1973

A simulation model of the control data 6400 scope operating system

Robert Kofi Baafi
Lehigh University

A SIMULATION MODEL OF THE CONTROL DATA 6400 SCOPE OPERATING SYSTEM

by

Robert Kofi Baafi

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1973

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

August 24, 1973
(date)

M W Shively
Professor in Charge

[illegible]
Chairman of Department

ACKNOWLEDGEMENTS

I wish to thank my advisers, Dr. M. W. Shiveley and Prof. A. F. Gould, for their guidance and assistance throughout the project. Supportive material from Dr. Shiveley was particularly appreciated.

I would also like to thank Richard Cichelli of the Lehigh University Computing Center Systems Group for making the Lehigh University Computing Center Daily Statistics available to me. His suggestions were very helpful.

Special thanks are due to my wife, Pamela, for her patience and understanding throughout the course of my Master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In the daily operation of a computer system, the need arises for a tool that can be used to evaluate the effect on system performance of changes in workload and modifications in hardware configuration, software parameters, and scheduling strategies. Simulation is the most flexible of the evaluation techniques.

This thesis describes a simulation model for the CDC 6400 computer system under the SCOPE 3.3 Operating System. The design goal of the model is to predict, with a 90 percent level of confidence, the performance of the system in terms of turnaround time, time-sharing response time, control point dwell, and central processing unit utilization. One of the most significant aspects of this model is that it simulates both the batch and interactive modes of the system's operation. Interactive jobs are modeled at the command level; and batch jobs are modeled at the task level. The validity of the model is improved by the use of parameters which represent the state of the system in terms of the availability of peripheral processors.

The results of a number of experiments conducted with the model are presented. The results show that it is necessary to compromise between batch turnaround time and time-sharing response time.

CHAPTER 1

PERFORMANCE EVALUATION

1.1 INTRODUCTION

Computer system performance evaluation is of vital importance in the design of hardware and software systems, the selection of computer systems, and the analysis of existing systems. The performance of a computer system depends not only on the hardware and the job load but also on the operating system that controls the flow of jobs through the system. The capabilities of the operating system significantly influence the performance of the computer. Compilers, assemblers, application programs and utilities are part of the computer system, and their performance affects the total system performance. The potential effect of software on the total system performance is widely recognized. The cost of software development and maintenance represents a significant portion of the total system cost. The evaluation of software in terms of cost and performance is therefore very important.

Lucas[30] has identified three reasons for evaluating the performance of a computer system: performance projection, selection evaluation and performance monitoring. Performance projection is usually involved in the design of a new hardware component or a software package. The design process involves the development of several design alternatives and the selection of the best alternative. The designer has to estimate the performance of systems that do not yet exist.

Performance is one of the major criteria considered when a new computer system is being purchased. Other factors besides performance are training and application programs provided by the vendor, reliability and expandability of the proposed configuration. The selection procedure includes both hardware and software evaluation.

Monitoring provides information on the actual performance of an existing system. A study on how the system can be improved usually accompanies the monitoring. A computer center is usually confronted with the question of how to deal with a growing load and for a changing job mix in the face of tight financial constraints. The needs may be met by altering the equipment configuration or changing scheduling and dispatching algorithms. Another solution may be to establish preferred job classes for different time intervals; for example, only jobs requiring central processor time of 200 sec. or less may be processed through the system between 9:00 A.M. and 12:00 P.M.

1.2 PERFORMANCE EVALUATION TECHNIQUES

The three basic approaches to computer system evaluation are the following:

(1) Monitoring
(2) Analytic evaluation, and
(3) Simulation

1.2.1 Monitoring

Monitoring is a method of collecting data on the performance of an existing system. Hardware or software probes are inserted into the system, and major system events are monitored and recorded. Two activities are involved in the monitoring process: the actual collection of statistics during operations and the analysis of the data -- which is usually done at a later time.

Hardware monitors measure parameters such as central processor time, peripheral processor time, disk read/write time, and waiting time. Given a specific set of readings, an analyst can calculate a number of performance measures such as total system idle time, compute only time, I/O only time, and etc. [3]. A major advantage of hardware monitoring is that the system is not disturbed in any way by the act of making measurements. Thus, the monitor described by Bonner [3] can measure many system parameters without affecting the normal operation of the system.

