantage of numbering conventions in order to minimize storage of pointer data and to the interfacing between modules which operate on these data. Those modules responsible for the major computational efforts probably should be specially coded to account for this constraint.

The second and third constraints impose a design requirement to optimize the allocation of disk space to storage of matrix data. The software facilities are provided, as described later, to "map" or assign data block storage on the disk. Careful attention to estimates for execution time relative to data transfer times is required to design "mapping" algorithms which will avoid potentially lengthy wait periods for disk I/O to take place. Practical considerations, however, dictate that this type of "tuning" of the application software be limited only to the interfaces between major time consuming modules. The relative overall speed of the ILLIAC will compensate for some inefficiencies of less critical modules.

The fourth constraint imposes more global considerations on the design of a large applications software package. The major concerns involved with current serial processors for checkpointing and restarting to protect the user during long running computations should be modified. An examination of the time required to regenerate data versus the time required to checkpoint and then restore that data will be required to establish new checkpointing criteria. The speed and capacity of the ILLIAC and its I4DM will generally show less time is required to regenerate matrices to insure cost effective utilization of the system. The overriding criteria will be to selectively checkpoint only the minimum size data blocks needed for restart rather than to implement the practice of arbitrarily checkpointing nearly every item as is done now in NASTRAN.

These hardware constraints apply primarily to the initial efforts of selecting a basic approach and design methodology for implementing any large application software package. The size of NASTRAN and the generality of its current capability also present major considerations. By retaining its original organization as a <u>modular</u> program, installation of NASTRAN on the ILLIAC can be accomplished utilizing either of two basic approaches:

1. The current NASTRAN system which now can operate on several of the remote host computers on the ARPANET can be modified to extend its

capabilities linking it to the ILLIAC for the execution of selected modules specifically designed for optimal utilization of the power and speed of the ILLIAC.

2. Directly implement all modules of NASTRAN on the ILLIAC first, and then selectively redesign the major modules for optimum utilization of the ILLIAC.

The first option would soon provide powerful and efficient code for a limited set of capabilities. The second option would allow execution of <u>all</u> current capabilities on the ILLIAC but at far from optimal efficiency. However, both options eventually would reach a commensurate level of efficiency, thus providing the NASTRAN user community with access to the most powerful computational facility now available.

The final selection of an approach for installing NASTRAN obviously depends heavily on the available software for its implementation and eventual optimization. The next section, therefore, is devoted to an analysis of the available software.

SYSTEM SOFTWARE

The system software available on the ILLIAC IV is divided into two categories: that used for producing application programs and that used for providing the job control and file control utilities.

There are four suitable application languages presently implemented for use on the ILLIAC:

GLYPNIR	Higher level languages
CFD	
IVTRAN	
ASK	Assembly language

Of these four, all but IVTRAN have seen extensive use on the ILLIAC. CFD is a FORTRAN-based language. It has been used in a wide variety of climatic and fluid dynamics problems. GLYPNIR, one of the first available languages for the ILLIAC, is based on ALGOL. It is being used now for structural analysis with the UAI-ILSA system and by several others who are performing earthquake studies. Many of the ILLIAC users also use ASK in certain applications where more efficient coding is desirable. IVTRAN, which is currently under development, is also a FORTRAN-based language. Though it has not yet seen much use, current studies indicate it is to be highly regarded as a major candidate for major software development on the ILLIAC.

Several additional languages such as COCKROACH and TRANQUIL have received some attention in recent years, but none has reached an operational state.

Two control languages are available to the user. These provide utility services for file editing and job control. The two systems are:

ACL	Job control language
FTP	File transfer services

Both of these languages operate under the TENEX operating system of the central system's PDP-10 computers. These are summarized in the first section to follow. Next to be presented are the four application languages, along with a summary of the key features of the three higher level alternatives. The final section of this chapter presents an overall comparison of the three languages, GLYPNIR, CFD and IVTRAN, to be used later in selecting the language for implementing the NASTRAN capabilities on the ILLIAC.

Process Control Languages

ACL (A Control Language)

ACL is the general control language for the ILLIAC System. It provides linkage between the ILLIAC IV, the central file system, and the PDP-10 system process controllers.

The language operates in two modes, batch and interactive, with several subsystems being restricted to only one mode. The language is sequential in nature with only simple error tests and jumps allowed.

A summary of services supplied is listed below:

1. Compile and Assembly Steps:

   Subsystems are provided to support the GLYPNIR and ASK languages. IVTRAN will be available in the future. No services are available for CFD. These must be obtained through host computers over the ARPANET.

2. Program Execution Steps:

   The normal services exist for submitting and monitoring the execution of user programs on the ILLIAC. Subsystems are also provided for the transfer of data to and from the I4DM and central memory.

3. Linkedit and Map Steps:

   Linking of relocatable code and formatting of the I4DM working disk storage for the ILLIAC are provided with these subsystems:

   <u>Linkedit</u> - Only simple linking procedures are supported. No overlay structures are allowed. Commands are also provided to control the manner in which unsatisfied external references are supplied.

   <u>Map</u> - The map subsystem allows the user to format his areas on the I4DM for use during execution. A utility is also available to produce a printed picture of the resultant disk layout.

4. File Utilities:

   Several subsystems are available for the editing and listing of files stored in the central file system. Three file types are supported, 7 or 8 bit ASCII formatted files, binary files, and ILLIAC Dump files.

   The file editor used primarily for maintenance of source code and output viewing provides very good searching and updating facilities combined with ease of use.

FTP (<u>F</u>ile <u>T</u>ransfer <u>P</u>rotocol)

The File Transfer Protocol is a processor for data transfers between host computers on the ARPA network. The primary function is to transfer data efficiently and reliably among hosts and to allow the convenient use of remote file storage facilities.

FTP provides for only limited data representation. Four data formats are

1. 8-bit ASCII formatted
2. 8-bit EBCDIC formatted

3. Image - continuous stream of binary data
4. Local - data is stored on logical bytes of a size specified by the user

The conversion of data between internal storage representations and the FTP representation is handled by the host computer before or after transfer. The system also provides access controls to prevent unauthorized or accidental use of files.

The following section describes the four primary application languages and summarizes their key features.

Application Languages

GLYPNIR

GLYPNIR is an ALGOL-based language for the ILLIAC with extensions to provide parallel processing capabilities. The GLYPNIR compiler runs on the B6700 computer and generates ASK code to be assembled and executed on the ILLIAC. A simulator (SSK) which runs on a B6700 also exists which does a bit-for-bit ILLIAC simulation of the GLYPNIR-generated machine code.

Being ALGOL-based, the language is quite flexible. Good capabilities exist for string-and-bit manipulation and partial word operations. Sufficient capabilities have been included to make efficient use of hardware capabilities. Direct operations on hardware registers and several routing capabilities are provided. ASK code may be inserted at any point in the GLYPNIR code and may use any GLYPNIR-defined variables. Good storage management facilities exist for assigning data to either PE or CU memory and to hardware registers. No provision is included for code overlaying.

Sufficient I/O capabilities exist for transferring data to and from the I4DM. A crude display feature is provided to format data for later output. The major drawbacks of the language come from its lack of formal support for maintenance and improvement. The language, though it is advanced, also tends to be difficult to program because of the large variety of constructs available to the user. That is, the existing FORTRAN code of NASTRAN could not readily be converted, and a major rewrite would be required for every module to be coded in GLYPNIR.

CFD

CFD is a FORTRAN-based language extended to include access to the parallel processing of the ILLIAC. The CFD code can be either compiled directly into ASK for execution on the ILLIAC, or it may be translated into equivalent FORTRAN for execution on any of the larger IBM computers. This allows a CFD program to be debugged externally from the ILLIAC. The language is actively supported and used at NASA Ames. Being FORTRAN-based, the language is restricted in its flexibility such as limited string-and-bit manipulation. Also, no provisions are available to use some of the hardware features of the ILLIAC. This includes no direct access to registers and limited routing capabilities. It is possible to include ASK code at any location in the CFD source statements. Good data assignment facilities are provided for storing data in either PE or CU memory. The language does not allow any type of mixed mode arithmatic.

Sufficient I/O capabilities exist for transferring data to and from the I4DM. No display feature is provided for formatted I/O and, if desired, must be accomplished with ASK.

Although easy to use, CFD is not as flexible or sophisticated as the GLYPNIR language. Unnecessary syntax restrictions also have been imposed to include parallel processing capabilities. These include the * in Column 6 on most cards and a required (*) subscript on all variables assigned to PE memory.

IVTRAN

IVTRAN is a FORTRAN-based language for use on the ILLIAC. The language syntax is very similar to standard FORTRAN, making conversion from existing code a minimal effort. A major feature of the IVTRAN system is the "paralyzer", a compiler option for use in this conversion effort. The paralyzer examines DØ loops of standard FORTRAN programs and converts them into more efficient DØ FØR ALL loops for use on the ILLIAC.

IVTRAN is being actively developed by Massachusetts Computer Associates. Although it is not available to users at this time, recent benchmarks give encouraging results. The compiler appears to generate very efficient code and has extensive optimization facilities. The user can also aid the compiler in optimization by specifying expected execution frequencies through the FREQUENCY statement.

The I/O facilities of IVTRAN are quite different from those of the other languages. The biggest difference exists in formatted I/O. IVTRAN has very good display features derived from the full implementation of the FORTRAN FØRMAT statement. Binary I/O is of a buffered variety which does not allow for asynchronous processing. A non-buffered bulk I/O is also available for transferring whole IVTRAN arrays. This technique takes better advantage of the I4 hardware and does allow asynchronous processing.

IVTRAN contains all the normal restrictions of a FORTRAN-based language. These include poor string-and-bit manipulation facilities and no partial word operations. However, good debugging facilities and a mixed mode arithmetic feature are included.

One of the principal disadvantages of IVTRAN appears to be its limited utilization of the unique characteristics of the ILLIAC. Although little actual experience with the language is available, to produce code which fully utilizes this parallel processing feature could be difficult. Only one language construct, the DØ FØR ALL clause, deals with this important concept. No direct control is available over PE modes (on or off) or use of CU registers or CU memory. Also, no provision is available for inserting ASK code directly within the IVTRAN code, although separate ASK subroutines may be coded.

The IVTRAN language attempts to relieve the programmer of many of the idiosyncrasies and difficulties of working in a parallel processing environment. The efficient use of parallel processing is highly dependent on the alignment of data in each PE memory. The facility for this in IVTRAN gives the programmer only limited control while the compiler attempts to do the majority of the work. Unfortunately, these attempts appear to have restricted the efficient use by the programmer of these important ILLIAC features.

ASK

ASK is the assembly language for the ILLIAC and is fully supported by the Institute for Advanced Computations. It allows full use of the hardware configuration of the machine. While powerful, coding is difficult and time consuming, as with any assembly language.

I/O capabilities are very good for assembly language programming. Macros are available that provide disk and formatted I/O similar to that of GLYPNIR.

Its major advantage is for use in optimizing selected sections of code. It may be used as direct inserts into GLYPNIR and CFD program code and as a macro language for subroutine constructs.

Key Language Features

There are many features that determine the suitability of a language for any application. Table 2.2 summarizes the basic features for easy comparison of the three best higher level languages available on the ILLIAC: GLYPNIR, CFD, and IVTRAN. The more important of these features are identified below.

The user should have direct access to and control over the hardware functions unique to the parallel processing environment for optimal program design. These include:

1. Routing of data between PEs.
2. Access to the hardware registers.
3. Access to the small, high-speed memory of the CU.
4. Direct control over the operational mode of each PE (on or off).
5. Access to asynchronous I/O capability.
6. Ability to insert ASK code where needed to gain efficiency for computation and I/O.
7. Access to central file system from the ILLIAC.

Other important features to be considered are:

1. Available data types, including string and complex.
2. Communication facilities between subroutines, including calling parameters and common.