A job accounting routine, such as the "Day File" generator of the SCOPE operating system, is the simplest form of a monitor. This type of monitor can provide reliable statistics on central memory usage, peripheral processor time, central processor time, number of cards read, number of lines printed, and etc. The operating system automatically stores the Day File data on disk; it can be retrieved for analysis at a later time. More sophisticated software monitors can be implemented by altering the operating system itself. Software monitors may also be implemented by writing special routines and submitting them as user jobs. The disadvantage of a software monitor is that it can and does degrade the system during operation. A software monitor takes memory space and uses central processor and peripheral processor time. The major advantage of a software monitor is its flexibility in determining what parameters are to be monitored and in making changes in the statistics to be collected.

Monitors are primarily used to analyse the behavior of an existing system. Monitoring can be used to identify system bottlenecks and inadequacies. Monitoring is also useful in developing a profile on the use of a system. For example, the percentage of jobs which require central memory space of less than 70,000 (octal) words can be determined by monitoring. This data can be used to determine which programming systems the installation should support. For example, if statistics show a high percentage of compile time compared to execution time, an installation may decide to replace their compilers with faster ones.

1.2.2 Analytic Evaluation

The second approach to the evaluation of computer systems is the analytic approach. An analytic model is a mathematical representation of a computing system [30]. A number of such models have been developed and many have proven to be satisfactory; Sherr [44] was able to construct a simple analytic model of the project MAC system at MIT, and Smith [46] was able to construct a model reflecting a paged time-sharing system.

An advantage of analytic models is that, unlike the direct measurement technique, they can be used to predict the behavior of the system under different conditions. They can be used to investigate the effect of different system configurations and algorithm modifications on the performance of the system.

Analytic models do have some serious problems. The major disadvantage of analytic models is that they are difficult to develop. A multiprogramming computer system such as the CDC 6400 exhibits a high degree of randomness because of the interaction between the hardware, software, and user programs. Such a complex system cannot be represented very easily with a series of equations. The simplifying assumptions often necessary to develop analytic models tend, in many instances, to reduce the validity of the model.

Another disadvantage of analytic models is that they usually do not have enough flexibility to permit the investigation without substantial additional effort, in the effect of modifications in the system configuration, software parameters, and scheduling algorithms. Analytic models take a lot of time to develop and revise.

1.2.3 Simulation

The third approach, simulation, is the most potent tool for the evaluation of complex computer systems. Simulation is defined by Maisel and Gnugnoli (34) as "a numerical technique for conducting experiments on a digital computer; this technique involves certain types of mathematical and logical models that describe the behavior of business, economic, social, biological, physical, or chemical systems (or some component thereof) over periods of time."

Simulation offers several advantages over the analytic modeling and monitoring approaches. Simulation permits the study of a real system without actual modification of the system in any way. Monitoring would require changes to the system, thereby interfering with normal operations. Substantial costs may also be incurred if new hardware or software components have to be purchased or developed to monitor the system.

Realistic simulation models of complex systems can be developed with a reasonable investment of time and money. Analytic models may require simplifying assumptions which render the model invalid.

An important advantage of simulation is flexibility. Simulation is the most flexible of the three performance evaluation techniques. Simulation models can be built to study almost any computer system. Once the simulation model has been developed, the effect of any number of modifications in system configuration, priority assignment rules, scheduling rules, and dispatching rules can be investigated with a minimum of additional effort. Another advantage of simulation is that it provides operational insight into the system being modeled. This makes simulation a very <u>rewarding</u> educational experience.

The greatest limitation of simulation is its high cost. The development of a simulation model requires extensive studies of the system being modeled, additional time for the development of the model, and considerable time for execution on a large and expensive computer.

This thesis describes the development of a simulation model of the CDC 6400 computing system under SCOPE 3.3 and INTERCOM 3.0. The primary objective of this project was to develop a valid simulation model of the CDC 6400 computer system which can be used to predict the performance of the system under typical loading conditions. Another objective was to study how changes in workload, modifications in scheduling and dispatching rules, and changes in system configuration, such as different control point assignments, affect the performance of the system. These objectives are discussed in more detail in section 1.4.

The CDC 6400 computer system is a fast multiprogrammed multiple-computer system, which has eleven independent computers consisting of one large central processor and ten smaller peripheral and control processors. The SCOPE (Supervisory Control of Program Execution) operating system, version 3.3, is a group of programs that control the operation of the computer system. Katzan (28) defines an operating system as "an integrated set of control programs and processing programs designed to maximize the use of the system's resources and to reduce the complexity involved in preparing a program for execution on a computer".

1.3 PREVIOUS WORK

Simulation models of a number of computer systems have been developed. While the number of simulation projects reported in the literature is large, very few are relevant to the simulation of CDC 6000 series computing systems. This section discusses some of the most important projects that have dealt generally with the simulation of multiprogramming and time-sharing systems. Articles dealing with CDC 6000 series computer systems are of special interest.

One of the earliest reported projects was the study of a computer center by Hutchinson [26]. In this project, Hutchinson was interested in the total operation of the computer center, including manpower assignment and cost distribution; the computer was just one element of the total system.

Another important project was the study of the Compatible Time Sharing System (CTSS) performed by Sherr [44] at MIT. This study used a software probe added to the operating system to gather data about user interactions with the system and system responses to user actions. Sherr developed a simulation model of the system and used the gathered data for input to the model and as a means of validating the results of the simulation. He studied the performance of the system with different scheduling algorithms. One item of particular interest was the "transaction" approach described by Sherr for the representation of interactive user jobs. This approach, which was used in the modeling of the time-sharing part of this project, is defined and described in Chapter 3.

Noetzel [38] described a tool he used to measure and evaluate a computer system while it was in operation. He called this tool the "Meta-System". The Meta-System combined on-line measurement with simulation. Significant events in the operation of the system were monitored by software probes and the data collected was fed as input to the simulation model. Potential modifications to the system were evaluated using the simulation model, and the actual system was adjusted accordingly. Only the trial runs of this system were reported.

Norland and Bulgren (39) described a simulation model of GECOS III, the General Comprehensive Operating System for the Honeywell H-600 line of computers. The GECOS III operating system contains a data-gathering facility that automatically supplies important statistics. The data supplied by the operating system was used to drive and validate the model. The approach taken by Norland is very similar to the approach used in the modeling of the batch system in this report.

Statistical distributions of job characteristics were used to drive the model.

Cheng [8] reported on the development of a trace-driven model of the IBM/360 computer at IBM. This was a landmark article which introduced the technique of trace-driven modeling. In the trace driven approach, data traced on a real running system is used to run the model. The workload and the activities of system components in response to the workload are supplied as input to the model in the form of trace data. The trace data is obtained dynamically by monitoring significant events while the operating system and the workload interact [8]. Shiveley [45] used this approach in his dissertation.

Projects using CDC 6000 series computers have appeared in the literature lately. MacDougal of Control Data Corporation has written a number of papers on the simulation of operating systems [32, 33]. One of his earlier papers [32] dealt with the simulation of an ECS - based operating system. In it, he described the design of a simulator for a proposed Extended Core Operating System (ECOS) for the CDC 6400 and 6600 systems. Another paper [32] described the construction of a basic simulation model for a disk-based multiprogramming computer system.

Considerable effort has been expended at the University of Washington on the simulation of their CDC 6400 computer system. A group there has developed a package called SIM 6000, which can be used to simulate the CDC 6400 under SCOPE 3.2 operating system [40,37,25]. One major deficiency in their model was that it ignored the time-sharing part of the system.

The University of Texas has done extensive simulation studies on the CDC 6600 [42, 43, 4]. Sherman [43] described a trace-driven modeling analysis of CPU scheduling in the UT-1 operating system. The UT-1 and UT-2 operating systems are software packages which were written at the University of Texas. They both have software probes which automatically collect operational data. After testing several scheduling rules on the model, Sherman concluded that a successful CPU scheduling method must be pre-emptive and must prevent a given job from holding the CPU for too long a period. Another recent paper from the University of Texas written by Browne and Lan [4] dealt with the interaction of multiprogramming job scheduling and CPU scheduling on the CDC 6600. Browne and Lan experimented with several scheduling strategies, including first-come-first-served, shortest-processor-time-first, smallest-cost-first, and smallest-memory-requirement-first. They evaluated both pre-emptive and non-pre-emptive CPU scheduling (or dispatching) strategies. They concluded that pre-emption is a key element for high throughput. They also concluded that the best scheduling strategy depends on the specified measure of effectiveness. If it is desirable to maximize I/O utilization, smallest-cost (cost=core size + processor service time) or shortest-processor-time is the most desirable strategy. If the primary objective is to maximize CPU utilization, shortest burst time (SBT) is the best CPU scheduling strategy.