TABLE 2.2. COMPARISON OF LANGUAGE FEATURES

	GLYPNIR	CFD	IVTRAN
Base language	ALGOL	FORTRAN	FORTRAN
Support -			
maintenance	very little	good	good
documentation	poor	good	under development
improvement	none	fair	good
Syntax -			
readability	good	good	good
programability	difficult	difficult	good
reserve words	≈ 120	≈ 25	none
loops	several types	DØ loop	DØ loop
mode control	several types	logical IF	DØ FØR ALL
branches	GØ TØ's	several types	several types
Code generation -			
efficiency	poor	fair	good
optimization	none	none	good
ASK inserts	yes	yes	no
Debugging -			
bound checking	yes	no	yes
subroutine call traces	yes	no	yes
conditional compilation	yes	no	no
variable displays	no	no	yes
Hardware-related features -			
routing functions	yes	no	no
register availability	yes	no	no
ADB* usages	yes	some	no
mode control	yes	direct control	no
Macros	DEFINE	no	no
Subroutines and functions -			
separately compiled	yes	yes	yes
in-line	yes	no	no
Built-in functions -			
trig	yes	yes	yes
log and exp	yes	yes	yes
type conversion	yes	yes	yes
bit manipulation	shifts & rotates	shifts & rotates	shifts

*ADVAST, DATA, BUFFER

TABLE 2.2. COMPARISON OF LANGUAGE FEATURES (cont'd)

	GLYPNIR	CFD	IVTRAN
Data types -			
real	32 & 64 bit	32 & 64 bit	32 & 64 bit
integer	32 & 64 bit	32 & 64 bit	32 bit
alphanumeric	yes	no	no
logical	yes	yes	yes
complex	no	no	no
string	yes	no	yes
Implicit declaration	no	FORTRAN conventions	FORTRAN conventions
Allowable bases	betw'n 2 & 36	base 10 only	8, 10, & 16
Data initialization	yes	some	yes
Subscripted variables	1 subscript	3 subscripts	unlimited subscripts
Dynamic storage allocation	yes	yes	no
Data alignment -			
by row	PE variables	(*) construct	physical skewing
by column	PE vectors	vectors	aligned index
equivalencing	AS construct	EQUIVALENCE	EQUIVALENCE, OVERLAY & DEFINE
Common/global data areas	Limited no. of global areas	COMMON	COMMON
Mixed mode arithmetic	yes	no	yes
Partial word operations	yes	no	no
Relational operators	complete	complete	complete
Logical operators	complete	AND, NØT, ØR	complete
I/Ø to I4DM disk -			
full row	yes	yes	yes
half row	yes	yes	no
quarter row	yes	yes	no
Formatted display	no	no	yes
Asynchronous I/Ø	yes	yes	yes
I/Ø to Tenex disk	yes	no	no

3. Available compiler-provided functions, including logical, mathematical, and bit manipulative functions.
4. Compiler ability to utilize temporary storage, including dynamic array allocation.
5. Ability to allocate data across PE memories and within the memory of one PE.
6. Adequacy of debugging facilities, including checks on bounds of array subscripts, temporary debug output, and subroutine call tracebacks.
7. Compiler efficiency and code optimization.
8. Code checking facilities (simulators) external to the ILLIAC.
9. Appearance and readability of code syntax.
10. The level of support the language will receive, including error correction and future enhancements.

The remaining features listed in Table 2.2 refer to specific technical items important to the implementation of any major application package on the ILLIAC. The following section presents a summary comparison of all three languages.

Language Comparison and Selection Criteria

In order to make the final selection, as will be done later in Chapter IV, this section presents a summary comparison of the four available application languages: ASK, GLYPNIR, CFD, and IVTRAN. The basic merits of these languages are compared to determine their relative suitability for use in the implementation of large application software systems on the ILLIAC.

The first and foremost criteria for selecting any ILLIAC language is its ability to utilize the parallel processing features of the ILLIAC Array. The language best suited for this purpose is ASK, the assembly language for the ILLIAC. Its drawback, of course, is the same difficulty inherent in using any assembly language. However difficult that may be, ASK must be used selectively to optimize both the inner computational loops and I/O processing.

Of the three higher level languages, GLYPNIR offers the greatest flexibility to the programmer in accessing the unique hardware features of the ILLIAC. CFD also provides ample access to these features but is constrained by limitations inherent in its FORTRAN-like constructs. Both, however, allow for direct insertion of ASK code for optimal efficiency. IVTRAN provides only for minimal utilization of the parallel processing features. Its major advantage is easy application to program conversion of existing code. If IVTRAN is used, together with selected subroutines coded in ASK, the overall conversion of an existing FORTRAN code can be performed efficiently, and effective utilization of the ILLIAC can be achieved.

The second major criteria for selecting a language is its I/O capability. The two basic types of I/O required are binary and formatted. Only IVTRAN provides formatted I/O comparable with standard FORTRAN. GLYPNIR provides very limited formatting and CFD provides none. Again, ASK may be used to provide this capability for both GLYPNIR and CFD. GLYPNIR, CFD, and ASK provide the best binary I/O capabilities. IVTRAN is limited to buffered binary I/O with no asynchronous capability.

The capacity to utilize the asynchronous I/O feature of the ILLIAC will be extremely important to avoid potentially severe I/O bound operations. Though GLYPNIR and CFD do provide for this capability, any large applications system should include a general purpose I/O package to manage these activities. This package should be written in ASK to best utilize the sophisticated I/O features of the ILLIAC.

A third criteria for selecting a language is the support it will be given in the future. Only GLYPNIR is not actively being maintained. CFD and IVTRAN will both be provided with ongoing maintenance for both error correction and improvements by their respective developers. ASK, of course, will be actively maintained, as it is the basic assembly language for the ILLIAC. CFD and GLYPNIR code can be tested outside the ILLIAC environment as an aid to debugging. This is important for the near future while the ILLIAC is in the checkout phase and turn-around remains a problem. IVTRAN is provided with a "paralyzer" to assist in the conversion of existing FORTRAN code, but checkout can only be performed on the ILLIAC Array itself.

The process of combining relocatable code produced by any of the compilers into an executable program is similar for all the languages. The code produced by each compiler can be processed by the LINKEDITOR provided on the ILLIAC system. What is lacking at present is an adequate overlay facility for managing large application programs. Necessarily, therefore, a manual overlay capability will have to be developed.

In summary, the selection of a language on overall capabilities is largely dependent on the tasks to be performed. For coding a new program for operation on the ILLIAC, most likely CFD or GLYPNIR would be the best choice. Both these languages allow the user the greatest flexibility to access the unique features of the ILLIAC. Thus, optimizing the code for full utilization of the parallel processing is possible. Of these two languages, CFD offers the distinct advantage of a more familiar syntax, and initial checkout of the code can be performed efficiently outside of the ILLIAC environment. This approach would be recommended if only selected NASTRAN modules were to be developed for optimal utilization of the ILLIAC. These selected modules then could be accessed via the ARPANET from any remote host on which NASTRAN is already operational.

If an entire application package such as NASTRAN, which has already been programmed in FORTRAN, were to be implemented on the ILLIAC, then IVTRAN would be the logical choice. The close resemblance of IVTRAN to FORTRAN and the availability of a "paralyzer" to assist in code conversion, minimizes the initial installation task. However, because the resulting code would make inefficient use of the ILLIAC, the more time-consuming modules would have to be recoded to fully utilize parallel processing. The singular advantage of this approach would be that a functioning integrated system could be available early in the process and immediate gains would be realized as each new optimized module is completed.

CHAPTER III

DESIGN CRITERIA

This chapter presents the fundamental design criteria for implementing NASTRAN on the ILLIAC System. Pros and cons of the two alternative approachs for installation are discussed. The technical feasibility of these approaches is related to the basic design items or building blocks of the NASTRAN System. The relationship between the possible installation approaches and installation effort, program efficiency, and user convenience are explored. Finally, a basic design for the installation of NASTRAN on the ILLIAC is selected.

The main purpose for this design would be to provide a user-oriented system, commensurate with current NASTRAN standards, which minimizes the need for user interaction with the ILLIAC IV System. There are two feasible approaches to be considered for installing NASTRAN on the ILLIAC. These are:

1. The "ILLIAC-Only" approach wherein 100% of the NASTRAN program is implemented directly on the ILLIAC.

2. The "ILLIAC + Host" approach wherein selected NASTRAN operations are implemented on the ILLIAC so as to be accessible via the ARPANET from an extended version of standard NASTRAN operating on a remote host computer.

The first approach would involve taking the machine-independent code of current NASTRAN and processing it by a FORTRAN compiler, translator, or other source conversion program. Thus, installation of NASTRAN could proceed in a "normal" manner even though the ILLIAC is not a "normal" machine. However, object code as initially generated by this method would probably not make efficient use of the ILLIAC hardware. As noted in Chapter II, this code still would execute on current third generation computers. Also, once installed and operational, the full power of the ILLIAC could be exploited by selectively optimizing the resulting source code to take full advantage of the unique parallel processing features of the ILLIAC.

The ILLIAC + Host approach is not so much the installation of NASTRAN on the ILLIAC as it is the extension of current NASTRAN to access the ILLIAC. In this approach NASTRAN could be installed on one or more of the conventional host computers (IBM, CDC, or UNIVAC) on the ARPANET. Functional modules which perform heavy computational chores would be specially coded and installed on the ILLIAC. Thus, the input card processing, geometry processing, output file processing, etc. would continue to be performed by a conventional NASTRAN computer. The real "number-crunching" portion of an analysis would be performed by the ILLIAC. This approach has the advantage that the ILLIAC programming effort would be concentrated on relatively few modules. The ILLIAC part of NASTRAN could be specifically tailored to fully utilize the power of the ILLIAC. Problems with source code conversions and inefficient utilization of the ILLIAC could be avoided. However, the <u>user</u> will have to expend more effort to make a NASTRAN run. For example, a scenario for solving a problem with this approach might be as follows:

1. Run NASTRAN on a conventional host computer to process input.
2. Log on to the ARPANET system to direct data files from the host to the ILLIAC IV System.
3. Run NASTRAN on the ILLIAC to obtain solution.
4. Log on to the ARPANET system to direct data files from the ILLIAC IV System to the host computer.
5. Restart NASTRAN on the host computer to obtain printed output.

As will be shown later, the end result of both approaches would provide the user with a highly efficient code for executing structural analyses on the ILLIAC. The time phasing of its availability would be different as shown below in Figure 3.1. The solid line shows the sequence in which full optimization is achieved as each increment of capability is added. Note that for the ILLIAC-Only approach, the process of optimization accelerates in time while for the ILLIAC + Host, the time for implementation of each new capability remains relatively constant. This difference reflects the fact that with the ILLIAC-Only approach, the code to be implemented with optimization is derived from existing tested capability and needs only to be modified, as opposed to being developed especially for and tested entirely on the ILLIAC. Table 3.1 has been prepared to help identify those modules which would be prime candidates for optimization. The heavy computational modules shown are the prime candidates for recoding under either approach. Conversely, those modules which might be modified frequently for future developments would not be prime candidates. Those modules with both characteristics would require the exercise of engineering and programming judgment before a decision could be made.

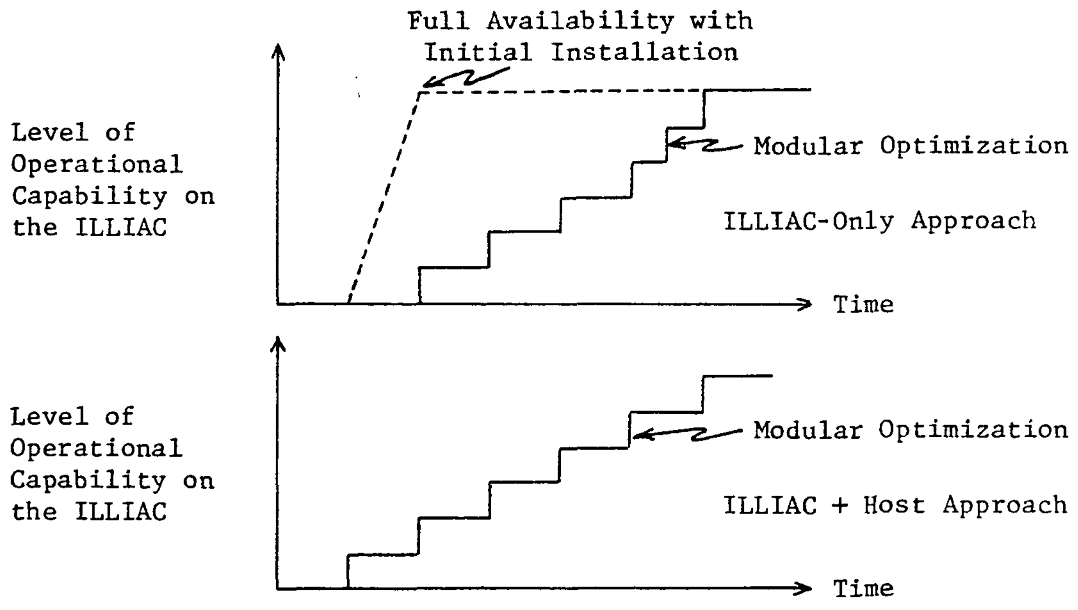

FIGURE 3.1. TIME PHASING OF AVAILABLE NASTRAN CAPABILITY ON THE ILLIAC.

TABLE 3.1. NASTRAN MODULE CHARACTERISTICS

MODULE	C	M	MODULE	C	M	MODULE	C	M	MODULE	C	M
ADD	X		GPWG			PLA2	X		SMA3		
ADD5	X		IFP		X	PLA3	X	X	SMP1	X	
BMG			IFP1		X	PLA4	X	X	SMP2	X	
CASE			IFP3			PLØT		X	SMPYAD	X	
CEAD	X		IFP4			PLTSET		X	SØLVE	X	
CHKPNT			IFP5			PLTTRAN			SSG1		X
CØND			INPUT			PRTMSG			SSG2		
DDR1			INPUTT1			PRTPARM			SSG3	X	
DDR2			INPUTT2			PURGE			SSG4		
DECØMP	X		JUMP			RANDØM			TA1		X
DPD			MATGPR			RBMG1			TABPRT		
DSMG1	X	X	MATPRN			RBMG2	X		TABPT		
DSMG2			MATPRT			RBMG3			TRD	X	
END			MCE1			RBMG4			TRNSP	X	
EQUIV			MCE2	X		READ	X		UMERGE	X	
EXIT			MERGE	X		REPT			UMFEDIT		
FBS	X		MPYAD	X		SAVE			UPARTN	X	
FRRD	X		MTRXIN			SCE1			VDR		X
GKAD	X		ØFP		X	SDR1			VEC		
GKAM			ØUTPUT1			SDR2			XCSA		X
GP1			ØUTPUT2			SDR3			XGPI		X
GP2			ØUTPUT3			SEEMAT			XSFA		
GP3		X	PARAM			SETVAL			XSØRT		
GP4			PARTN	X		SMA1	X	X	XYPLØT		
GPSP			PLA1	X	X	SMA2	X	X	XYTRAN		

C = Module performs much computation
M = Module may be modified frequently

The main focus of this chapter, therefore, will be to identify the criteria for selecting the best of these two feasible approaches to installing NASTRAN capability on the ILLIAC. The relative efficiency and estimated manpower cost for implementation of both approaches is developed and summarized in Table 3.2. Man-time estimates are developed from the criteria presented for each of the design items considered. The results of this comparison show that:

1. The manpower cost for implementing the ILLIAC-Only approach is estimated to be from 25% to 30% less than that of the ILLIAC + Host approach.
2. The optimized capability for the ILLIAC-Only approach is estimated to be within 10% of that of the ILLIAC + Host approach.

In addition to these two selection criteria of cost and efficiency, a third criteria is also established giving consideration to user convenience. Noting that the ILLIAC-Only approach provides for an integrated system, directly by user from his office over ARPANET, that is entirely operational on the ILLIAC. The multiple step processing sequence required for the ILLIAC + Host approach is avoided. This observation, together with the estimates for lower cost of installation, and relatively minor differences in efficiency, indicates that the ILLIAC-Only approach must be selected as the best of the two alternatives.

The discussion of each design item used to establish the cost and efficiency estimates summarized in Table 3.2 are presented below. These items are subdivided into four major sections as follows:

1. Systems and Maintenance
2. The Executive
3. Utilities
4. Engineering Computations

The final section of this chapter presents an overview of those three selection criteria. Examples are also given to show that NASTRAN, using the parallel processing capability of the ILLIAC, could provide the user with an effective gain in efficiency from 10 to 500 times that of current third generation serial processors.

TABLE 3.2. NASTRAN INSTALLATION ESTIMATE FOR THE ILLIAC IV

DESIGN ITEMS	ILLIAC-ONLY				ILLIAC + HOST	
	Translated		Optimized			
	Effort	Efficiency	Effort	Efficiency	Effort	Efficiency
I. SYSTEMS AND MAINTENANCE						
Compiler Development	4		4		0	
Required Code Modifications	3		3		20	
Linkage Editor and Overlays	0		0		0	
Operating System Interfaces	0		0		0	
Maintenance	10		20		23	
Implementation of Future Capabilities	0		0		0	
II. EXECUTIVE						
Operations Sequence Control	4		6		20	
File Allocation	1		4		4	
Checkpoint/Restart	0	10	5	20	5	20
III. UTILITIES						
I/Ø Routines	7	5	10	10	10	10
String Notation and Matrix Packing	5	5	7	20	7	20
Open Core	0		0		0	
Miscellaneous Utilities	1		1		1	
IV. ENGINEERING COMPUTATIONS						
Matrix Operations	15	15	30-50	25	30-50	25
Functional Modules	10	2	20-30	15	20-40	20
TOTALS	60	37	110-140	90	140-180	95

Values for "effort" are expressed in man-months.
Values for "efficiency" are computing speed ratios for the ILLIAC to an IBM 370/165.

SYSTEMS AND MAINTENANCE

Compiler Development

This design item consists of the development or modification of FORTRAN source conversion programs and/or the modification of an existing ILLIAC compiler.

For the ILLIAC + Host approach, no effort is required. ILLIAC code would be developed from scratch. Therefore, existing ILLIAC compilers can be used.

For the ILLIAC-Only approach, two options are available. Either a source conversion program can be written to convert NASTRAN's FORTRAN code to an ILLIAC language, or the "paralyzer", which converts FORTRAN to IVTRAN, can be used with minor modifications. The latter case is used in estimating the level of effort for this design item in Table 3.1.

Required Code Modifications

Required code modifications for the ILLIAC-Only approach consist of adding another allowable machine type to the machine-independent code. In particular, this consists of modifications to subroutine BTSTRP plus all routines which reference the "machine-type" parameter in the /SYSTEM/ common block. It also includes checking all routines which use machine-dependent information such as number-of-bits-per-word, number-of-bits-per-character, etc.

For the ILLIAC + Host approach, this item consists of developing an entire Executive System for NASTRAN to execute on the ILLIAC. This Executive would duplicate most of the functions of the standard NASTRAN Executive. It would control the sequence of module execution, allocation of data blocks to the modules, checkpoint/restart, etc.

Linkage Editor and Overlays

This item consists of the process of obtaining ILLIAC IV machine code from the object code produced by a compiler. The present linkage editor for the ILLIAC has no overlay capability. However, a non-automatic overlay may be implemented by the ILLIAC User Support Group of the IAC if there is sufficient demand. In order to use this feature, a source program would have to make a call to the executive loader in order to load the required program segment before calling any subroutines in that segment.

For the ILLIAC-Only approach, the link drivers XSEM1 to XSEM14 would be replaced with machine-dependent code. This code would manually load the program segment into core for each module to be executed.

For the ILLIAC + Host approach, a similar scheme would be used to load each program segment. This scheme would be incorporated into a yet-to-be-designed Executive System for the portion of NASTRAN to be installed on the ILLIAC.

Operating System Interfaces

This item consists of designing procedures to effectively utilize the job control language, dynamic runstream modification, asynchronous input/output, and data transfers between secondary storage devices, etc. These interfaces are simple and well-defined because the ILLIAC is not a multiprogramming computer. That is, it can execute only one user program at a time. The major system interface, such as the equivalent of the GINØ routines, are treated as separate design items. Therefore, for either installation approach, the effort required for this design will be small.

Maintenance

The maintenance design item includes the normal maintenance functions for NASTRAN. These involve error correction, documentation, updating, and incorporation of new developments into the system, etc. For purposes of assigning a value for the level of effort for this item, consideration has been given not only to the actual installation, but also to future maintenance.

During installation, the major effort for this item will be documentation. Naturally, the amount of documentation required varies with each installation approach. For the ILLIAC-Only approach, the source conversion program would be documented, as would all subroutines which were recoded. Also, a section in Chapter V, Operating System Interfaces, in the NASTRAN Programmer's Manual would have to be added to cover the ILLIAC installation.

For the ILLIAC + Host approach, all new modules and subroutines for the ILLIAC would be documented. Additional documentation also would be required for the ILLIAC Executive System and the ILLIAC: Host interface. As in the ILLIAC-Only approach, a section in Chapter 5 of the Programmer's Manual would be added.

After installation, the major effort for this item would be error correction, updating, and incorporation of new developments. For the ILLIAC-Only approach, this effort would be minimal. However, for the ILLIAC + Host approach, this effort could be considerable because of the specialized code involved.

Implementation of Future Capabilities

The implementation of future capabilities developed in the FORTRAN version of NASTRAN would be easiest for the ILLIAC-Only approach. For the ILLIAC + Host approach, considerable effort would be required because the new capability must be completely recoded. Of course, the original FORTRAN version would be available to the user on the remote host computer during that development period.

EXECUTIVE

Operations Sequence Control

This design item is concerned with controlling the execution sequence of the functional modules. For the ILLIAC-Only approach, little effort would be required because existing code can be used. However, for the ILLIAC + Host approach, new routines will be needed to account for the special purpose design of and unique interfacing between modules.

File Allocation

This item is concerned with the allocation of data blocks for the functional modules and with formatting the ILLIAC disk memory (I4DM) for lowest possible access times.

For the ILLIAC-Only installation approach, file allocation would be performed by the standard NASTRAN routines GNFIAT, GNFIST, and XSFA. GNFIAT is machine-dependent and it would therefore be recoded for the ILLIAC. In this implementation, GNFIAT would have the task of specifying initial I4DM formats for the NASTRAN data blocks. During execution of the functional modules, these formats would be extended by GINØ. Thus, GNFIAT would specify the starting disk addresses for data blocks and GINØ would automatically obtain extents on I4DM as needed. Permanent files such as NPTP, ØPTP, PLT2, and so on would be allocated by the user with ACL, the ILLIAC System Control Language.

For the ILLIAC + Host approach, file allocation would most likely be handled dynamically by the module's own I/Ø routines. Thus, the user would use ACL to allocate his permanent files for input/output and checkpointing, plus a large scratch area on the I4DM. This scratch area would be managed during execution by the I/Ø routines.

Checkpoint/Restart

For the ILLIAC-Only approach, checkpoint/restart can be performed simply by allocating NPTP and ØPTP to the Central File System instead of the ILLIAC disk. Thus, when an ILLIAC run terminates, the checkpoint file would already be stored on permanent storage. A more efficient method, however, would be to "mark" the areas on the ILLIAC disk which have been "checkpointed". These areas would be copied to the Central File System either when execution terminates or when more disk space is needed for other data blocks. With this method, module XCHK would have to be modified.

The same method of marking checkpointed ILLIAC disk areas also would be used in the ILLIAC + Host approach. A new checkpoint module would be developed for this purpose.

UTILITIES

I/Ø Routines

This item encompasses all routines from the highest level GINØ entry points (ØPEN, CLØSE, READ, WRITE, etc.) to the actual I/Ø requests. All of these routines are machine-dependent. For either installation approach, these routines would be coded in ASK, the Assembler Language for the ILLIAC. For the ILLIAC + Host approach, these I/Ø routines might differ functionally from the GINØ routines. However, development effort would be about the same for either installation approach.