The two basic types of simulators are trace-driven simulators and statistically-driven simulators. The difference lies in the way input data is supplied to the simulator. A trace driven simulator uses raw data monitored during the normal operation of the system to drive the model. A statistically-driven simulator generates job parameters and system service parameters from statistical distributors and uses the generated data to drive the model. The simulation model discussed in this thesis is of the latter type.

The current project parallels that of the University of Washington - Seattle, Washington [25,37,40]. The difference is that the simulator described by this thesis models both the batch and time-sharing aspects of the computer system. It also models the dynamic allocation of memory during program execution. The University of Washington project dealt with SCOPE 3.2; this project examines operations under SCOPE 3.3.

1.4 OBJECTIVES AND SCOPE OF THIS PROJECT

The main objective of this project was to develop a valid simulation model of the CDC 6400 computer system under SCOPE 3.3/INTERCOM 3.0 operating system which can be used to predict the performance of the system under dynamic and different loading conditions. Another objective was to determine how changes in the workload, modifications in scheduling and dispatching rules, and changes in system configuration, such as different control point assignments, affect the performance of the system.

The performance measures selected for this evaluation study were: job turnaround time, interactive job response time, job throughput and equipment utilization. Turnaround time is the total time a user job spends in the system. It is the time lag between the submission of the card input deck and the receipt of output in the form of a punched output deck and/or a printout. The response time is the time between the initiation of a request (issue of a command) at a remote console and the beginning of the reply from the system. Response time is a function of the number of users of the system, the nature of their requirements, the nature of the specific request, and the design of the system. Throughput is the number of jobs completed by the system in a specified interval of time.

Response time, throughput, and turnaround are all meaningless figures unless they are backed up with information about the circumstances under which they were collected. Average, maximum and minimum figures for these measures of effectiveness (MOE'S) are based on statistics collected over a period of time and are more meaningful than isolated individual MOE'S. In this project, the average, maximum and minimum figures for the MOE'S were used to evaluate the system. No attempt was made to predict how long a specific job spent in the system since it would have been almost impossible to specify all its interactions with other jobs in such a dynamic system.

Utilization of a system component is the percentage (or fraction) of time when the component was busy. It is the sum of all busy time intervals divided by the total time when the component could have been working. In this project, utilization statistics were collected for the card reader, the central memory (occupancy), the central processor, the disks, the line printers, the card punches, and the seven control points.

The effects of changes in the scheduling and dispatching strategies on each of the MOE's were investigated. Scheduling can be found in the literature to refer to the selection of a user job in the input queue for central memory assignment, the selection of a job in the central memory for execution by the central processor, and the selection of an input/output request for input/output processing. In this report, scheduling is used to refer only to the selection of a job in the input queue immediately prior to central memory assignment. Dispatching is used to refer to the selection of a job within the central memory for execution by the central processor.

The different kinds of scheduling and dispatching algorithms are limited only by one's imagination. For this study, the following strategies were considered: priority, round-robbin, pre-emptive dispatching, non-pre-emptive dispatching, smallest-memory- requirement-first (SMF), largest-memory-requirement-first (LMF), and shortest-processor-time-first (STF). In priority scheduling or dispatching, the job with the highest priority gets the central memory space or the CPU. Pre-emptive means that when a higher priority job requests the use of a facility which is currently being used by a lower priority job, the lower priority job loses the facility to the higher priority job. Round robbin is the scheduling or dispatching strategy where the first job that requests a facility gets it when the facility becomes available; the current user goes to the end of the queue if he requests the use of the facility again. The other strategies are self-explanatory.

Simulation was selected for this evaluation study because of the flexibility inherent in the technique.

This chapter has outlined the three approaches to the evaluation of computer systems. The objectives of this simulation project and the principal measures of effectiveness were also discussed. Chapter 2 describes the computer system being modeled; it is organized into three parts. The first part which is introduction discusses the CDC 6000 series computer systems and gives a brief historical background on the development of the SCOPE 3.3 operating system.

Chapter 3 describes the simulation model. The features of the model, the level of detail, the constraints on the investigation, and the design of the simulation model are discussed. Chapter 4 discusses the validation of the model and the results of the experiments conducted using the model. Finally, Chapter 5 concludes this thesis with a summary of the important issues in the development of the simulation model and presents suggestions for further studies.