String Notation and Matrix Packing

This design item is concerned with packing and unpacking of matrix data and the transfer of matrix data between ILLIAC Array Memory and Disk Memory. For the ILLIAC-Only installation approach, ASK subroutines would be developed to perform pack/unpack operations. This corresponds to the way in which these routines are currently implemented on the NASTRAN computers using machine-dependent assembler language. For the ILLIAC + Host approach, functionally similar subroutines would be implemented. Since entirely new modules would be developed for this approach, new schemes for matrix packing and storage would also be developed. The level of effort needed to design and implement these new schemes is judged to be about the same as to recode the matrix packing routines currently in NASTRAN for release with Level 16.

Open Core

The implementation of open core on the ILLIAC is quite easy for either of the installation approaches. Since ILLIAC Array Memory is of fixed length and the ILLIAC does not support multiprogramming, the KØR5Z function need only compute the number of words of core between the starting absolute address of open core and the highest possible absolute address.

Miscellaneous Utilities

Miscellaneous utilities include various machine-dependent routines for performing link switching, bit manipulation, sampling CPU and elapsed time, writing console messages, generating plot files, etc. Their implementation for either installation approach will be a relatively minor task.

ENGINEERING COMPUTATIONS

Matrix Operations

This design item is concerned with the installation of the subroutines and modules which perform matrix multiplication, addition, decomposition, eigen-value extraction, etc.

For the ILLIAC-Only approach, all of the machine-independent matrix routines would first be installed "as is". The machine-dependent routines would be recoded in ASK. The code installed in this initial implementation would generally perform inefficiently. Therefore, the matrix subroutines would be recoded one by one. Of course, while this recoding is being done, the original versions will still be available for performing analyses. The priority schedule used for recoding these routines and modules would be based on:

1. The amount of computational time used by the routines.
2. Frequency of occurance in each Rigid Format.
3. Frequency of use of each Rigid Format.

The programming effort required to recode all of these routines would naturally be high, though certainly not more than would be required with the ILLIAC + Host approach.

In the ILLIAC + Host approach, the routines which perform matrix operations on the ILLIAC would be redesigned from scratch. This will result in the most efficient code, and it will require a substantial development effort.

Functional Modules

This design item includes the installation of all the NASTRAN functional modules except those concerned with the matrix operations discussed above. For the ILLIAC-Only installation approach, this installation can be straight-forward. The modules would simply be recompiled. The sheer number of subroutines involved makes this item a substantial effort. Recoding selected modules to increase computing speed and efficiency naturally would require additional effort.

For the ILLIAC + Host approach, NASTRAN functional modules might not be recoded and installed on the ILLIAC on a one-for-one basis. It is likely that only logical groups of modules (e.g., element matrix generation, assembly, and solution) would be installed on the ILLIAC. The level of effort required here would depend on the number of module sets to be installed on the ILLIAC. The effort, of course, would be even greater than that required to modify the initial code installed under the ILLIAC-Only approach.

SELECTION CRITERIA

This section presents the three basic criteria for selecting a basic design approach to be used for installing NASTRAN on the ILLIAC IV. The two primary options studied are shown in the decision tree presented in Figure 3.1. Three major items are considered in arriving at the final selection. These are the anticipated level of effort for installation, the efficiency of execution, and user convenience. Each of these items is discussed below.

Level of Effort

As can be seen from Table 3.2 presented at the beginning of this chapter, the ILLIAC + Host approach is estimated to require about 25% more effort than the ILLIAC-Only approach. Based on this one parameter alone, the ILLIAC-Only approach would be preferred. Also, note that, after expending only about one third of the total effort for the ILLIAC-Only approach, a full NASTRAN System would be operational on the ILLIAC. Once installed, comparative performance studies can be made to define a priorities checklist of candidate modules to be optimized in order to achieve the full benefits of parallel processing.

Efficiency

The overriding question related to efficiency must first be directed to the issue of whether or not it is at all worthwhile to install NASTRAN on the ILLIAC. Only after this question is answered, is it appropriate to examine the efficiency of alternative approaches.

The potential efficiency of NASTRAN operations performed on the ILLIAC compared to both the IBM 370/165 and the CDC 6600 computers is summarized in Table 3.3. For this comparison, the process of matrix decomposition was selected as a representative operation involving large amounts of both computation and input/output processing. Also included in this table are the run time estimates for matrix decomposition using the initial implementation of the algorithm under the ILLIAC-Only approach prior to its optimization.

Table 3.4 shows the timing constants used for these comparisons. The timing constants for the ILLIAC were computed based on the estimated number of ASK instructions generated before recoding and the actual number of ASK instructions used in the optimized code of the ILSA program being developed by UAI for DNA. The constants P and I, representing packing and I/Ø operations respectively, include the actual time to read/write on secondary storage as well as the CPU time incurred. The total times shown in Table 3.3 are computed using equations 9 through 13 in Section 2.2.1 of the NASTRAN Theoretical Manual. For the decomposition without spill, the extremely high speed of the ILLIAC I4DM disk compensates for the inefficient use of the CPU as that total execution is greatly reduced.

Using this typical example, performance improvement ratios of from 25 to 130 could be achieved on the ILLIAC when compared to current third generation serial processors. Two considerations must be given to evaluating these very significant improvement potentials indicated in Table 3.5. The first is that not all modules can be recoded to achieve this marked an improvement. Such modules include table and matrix assemblers as well as output generators.

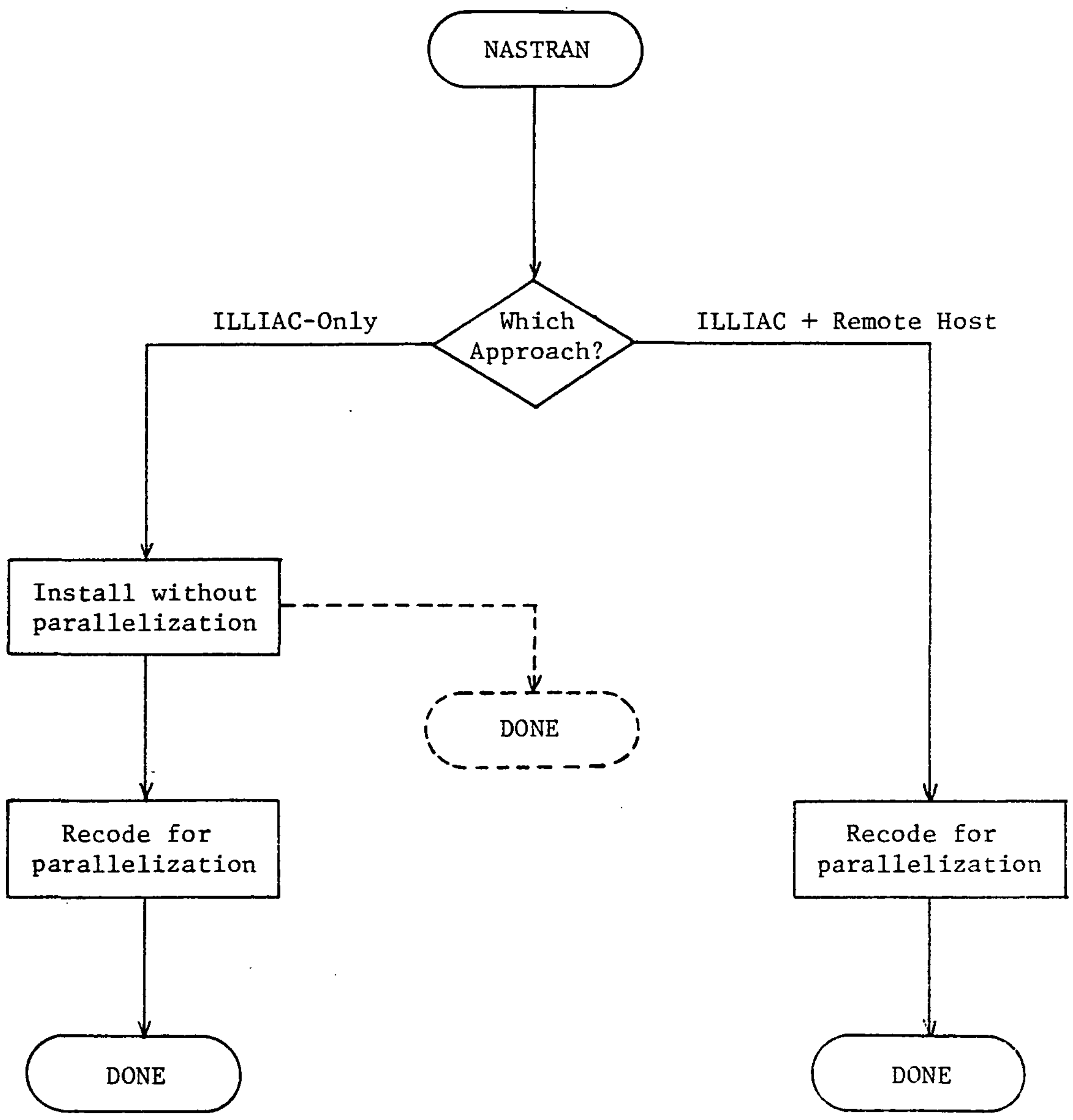
NASTRAN
Which Approach?
ILLIAC-Only
ILLIAC + Remote Host
Install without parallelization
DONE
Recode for parallelization
DONE
Recode for parallelization
DONE

FIGURE 3.2. INSTALLATION APPROACH DECISION TREE

TABLE 3.3. TIMING COMPARISON

	IBM 370/165	CDC 6600	ILLIAC IV (before recoding)	ILLIAC IV (after recoding)
Decomposition (no spill) Matrix order 10000 dof Semibandwidth 300 dof Active columns 50 dof	0.6 hours	1.3 hours	0.7 hours	0.01 hours
Decomposition (with spill) Matrix order 10000 dof Semibandwidth 600 dof Active columns 50 dof Working storage 50000 words	100 hours	150 hours	9 hours	4 hours

TABLE 3.4. TIMING CONSTANTS (MICROSECONDS)

Coefficient	IBM 370/165	CDC 6600	ILLIAC IV (before recoding)	ILLIAC IV (after recoding)
M_B	2	5	3	0.05
M_C	5	12	7	0.1
I	240	340	22	10
P	120	170	11	5

Notes: 1. M_B = time to process one term inside the band

M_C = time to process one active column term

I = time to store and retrieve one term of intermediate results on a secondary storage device

P = time to store one term of final results on a secondary storage device

2. Constants for IBM 370/165 and CDC 6600 were obtained from the MSC/NASTRAN Applications Manual, Section 7.3.

3. Constants for ILLIAC were calculated from instruction timing data for a sample assembler language program.

4. I and P include the real time for the data transfer between core and secondary storage.

5. I is estimated, in all cases, to be twice P.

TABLE 3.5. PERFORMANCE IMPROVEMENT RATIOS

	IBM 360/165 to ILLIAC IV	CDC 6600 to ILLIAC IV
Decomposition (no spill)	60	130
Decomposition (with spill)	25	37.5

However, the raw power achieved by the basic speed of the ILLIAC will still yield marked improvement in the execution times for these modules if only modest parallelization is achieved.

Second, the hardware constraints discussed at the conclusion of Chapter II of this study indicate that limitations may be imposed on the size of problems that can be efficiently solved on the ILLIAC. These constraints arise from the limited storage available on the I4DM and the slow transfer rates to the I4DM from central memory. The latter constraint could severely increase the overall execution times for certain large problems requiring run-time spillage from the I4DM to central memory. However, even with this constraint, and noting that similar sized problems would severely tax the largest of modern serial processors, the ILLIAC should prove to be a most viable alternative computing resource.

Having shown the dramatic efficiency gains to be realized by installing NASTRAN on the ILLIAC the question of which approach to use for installation can now be asked. Referring again to Table 3.2, the completed installation of the ILLIAC-Only approach would only be about 10% slower than the ILLIAC + Host approach. Of course, the initial installation of NASTRAN with the ILLIAC-Only approach would suffer significant inefficiencies at first. As selected modules are upgraded, the effects of their improved operating efficiency would be immediately apparent.