CHAPTER 2

THE CDC 6400 COMPUTING SYSTEM

2.1 INTRODUCTION

2.1.1 CDC 6000 Series Computers

The CONTROL DATA 6000 series computer systems are highly modular, large-scale, solid-state, general-purpose digital computer systems. This series consists of the 6200, 6400, 6500, 6600, and 6700. Each of these computer systems consists of a "main frame" and a number of peripheral equipment. A complete system requires input and output peripheral equipment such as card readers, magnetic tape units, disk storage units, printer units and an operator console. The "main frame" contains the central memory (CM), one or two central processors (CPU's), ten peripheral and control processors (PP's) and twelve data channels. One option is extended core storage (ECS); it can be included in some of the systems (6400, 6500, 6600, 6700) to augment rapid access memory. The 6500 and 6700 each have two central processing units; all the others have one.

There are two types of central processing units used in the 6000 series computers: the unified arithmetic central processor and the functional central processor. These are terms used by CDC to describe the two CPU's. The unified CPU is used in the 6200, 6400, and the 6500. The 6500 has two of these unified CPU's. The unified arithmetic central processor executes instructions in a serial order. The functional central processor has ten functional arithmetic and logical units which can operate concurrently on unrelated instructions. This functional CPU is used in the 6600 and the 6700. The 6700 contains one unified CPU and one functional CPU.

The only difference between the CDC 6600 and the CDC 6400 is that the CDC 6600 has the faster functional central processor while the CDC 6400 has the unified central processor.

2.1.2 Historical Background

The CDC 6600 computer was primarily conceived and developed by one individual, Seymour Gray. He worked with a group of hardware engineers on his farm at Chippewa Falls, Minnesota, to develop and fabricate the first few units of the CDC 6600.

The first 6600 was completed and delivered without software to the Livermore Atomic Energy Laboratory in Livermore, California, in 1964; it marked the beginning of a new generation of computing systems. This computer consisted of one central processor and ten peripheral and control processors. Control Data at this time, decided to include the 6600 in its standard product line. A special software group was organized to develop a sophisticated operating system to market with the sophisticated hardware of the 6600. Unfortunately, the unit of the 6600 which this group had to work with turned out to be more unreliable than any other built before or since. Its reliability drastically curtailed the effectiveness of this software group.

In the meantime, the hardware group at Chippewa Falls realized that they needed a simple operating system in order to conduct extensive tests on the hardware. They rapidly developed a system, the Chippewa Operating System.

The delivery of the 6600 in the general market did not really begin until 1965. The early users of the 6600 had to either develop their own software or use the Chippewa system which was still not completely debugged. The official CDC 6600 operating system, SIPROS (Simultaneous Processing Operating System) was still under development. At the first users' meeting in August 1965, CDC announced it's decision to fully back the Chippewa Operating System because of the extensive initial use. Then, rather than supporting a single software effort, Control Data attempted to split its meager resources.

The performance of SIPROS was never adequate. After a few frustrating years CDC decided to terminate all support of SIPROS and concentrate on the Chippewa System. The Chippewa System eventually evolved into the current operating systems, SCOPE.

2.2 HARDWARE ORGANIZATION

2.2.1. Overview

The Control Data 6400 is a multiprogrammed file-oriented system. It consists of eleven independent computers, one large central processor and ten small peripheral processors. The large central processor is the main computing element and is similar to a typical central processing unit of a general purpose computer. All the eleven computers have access to the central memory. The central processor has no connection with the ten peripheral processors except through the central memory. The peripheral processors can communicate with the external peripheral equipment and with each other via the twelve independent bi-directional data channels. Figure 2.1 is a general block