By comparison, the ILLIAC + Host approach offers the singular advantage that, as each new set of capabilities is installed, the full effectiveness of the ILLIAC for those operations would be available and could be accessed from the remote host computer. If only selected NASTRAN capabilities were to be implemented on the ILLIAC, this approach would be preferable. Otherwise, the relative speed of execution with either approach is not a major deciding factor.

Efficiency, however, also includes factors other than speed of execution. Total turn-around time from input to output is a critical issue to the eventual user of the system. This design item is discussed next.

User Convenience

The fewer steps and the less work a user must perform to complete a NASTRAN execution on the ILLIAC, the more cost effective and convenient it is for him to use NASTRAN on the ILLIAC. In the ILLIAC-Only approach, the user has only one batch job to run in order to get a completed NASTRAN execution. The ILLIAC + Host approach requires three separate execution steps: (1) NASTRAN on host, (2) ILLIAC, and (3) NASTRAN on host. Therefore, considering user convenience and the ever present potential for system failures at each step of a multistep sequence, the ILLIAC-Only approach is preferred.

Conclusions

The following three conclusions may be drawn from the discussions of design and selection criteria presented above:

1. Installation of NASTRAN on the ILLIAC IV is feasible and practicable.

2. Installation of NASTRAN on the ILLIAC is justified on the basis of dramatically increased execution efficiency derivable from parallel processing - e.g., the ILLIAC could execute 25 to 130 times faster than current serial computers.

3. The preferred approach is the "ILLIAC-Only" approach to full NASTRAN installation on the ILLIAC with subsequent optimization of selected modules to achieve the full potential of parallel processing.

Based on these conclusions, which are derived from consideration of cost effectiveness, overall efficiency, and user convenience, the overall systems design can now be developed for implementation of NASTRAN on the ILLIAC IV. The specifications for this implementation are outlined in the following chapter.

CHAPTER IV

DESIGN SPECIFICATIONS

This chapter presents the design specifications for implementing NASTRAN on the ILLIAC. The bases for selecting an approach were discussed in Chapters II and III. The specifications presented here, therefore, reflect the considerations given in those two chapters related to the ILLIAC IV System hardware and its supporting software as well as to the NASTRAN program itself.

Both alternatives studied, the "ILLIAC-Only" approach and the "ILLIAC + Host" approach, were determined to be not only desirable, but feasible and practical as well. The ILLIAC is now operational and has been demonstrated successfully as a tool for executing application codes similar to NASTRAN. Because of its availability, implementation of NASTRAN could start immediately.

Both alternative approaches would provide for optimum utilization of the ILLIAC. Execution of NASTRAN on the ILLIAC was shown to offer a very significant increase in efficiency over current third generation serial processors. And, the resulting capability would be accessible via the ARPANET from most major industrial centers in the United States. The ILLIAC-Only approach, however, offers certain advantages over the alternative ILLIAC + Host. These are:

1. The user executes his entire analysis only on the ILLIAC. He may utilize <u>any</u> host computer on the ARPANET to assemble his input data card file and to receive listings and/or plots of his output.

2. Execution turn-around will be faster and less subject to system problems by avoiding the execution of segments of his analysis on a remote host computer.

3. The cost of implementation is estimated to be lower and the estimated cost of subsequent development and maintenance would also be lower.

As summarized in the final section of Chapter III, the ILLIAC-Only approach has been selected for implementing NASTRAN on the ILLIAC. The sequence of major events to accomplish this implementation would be as follows:

1. Adapt the existing IVTRAN compiler and its accompanying paralyzer as a preprocessor for initial optimization of existing FORTRAN code.

2. Develop an general purpose I/O package to provide efficient data management functions similar to current NASTRAN GINØ.

3. Develop a specialized Executive Sequence Monitor to control data file transfers, checkpoint/restart, program overlay, and execution of functional modules.

4. Develop an optimized set of utility routines for use by the functional modules to perform matrix packing and unpacking, bit manipulation, core size computations, and other necessary functions.

5. Develop optimized code to replace the current machine-dependent code of NASTRAN.

6. Install the machine-independent code using the tools developed for translating FORTRAN code into IVTRAN code for the ILLIAC.

7. Document and demonstrate the initial installation of NASTRAN and measure the relative performance of the functional modules to establish priorities for Step 8.

8. Optimize, document, and demonstrate the functional modules selected according to priorities established in Step 2.

The specifications presented below to accomplish these steps are subdivided into five major sections. The first section outlines the overall organization of the program and the environment in which it will be installed. The second section outlines the specifications for the system maintenance and development utilities required. The last three sections define the specifications for developing the Executive Sequence Monitor, for optimizing the utility functions, and finally for installing the modules which perform engineering computations. Detail specifications for subsequent optimization of the computational modules are not presented here. These specification would involve details of the individual computational algorithms being optimized and therefore are beyond the scope of this study. They are, however, proper province of the design effort to be undertaken at the conclusion of Step 7 mentioned above. The general requirements for these optimization tasks are presented as part of the next section.

PROGRAM ORGANIZATION

This section outlines the overall design philosophy and ILLIAC as shown in Figure 4.1. The NASTRAN program is hierarchically divided into two levels. The highest level performs the executive functions to be controlled by the new NASTRAN Executive Sequence Monitor. This executive system provides a set of operations that are independent of the problem to be solved. The actual problem solution is provided by a lower level set of functional modules controlled by the Executive Sequence Monitor. Each module is independent of all other modules in the sense that the modification or addition of a module would require only table updates in the Executive Sequence Monitor and updates to only those other modules affected by changes (if any) in the communication data blocks. Communication between modules is only allowed through auxiliary files and a parameter table maintained by the Executive Sequence Monitor.

The Executive Sequence Monitor is also charged with other functional responsibilities usually performed by the resident operating system on conventional third generation computers. These include:

1. The loading and overlay of code for exeuction on the ILLIAC.

2. The generation of file control commands (in ACL) to monitor and define checkpoint/restart operations involving data transfers between the Central System and the I4DM, and between the Central System and the UNICON.

3. The physical mapping of data onto the I4DM to efficiently utilize the high transfer rates to and from ILLIAC memory.

The solution of a structural analysis problem is performed by the functional modules. These modules are divided into three categories:

1. Preface Modules - These modules are the first executed in the solution of a problem. They perform two functions:

    a. Process the NASTRAN input deck
    b. Perform general problem initialization

2. Utility Modules - These modules perform functions that are independent of problem solution. These include the basic matrix operations and the user-controlled options such as output processing and reporting.

3. Structurally Oriented Modules - These modules perform all the problem-dependent functions such as matrix generation, matrix assembly, load generation, stress recovery, and plotting.

The order of execution of these modules, which is controlled by the Executive System Monitor, is dependent on the general type of analysis requested or by the sequence of DMAP instructions selected by the user. The independent design of each module is required to facilitate future maintenance and development efforts. These requirements provide for:

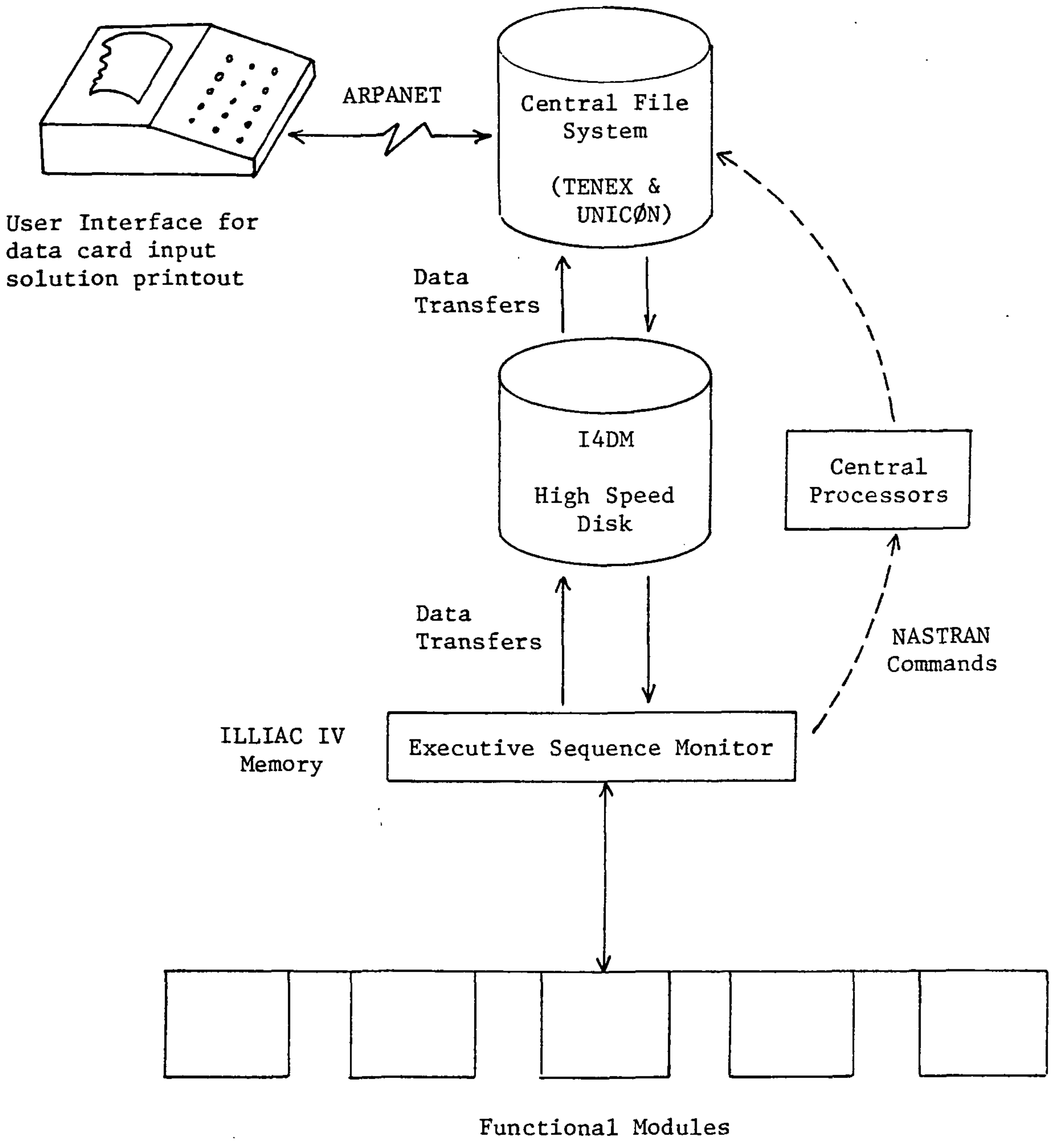

FIGURE 4.1. NASTRAN PROGRAM ORGANIZATION ON THE ILLIAC

1. The ability to modify small sections of relevant code without affecting other modules.

2. Selectively optimizing existing modules to utilize the parallel processing features of the ILLIAC while minimizing impact on other modules.

3. Adding new modules and capabilities with minimal changes to the existing ILLIAC System, especially those already tested in the NASTRAN environment on conventional serial processors.

Unlike conventional application software development efforts, which typically find machine-independence to be a major design specification, this installation of NASTRAN should exploit the special hardware features of the ILLIAC in order to reap the full benefit of its power and efficiency. This philosophy, however, imposes certain limitations on the above requirements for module independence.

As individual segments of NASTRAN's analytical capability are assigned for optimization following the initial installation of NASTRAN on the ILLIAC, detail specifications for each set of associated modules must be prepared. By grouping these modules together, the unique paths of data communication between modules can be identified. With this information, together with the detail knowledge of the processing steps to be programmed, the data storage on, and data transfers to and from the I4DM can be optimized. Careful attention must be given to the design of these special purpose data files. Additional utilities may have to be developed to pre- or post-process these files in order to maintain the integrity of all communications between these and the previously existing modules of NASTRAN. The high transfer rates between I4DM and the ILLIAC memory justifies the special formatting of data blocks which may be read and processed repeatedly, as would be the case with any iteration analysis procedure. Also, specialized formats could be used to minimize routine of data between individual PE memories. In summary, therefore, the basic design optimization requirements are:

1. Selectively identify sets of analytically related modules for optimization as a group.

2. Identify unique data communication paths between analytically associated modules.

3. Design the formats for these data to allow optimization of computational algorithms with special attention to efficiency of I/O operations between I4DM and ILLIAC memory.

4. Include in any such specifications the necessary utilities to pre- or post-process those data files used to maintain generality of communication links between the set of module to be optimized and the other modules of NASTRAN.

In order to meet these overall program design objectives, certain non-NASTRAN capabilities must be made available. These include the maintenance utilities for optimizing existing FORTRAN codes, for compilations, and for linkage editing. Specifications for these utilities are given in the next section.

SYSTEM INTERFACE REQUIREMENTS

This section describes the components of the system involved in the interface between the user and NASTRAN on the ILLIAC for both problem solving and program development. These components include the hardware facilities and the software necessary for communication and program implementation.

User Interface Requirements

The user interface to NASTRAN operating on the ILLIAC is quite similar to that of NASTRAN operating on third generation machines of today. The main difference results from the necessity of accessing the ILLIAC over the ARPANET. This interface is shown in Figure 4.2.

The host computer is required only as a interface for the user. It performs no computational role in the NASTRAN problem solution. Its main function is to provide the user with card input and printed and plotted output capabilities. The local file storage will also provide for the temporary allocation of user input and output files until final disposition is determined. The unique advantage to this approach is that the host computer need not support NASTRAN itself. It is therefore possible for those users with <u>any</u> host computer on ARPANET, such as a PDP-10, or IBM 360/44 etc., to execute NASTRAN on the ILLIAC.

Once the NASTRAN input, which remains unaltered, is loaded onto the host computer, the user must utilize the services of a FTP processor, Chapter II, to transfer his data over the ARPANET to the ILLIAC complex. This process must also be repeated to transfer results back to the host computer after the execution of NASTRAN on the ILLIAC.

The execution of NASTRAN on the ILLIAC is a straightforward process. Once the desired input files have been transferred to the central file system, the user must utilize the ARPANET to access the ACL system of the ILLIAC complex. He may then submit a batch job to execute NASTRAN on the ILLIAC.

As future capabilities are provided by the ILLIAC complex, the necessity of the user interactively transferring data between the ILLIAC and host may become unnecessary. The transfers would then be made directly from ACL with no direct interface with the FTP processors being necessary.

A sample batch job executed on the ILLIAC with this capability might contain the following ACL statements:

```
CPYNET (INPUT,HØSTID,USERID),        transfer input from Host to ILLIAC
ALLØC  NASTRAN MAP,ID                allocate I4DM space
MØVE INPUT,I4DM:INPUT                transfer input to I4DM
RUN  NASTRAN EXEC                    execute NASTRAN
MØVE I4DM:ØUTPUT,ØUTPUT              transfer output to central file system
DALLØC ID                            free ILLIAC and I4DM
CPYNET ØUTPUT,(ØUTPUT,HØSTID,USERID) transfer output
```

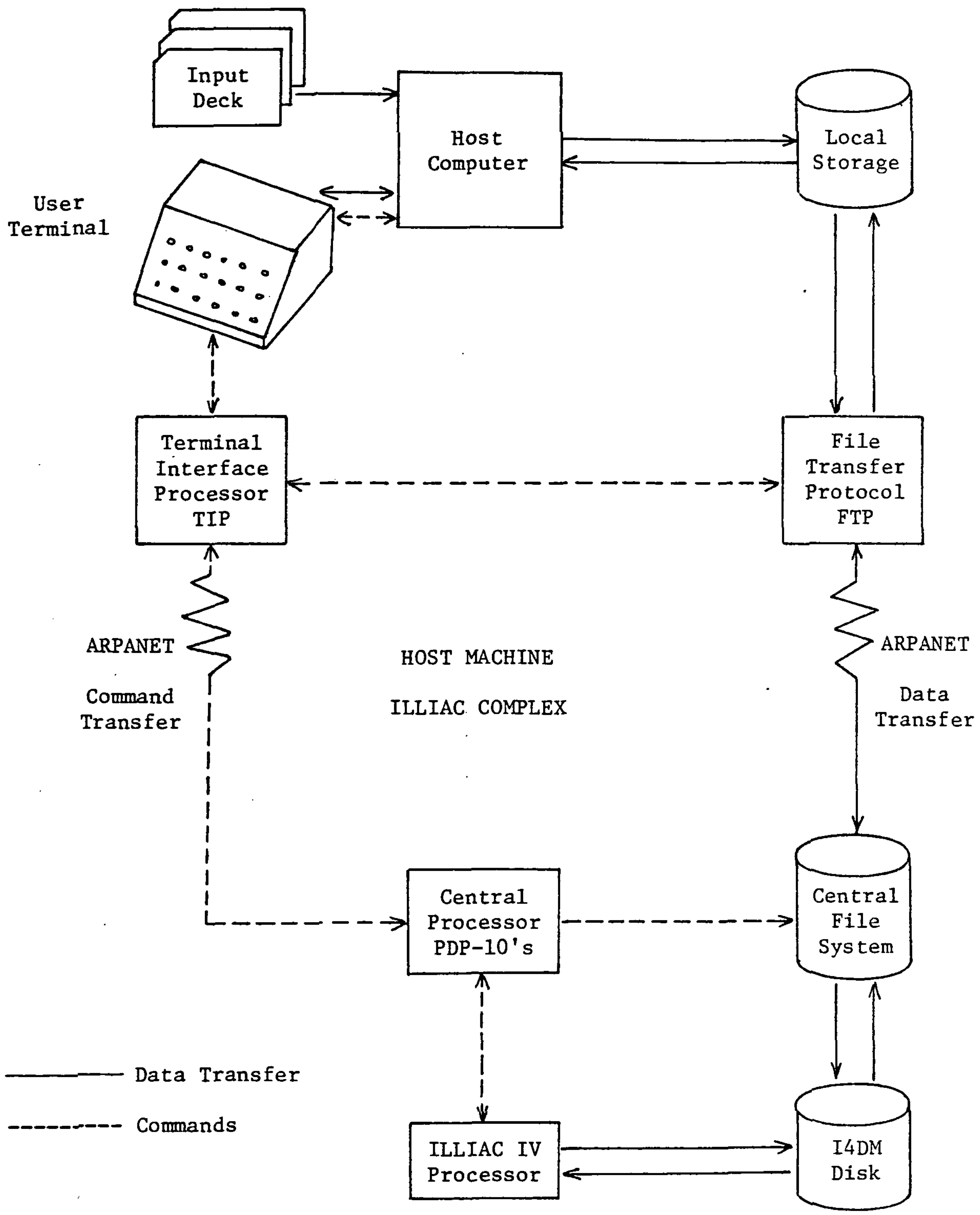

FIGURE 4.2. USER INTERFACE TO ILLIAC NASTRAN

The actual execution of NASTRAN is controlled by the Executive Sequence Monitor (ESM) described later. Once transfer of the files is complete, control is passed to the ESM which takes over, loads the program, allocates scratch space, manages internal data files, and directs the execution sequences requested by the user. Development of the ESM and the functional modules of NASTRAN for implementation requires development of certain maintenance utilities as described next.

Maintenance and Support

One of the major issues in the design of a large software system is the level of effort required for maintenance and future improvements once the initial system is installed. The section discusses how this support is provided for NASTRAN on the ILLIAC IV.

The most important feature of NASTRAN which aids in maintenance is its modular design. As discussed previously, each module is an independent program which is loaded and executed by the executive system. This means that maintenance and modifications to any one module may be done by handling only a small portion of the total code comprising the NASTRAN program. This is an important factor from the standpoint of cost for both computer time and manpower.

The system software available to support NASTRAN on the ILLIAC is discussed in Chapter II. As previously mentioned, the IVTRAN language was chosen because of its similarity with the FORTRAN language operating on most other computers today. The ILLIAC-Only approach selected for installing NASTRAN, which is a program written primarily in FORTRAN, therefore requires the following already existing utilities to be provided and/or modified:

1. A "paralyzer" to initially optimize and translate existing FORTRAN code into the IVTRAN language
2. The IVTRAN compiler
3. Linkage editor

A "paralyzer" for IVTRAN already exists as described in Chapter II above. This utility reads existing FORTRAN code, performs selective optimization, and generates an equivalent IVTRAN code capable of compilation and execution on the ILLIAC. The current capabilities of this "paralyzer" should be extended to include additional features such as:

1. Identify code sequences suitable for specialization to subroutines for optimal coding in ASK.
2. Restructure multiple entry point routines.
3. Develop block structures to facilitate subsequent debugging efforts using structured programming techniques.

Once the "paralyzer" processing has been performed, the resulting IVTRAN code must then be manually checked and updated prior to compilation. Such steps are required to remove obviously inefficient constructs, check the efficiency of alterations performed by the "paralyzer", and plan the development of any required ASK subroutine inserts. The manually updated code would then be ready for compilation.

The IVTRAN compiler itself may require enhancements to provide for special purpose syntax developed from the special purpose optimization design of the "paralyzer". Otherwise, the current compiler should be adequate. Because the ILLIAC is now operational, the usual advantages of an independent simulator checkout of the resulting code are minimal. Verification of the IVTRAN code could proceed directly on the ILLIAC. Also, by the time the NASTRAN code is actually ready for initial installation, additional debugging, testing, and core display facilities should be available as part of the ILLIAC System software.

The third major item required is the Linkage Editor. The currently available ILLIAC control language, ACL, (Chapter II), provides a linkage editing capability which does not yet include overlay features. These overlay requirements must, therefore, be incorporated as part of the Executive Sequence Monitor (ESM) described next.

EXECUTIVE SEQUENCE MONITOR

The Executive Sequence Monitor (ESM) is charged with the full responsibility of directing the execution of NASTRAN on the ILLIAC. Its design will be an adaptation of the current NASTRAN executive system to be driven via the current DMAP control language. The ESM takes over full control from the central processor once the central processor has executed the user ACL commands submitted from his terminal, see Figure 4.2 discussed earlier. These ACL commands direct the central processor to load the ILLIAC disk with both the executable code and the data requested for input by the user. The ESM is then transferred from the I4DM to ILLIAC memory and execution begins. At this point, the ESM takes over. The essential functions of the ESM are:

1. Establish, protect, and communicate values of parameters for each module.
2. Allocate secondary file space.
3. Maintain a restart capability for restarting program execution after either a scheduled or unscheduled interruption.
4. Control execution according to options specified by the user.
5. Load and overlay the program segments required for each subsequent step in the execution sequence.

Checkpoint/Restart

NASTRAN is designed to execute large problem analyses which require large volumes of data and large amounts of computation. Often such analyses must be performed in segments so as to allow the user to examine intermediate output. Even with the best system, a hardware error is not uncommon for long running jobs. Also, data errors, analysis aborts, and I/O errors all contribute to the requirement for a checkpoint/restart capability which must be provided by the ESM.

Unlike conventional NASTRAN as exercised now on serial processors, arbitrary and possibly indiscriminate checkpointing of data files is most undesirable. As noted earlier in Chapter II, the data transfer rates to and from the I4DM and Central Memory is slow compared to the execution power of the ILLIAC. Often, therefore, the cost effective approach would be to recalculate rather than restore. For example, a data file that may take 3 minutes to read onto the I4DM may have been generated in 5 seconds on the ILLIAC. Once execution is initiated, therefore, these 3 minutes to load the I4DM would be idle time for the ILLIAC.

The new NASTRAN checkpoint/restart capability, though similar to the current approach, would checkpoint a data block from the I4DM to Central Memory <u>only</u> under the following two conditions:

1. The available space on the I4DM is exhausted requiring the release of areas which are no longer needed.
2. The desired data to be checkpointed requires a sufficient amount of time to regenerate such that the added expense of a checkpoint operation is cost effective.

Should the ILLIAC System terminate abnormally, the data on the I4DM to be saved would be transferred to the UNICON via Central Memory by the central processors as a normal termination procedure.

Manual Overlay

Because the ILLIAC is a machine dedicated to the execution of user code only, the NASTRAN executive system must therefore assume some of the responsibilities customarily performed by the operating system on a third generation machine. These additional responsibilities include:

1. Controlling the overlay of program segments for each functional module.
2. Issuing control commands (ACL) for all data management operations to and from the central memory and the I4DM as required for checkpoint/ restart and as requested for input and output to each module.

Only the ESM can load and start the execution of a module. The order in which modules are executed is determined by the General Problem Initiator (GPI) which is one of the preface modules of the ESM. It constructs a set of control statements based on input requests which are later used by the executive system to control problem execution.

Intermodule Communications

Communication between modules is only allowed through auxiliary files and a parameter table maintained by the executive system.

The executive system must be open ended in the sense that it can control an unlimited number of modules, data blocks, and parameters. Modification of the executive system necessary for the change, addition, or extension of functional modules is restricted to changing entries in control tables stored within the executive.

Maintenance and Updates

This independence allows modules to be modified individually without the need to manipulate the entire NASTRAN System. The feature is advantageous for several reasons:

1. Maintenance tasks need deal only with a small section of relevant code.
2. Individual modules may easily be modified to utilize the parallel features of the ILLIAC without interferring with existing modules.

UTILITIES

The new NASTRAN system will require development of certain general purpose utility routines to facilitate the exploitation of the parallel processing and asynchronous I/O features of the ILLIAC. These utilities can be subdivided into three basic categories:

1. General purpose I/O routines commensurate with current NASTRAN GINØ.
2. General matrix operation routines to replace MPYAD, SDCØMP, FBS1, MERGE, PARTN, etc.
3. General purpose routines for bit manipulations, matrix packing and unpacking, and sorting, etc.

Certain of these optimized capabilities must be developed first and others may be postponed until later. The requirements for each set are given below.

Input/Output

The general purpose input/output (GINØ) routines are the most important general capabilities requiring optimization at the earliest stages of NASTRAN installation on the ILLIAC. The efficient handling of I/O in the ILLIAC IV single user environment is of great importance. This generalized I/O package must be provided in the new NASTRAN system to handle the bulk of data transfers between PE memory and the I4DM. To do its job efficiently, GINØ will be allocated the entirety of available space on the I4DM. GINØ, therefore, will have complete responsibility for managing that space as required during execution of each module. This package will also provide various levels of support to the functional modules utilizing it.

At the highest level, GINØ functions would be quite similar to those now provided on the three NASTRAN computers today. These functions include the allocation of space on the I4DM and the blocking of records to reduce I/O time. Because of the single user environment of the ILLIAC, it becomes necessary for GINØ to utilize a double buffering technique. This technique allows for asynchronous I/O whereby I/O and computations can be performed simultaneously.

Because of the high data transfer rate available on the I4DM, greater efficiency can be obtained by the careful formatting or mapping of data on the disk. In this manner, the time waiting on disk rotation can be significantly reduced by insuring the required data is mapped properly. This positioning of data on the I4DM is very dependent on the type of computations being performed. Therefore, a generalized package will be able to achieve only a partial optimization of ILLIAC I/O.

A second level of GINØ operation will therefore allow the computation modules to specify to GINØ the type of formatting desired for each data block. This type of information would include data on the expected timing between read and write operations. This type of information allows GINØ to more efficiently allocate disk space for the executing module.

As certain functional modules are rewritten to utilize the unique computational power of the ILLIAC, the idea of a generalized I/O package becomes less desirable. As modules are recoded, a large amount of the required design effort is spent in developing an efficient scheme for storage of data on the I4DM and the dynamic overlapping of I/O and computational operations. Sets of NASTRAN modules that frequently execute in conjunction with each other, e.g., decomposition and forward/backward substitution, could be given maximum freedom to specialize the organization and format of those data files that are used for their intercommunication. On the other hand, a functional module cannot be allowed complete freedom because it has little knowledge of what data is presently stored on the disk and that data must be protected.

The third level of GINØ would provide these modules the desired freedom and, at the same time, retain overall control of the I/O operations. This would be accomplished by allocating a block of disk space to the module for its own use as a scratch area. The module itself would be responsible for all I/O to that block, but GINØ would allow no operations outside that block.

Matrix Operations

The general matrix operation modules of NASTRAN are prime candidates for optimization because of their frequent use in every analysis performed by NASTRAN. The specifications for implementation, however, depend on whether the code from which they are to be developed is:

1. Machine-dependent, e.g., FBS, MERGE, MPYAD, PARTN, etc.
2. Machine-independent, e.g., ADD, DECØMP, TRNSP, etc.

The machine-dependent routines must necessarily be coded especially for the ILLIAC at the outset. Even selected segments of the machine-independent code should be given serious consideration for those are also frequently called from other functional modules. Hence, the efficiency of these utilities would greatly enhance the performance of NASTRAN upon initial installation.

Because these general matrix operations are of such critical importance to the overall performance of NASTRAN, detail specifications for each must be developed before actual implementation can proceed. This is required in order to assure consistency in definition of all operations and data storage schemes to be used. These preliminary design requirements must be developed giving consideration to:

1. Definition of array storage conventions within the ILLIAC PE memories. This includes providing standards for development of the calling routines and for providing options for alternative storage schemes appropriate to the functions being performed in such calling routines.

2. Specialized storage schemes and disk mapping criteria for internal file communications.

3. Specialized storage schemes and disk mapping criteria for external files used for intermodule communication.

This latter requirement is critically controlled by limitations imposed for maintaining generality of data storage conventions in the event any such files may be used by modules other than those for which they were intended. For example, the upper and lower triangular components of a decomposed matrix generated by DECØMP are unlikely to be used by other than the forward/backward substitution module FBS. Therefore, these two matrices should be designed specifically for optimizing these basic matrix operations.

Special attention should be given also to the following detail design considerations in order to achieve full optimization of I/O and computations:

1. Routing - data transfer between PEs should be minimized by:

   a. Designing the output module so as to output the matrix data from the PE into which it later will be used.

   b. Developing schemes whereby routing is performed in parallel so that each move operation is designed to include one word from each PE.

   c. Summing across PE arrays in parallel (a maximum of 6 moves only are required to compute the sum of 64 terms stored one per PE!).

2. Notation - matrix terms can be implicitly identified by bit position in row and column identifiers. Special matrix notation schemes can be designed to save space and for mode control (on/off) of each PE processing that data. Note that the ILLIAC offers excellent bit manipulation software and hardware using the 64-bit word to control the functions of each of the 64 PEs.

3. Mapping - matrix data should be blocked for disk storage so that by mapping the position of each block, effective overlapping of I/O with computation can be achieved using the asynchronous I/O capability of the ILLIAC. Spacing between blocks on disk is dependent on the anticipated computation time for both generation of, and subsequent operations on the data.

Input and output processing is by far the more critical design item. The computational speed of the ILLIAC is so high, it can practically be ignored for matrix operations. Hence, matrix notation schemes are critical to further optimization of NASTRAN on the ILLIAC. To facilitate that optimization, the conventions for standardization customarily used on serial processors must be extended to include the above considerations. The greater flexibility required can be provided via the matrix packing/unpacking and format conversion utilities described next.

Utility Routines

A number of utility routines must be developed to provide the bit manipulation, core size, timing, logic, matrix packing/unpacking, and searching functions required by nearly every NASTRAN module. These operations should be developed directly in ASK.

A second category of utilities are also required. These routines would provide input card processing, message writers, printout control, and debug utilities needed during the development phase. These can be provided directly with the IVTRAN formatted I/O capability.

As noted above, new sets of utilities also must be designed to provide for format identification and for the essential format conversion of matrix data needed for optimized I/O. The primary function of these utilities will be to assist in communication between optimized NASTRAN modules. Their design should be developed as part of the initial installation effort. Both the standard string notation conventions and the special purpose mapping and block formats needed for intermodule communication must be provided. The matrix trailer should be extended to provide this added control information. The calling sequence to be provided must allow for the calling module to specify the format desired and mapping criteria to be used. These new utilities must also adhere to the control requirements imposed by the Executive Sequence Monitor and the general purpose GINØ management facilities described earlier.

ENGINEERING COMPUTATION MODULES

These modules are defined as structurally oriented functional modules. They provide for the following general categories of engineering computations:

1. Preface operations - for input data table assembly
2. Matrix generation - for stiffness, mass, damping, and loads
3. Matrix assembly
4. Matrix solution algorithms - static, dynamic, non-linear
5. Stress recovery
6. Plotting

These modules are generally coded in FORTRAN and can therefore be handled by the "paralyzer" and compiled from the IVTRAN code generated. Before compilation, however, careful examination of the "paralyzer" output is required to validate the syntax generated, to insert ASK subroutines where practicable to provide an initial level of optimization, and finally, to incorporate the new utility routines described above.

Once initial installation and testing is complete, then each module, or set of associated modules, can be examined to establish priorities for further optimization. These priorities should be established on the following criteria as discussed in Chapter II:

1. Need - frequency of utilization expected
2. Initial costs - effort required to implement
3. Cost effectiveness - potential gains to be achieved compared with performance as initially installed
4. Maintenance - frequency of expected subsequent modification

SUMMARY

NASTRAN on the ILLIAC will look quite similar to NASTRAN operating today. This approach was selected because it would entail the least amount of initial effort to get a fully operational system up on the ILLIAC. The specifications presented above provide for the flexibility to subsequently select segments of the program for recoding to fully utilize the parallelism of the ILLIAC hardware. This flexibility is primarily due to the modular design of NASTRAN. On the ILLIAC, this modular design is extended even farther by constructing each functional module as an independent program, capable of being loaded and executed by the new NASTRAN Executive Sequence Monitor.

Because the ILLIAC operates in a single user environment, great care must be given to efficiently utilize all the real time taken by the system while running on the ILLIAC. This is accomplished by extending the checkpoint and GINØ functions of NASTRAN to more efficiently utilize this time. The data formats for input by, and output to the user are unchanged. The internal data file structure is, for the most part, unchanged. Only that data used to communicate between the major time-consuming modules is allowed to be specially designed to assist in optimizing those operations. Standards are established and new utilities are provided to meet these standards so that generality can be retained.

The user interface with NASTRAN on the ILLIAC, therefore, is nearly identical to current NASTRAN. The only difference is the necessity of accessing the ILLIAC via the ARPANET. However, this can be achieved from practically every major industrial center in the United States and it requires minimal facilities at the user end. Furthermore, the maintenance effort is centered on one machine with the capacity to service practically any number of users. As further optimization is achieved, errors are corrected or new capability is added, the user will have immediate access to these enhancements. Finally, due to this centralization of the maintenance and development effort, a similar centralization of analytical expertise on the part of the development staff can also be achieved.

CHAPTER V

TASK DEFINITIONS

This chapter outlines the individual tasks for implementing NASTRAN on the ILLIAC IV System according to the design specifications presented in Chapter IV. These tasks are organized and presented below in the form of a work statement so that individually, they are clearly and easily identified. Chapter VI, to follow, presents the estimated man-time and recommended schedule for performance.

TASK I - MAINTENANCE AND SUPPORT FACILITIES

This task involves development of all system interface facilities required for subsequent installation of NASTRAN on the ILLIAC IV System. The principal effort is in the modification of the IVTRAN "paralyzer" to preprocess existing FORTRAN code and identify code sequences suitable for ASK subroutinization, restructure multiple entry point routines, and develop block structured program output. Together with these modifications, the IVTRAN compiler must be adapted to accept complex variables, specialized syntax output by the paralyzer, and standard FORTRAN common blocks.

TASK II - UTILITY ROUTINES

This task involves design and development of key utilities required for initial installation of NASTRAN. These include:

1. General Purpose Input/Output

A General Input/Output package (GINØ) will be developed for ILLIAC NASTRAN. GINØ will use a double buffering method to achieve asynchronous input/output. GINØ will have the following capabilities:

a. Sequential input/output.

b. Random input/output.

c. Dynamic ILLIAC IV Disk Memory mapping.

d. Dynamic ILLIAC IV Disk Memory allocation and de-allocation with protection for non-allocated areas.

Three levels of development are required. First, to provide standard capabilities commensurate with existing NASTRAN code. Second, to allow computational modules to specify formatting. Third, to provide access by each module to scratch areas for any special purpose I/O.

2. Special Purpose Utilities

These utilities perform miscellaneous functions of a specific nature which may be used by any module. These functions include:

a. Standard packing/unpacking for conventional matrix data formats.

b. Special packing/unpacking for special purpose matrix data formats required for optimization.

c. Machine-dependent functions of MAPFNS as well as bit manipulation, core size, time, and data search functions.

d. Input and output utilities for interpreting input cards, formatting output, message writers, and debug facilities.

3. Matrix Operations

Several general purpose matrix operation modules are used frequently in every NASTRAN analysis. Design specifications will be developed for each of these operations to assure compatability with the new GINØ and Executive Sequence Monitor. Development may, however, be done in two steps:

a. Operations originally coded as machine-dependent will be optimized initially.

b. Operations originally coded as machine-independent may be installed directly and later optimized as dictated by the detail specifications prepared above.

TASK III - EXECUTIVE SEQUENCE MONITOR

Detail specifications will be developed for the Executive Sequence Monitor according to the design specifications presented in Chapter IV. Upon completion of the detail specifications, the ESM will be implemented to provide the following functions:

a. Interface with the Central Processor to control data communications between Central Memory and the ILLIAC I4DM disk.

b. Control Checkpoint and Restart operations.

c. Control data transfers between the I4DM and ILLIAC memory as required for communication between modules.

d. Allocate secondary storage for use internally as required by each module.

e. Control the executive sequence retaining the current matrix abstraction capability (DMAP) of NASTRAN.

f. Load and overlay each program segment.

g. Transfer control to the functional module.

TASK IV - ENGINEERING COMPUTATION ROUTINES

All of the machine-independent routines of NASTRAN will be installed on the ILLIAC. Required modifications to these routines will be effected in the NASTRAN FORTRAN subset. The FORTRAN routines will then be converted with the IVTRAN "paralyzer" and compiled with the IVTRAN compiler. Adaptation of these routines will be made to conform to the requirements of the special purpose utilities, the new GINØ, and the Executive Sequence Monitor. Any optimization to be implemented initially would be restricted to satisfying the structured programming requirements indicated by the output from the "paralyzer".

TASK V - DOCUMENTATION AND DEMONSTRATION

As each component of the new system, including all maintenance and support facilities are developed, full documentation will be provided prior to installation. Structured programming concepts will be employed to assure full integrity of the completed system. Hence, each segment of code will be demonstrated prior to integration. Following integration, demonstration of its performance will be repeated and the associated documentation will be updated.

Documentation will follow current NASTRAN standards for Programmer's, User's, Theoretical, and Demonstration Problem Manuals.

TASK VI - PERFORMANCE EVALUATION

Upon initial implementation of each NASTRAN capability, performance of that segment of code will be tested and evaluated for large problem execution. As each integrated capability, e.g., static, dynamic, non-linear, becomes available, several full-size structural models will be tested on the ILLIAC. These problems will be used to obtain timing and performance data on all of the NASTRAN modules. This information plus information on the frequency of use of the functional modules and the Rigid Formats, will be used to develop a list of priorities. This list will indicate for each module the priority for recoding to obtain maximum efficiency on the ILLIAC.

The checkpoint/restart process of ILLIAC NASTRAN will be evaluated to determine: (1) which data blocks should be checkpointed, (2) what size and type of analyses should use the checkpoint feature, and (3) the desirability of recoding module XCHK to achieve greater efficiency.

TASK VII - MODULE OPTIMIZATION

The list of priorities for module optimization will be developed to determine the priority in which modules are to be recoded. All recoding will be performed in IVTRAN and ASK. Recoded modules and subroutines will make efficient use of the parallel nature of the ILLIAC IV system architecture.

CHAPTER VI

ESTIMATED COST AND SCHEDULE

This chapter is devoted to developing the estimated costs, in terms of man-time, and schedule required to implement NASTRAN on the ILLIAC. The specifications presented in Chapter IV were used to identify the individual tasks defined in Chapter V.

The first phase, Tasks I through VI, for initial installation is estimated to require 60 man-months of effort over approximately 18 months. The second phase, Task VII - Optimization, is estimated to require 50 to 75 man-months and can be accomplished in 18 to 24 months following initial installation. The time-phasing of the first five tasks, including detailed manpower loading for each, is shown in Table 6.1.

These estimates are based on UAI's current experience and practical working knowledge of the ILLIAC System, its reliability, the capabilities of its software, and the expected level of support each has been given in the past. The operating conditions and assumptions made in developing these estimates are as follows:

1. Four (4) hour turnaround for each job from submittal to returned hard-copy output.
2. Capability to control user programs under TENEX on the ILLIAC IV System's PDP-10 Central Processor.
3. Prompt and able response from the ILLIAC IV User Support Group.
4. Complete and up-to-date documentation of the ILLIAC IV System hardware and software, including listings and source for the IVTRAN "paralyzer" and compiler.
5. Full documentation and complete source code for the latest tested version of NASTRAN.
6. Availability of serial processor with standard NASTRAN to obtain comparison test data.

The rational established for the phasing of each activity as shown in Table 6.1 is as follows for each task:

TASK I - Development of the IVTRAN "paralyzer" and compiler is essential to all subsequent implementation efforts.

TASK II - All communications require detail specifications for GINØ and the utilities necessary for internal processing of every module. Matrix operations to be installed as is or with optimization are also used in the execution of every NASTRAN Analysis.

TABLE 6.1 DETAIL MAN-TIME AND SCHEDULE ESTIMATES

First Phase - Installation	Man-Months	Months from Authority to Proceed																	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
I - Maintenance and Support	4	.5	1.5	1.5	.5														
II - Utilities																			
1. GINØ	7		.5	1.5	1.5	1.5	1.5	.5											
2. Special Purpose	6				1	1.5	1.5	1	.5	.5									
3. Matrix Operations	12							1	1.5	1.5	2	2	1.5	1.5	1				
III - Executive Sequence Monitor	5						.5	1	1.5	1.5	.5								
IV - Engineering Computation Routines	10										1	1.5	2	2	2	1	.5		
V - Documentation and Demonstration	10	2.5	2	1	1	.5									.5	.5	1	1	
VI - Performance Evaluation	6															1.5	1.5	1.5	1.5
Manpower Loading	60	3	4	4	4	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3	3	2.5	1.5

TASK III - The Executive Sequence Monitor is required primarily for integrated testing, hence, can be postponed until all details of GINØ and special purpose utilities have been worked out. However, it is required prior to installing the machine-independent code.

TASK IV - The final programming task of installing the engineering computational modules can best be performed last to take advantage of all the utility and Executive Sequence Monitor developments.

TASK V - Documentation is an essential task at the outset to plan the specific details of implementation. Once developed, updating is carried out as part of the installation effort. The final demonstration of the integrated system and user manual preparation completes this task.

TASK VI - As each analysis capability is functionally integrated, performance evaluation can begin to establish the priorities for the second phase of NASTRAN optimization on the ILLIAC.

The next, and final chapter, presents a summary of the conclusions drawn from this study.

CHAPTER VII

SUMMARY AND CONCLUSIONS

The following major conclusions can be drawn to summarize the above study of the feasibility of installing NASTRAN on the ILLIAC IV:

1. Installation of NASTRAN on the ILLIAC IV is feasible and is recommended.

2. The ILLIAC IV is now operational and has been demonstrated to be an excellent resource for finite element structural analysis processing with programs as complex as NASTRAN.

3. Adequate hardware and software support is provided to make optimal utilization of that capability a feasible objective.

4. NASTRAN can be installed to provide an integrated system operating entirely on the ILLIAC within approximately 18 months with an estimated 60 man-months of effort.

5. Full optimization of NASTRAN's current capability can be achieved within 18 to 24 additional months with an estimated 50 to 65 man-months of effort.

6. The installation of NASTRAN on the ILLIAC IV would provide access to users throughout the United States via the ARPANET.

7. The resulting centralized system provides for reduced overall costs of maintenance and development, and provides all users immediate access to all new developments.

With this resulting capability, large problems can be solved efficiently and within practical limits of total computer time. The ILLIAC IV is especially well suited to highly iterative types of problems such as would be encountered in nonlinear static and dynamic analyses. Though the power of the computer is extremely great compared with current serial processors, it does suffer from certain limitations:

1. The ILLIAC IV is dedicated to executing a single job at a time. Therefore, the effective total time of execution must include all phases of processing.

2. Data transfer from Central Memory to the ILLIAC IV disk for processing is slow compared with the transfer rates from disk to ILLIAC IV memory. This imposes practical limits on the size of problems to be solved and limits the practicality of a full checkpoint/restart capability.

3. The space available on the ILLIAC IV disk is limited, thereby accentuating the problem of slow data transfer to and from Central Memory.

In spite of these limitations, the overall capacity and speed of the ILLIAC IV offers a significant improvement over conventional third generation serial processors.

REFERENCES

1. Field, E. I., Johnson, S. E., Stralberg, H., "Software Development Utilizing Parallel Processing," Proceedings of the Symposium on Structural Mechanics Computer Programs, University of Maryland, 1974.

2. Frazier, G. A., Alexander, J. H., Petersen, D. M., "3-D Seismic Code for ILLIAC IV, Interim Report," SSS-R-73-1506, Systems, Science and Software, La Jolla, Calif., Feb. 1973.

3. Graham, W. R., "A Review of Capabilities and Limitations of Parallel and Pipeline Computers," Numerical and Computer Methods in Structural Mechanics, Academic Press, Inc., New York and London, 1973.

4. ILLIAC IV Systems Characteristics and Programming Manual, 55000D IL4-PM1, Burroughs Corporation, Defense, Space and Special Systems Group, May 1972.

5. Interim Systems Manual, ILSA, DNA001-72-C-0108, Universal Analytics, Inc., Los Angeles, Calif., Feb. 1973.

6. McIntyre, D. E., "An Introduction to the ILLIAC IV Computers, Datamation, April 1970, pp. 60-67.

7. McIntyre, D. E., "ILLIAC IV Language Evaluation - A Preliminary Report," ILLIAC IV Document No. 213, University of Illinois, May 1970.

A. NASTRAN Theoretical Manual, R. H. MacNeal, ed., NASA SP-221(01), April 1972.

B. NASTRAN User's Manual, C. W. McCornick, ed., NASA SP-222(01), June 1971.

C. NASTRAN Programmer's Manual, F. J. Douglas, ed., NASA SP-223(01), Sept. 1971.

8. "Press Seminar Handout," Prepared by the Institute for Advanced Computations, Ames Research Center, Moffett Field, Calif., Aug. 1973.

9. Systems Guide for the ILLIAC IV User, SG-I1000-0000-C, Institute for Advanced Computation, Ames Research Center, Moffett Field, Calif., July 1973.

10. Winograd, S., "On the Time Required to Perform Addition," J. Ass. Comput. Machinery, Vol. 12, No. 2, 1965, pp. 277-285.

11. Winograd, S., "On the Time Required to Perform Multiplication," J. Ass. Comput. Machinery, Vol. 14, No. 4, 1967, pp. 743-802.

12. Wirsching, J. E., and Alberts, A. A., "Application of the STAR Computer to Problems in the Numerical Calculation of Electromagnetic Fields," AFWL-TR-69-165, Air Force Weapons Laboratory, New Mexico, April 1970.

13. Wirsching, J. E., et al., "Application of the ILLIAC IV Computer to Problems in the Numerical Calculation of Electromagnetic Fields," ARWL-TR-69-91, Air Force Weapons Laboratory, New Mexico, March 1970.