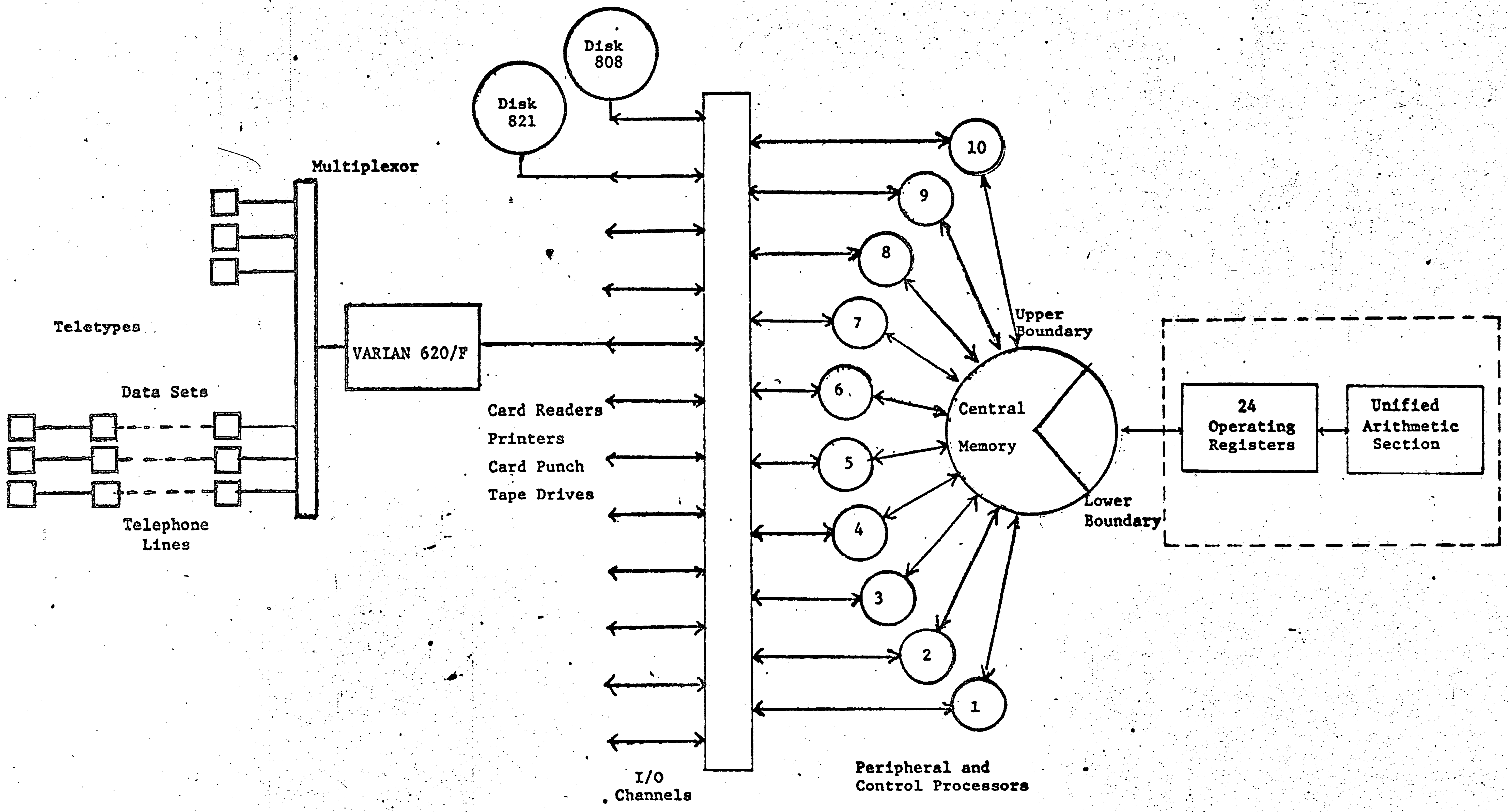


FIGURE 2.1 BLOCK DIAGRAM OF THE CDC 6400 SYSTEM

diagram of the CDC 6400 computer system.

A large variety of peripheral devices is available with the 6400 computer. The peripheral devices attached to the 6400 at Lehigh University include:

1) two disk storage units, the CDC 808 and 821 units,
2) four magnetic tape units,
3) telephone data set controllers,
4) one card reader,
5) two line printers,
6) one card punch,
7) two CALCOMP plotters,
8) one paper tape reader, and
9) one paper tape punch.

There is a VARIAN minicomputer (VARIAN 620/F) which handles all communication between the system and all remote terminals. The paper tape reader, the paper tape punch, and the two CALCOMP plotters are also connected to the VARIAN.

The central processor, the central memory and the peripheral processors are discussed in the next sections in more detail.

2.2.2 Central Processor

The CDC 6400 computer system has the basic unified arithmetic central processor. In the unified arithmetic central processor, instructions are executed serially, with no concurrency. The central processor is a very high-speed arithmetic processor, which consists of an arithmetic unit and a control unit. The arithmetic unit contains all the logic necessary to execute arithmetic, manipulative and logical operations. The control unit directs the arithmetic operations and provides interface between the arithmetic unit and the central memory. It also performs instruction retrieving, address preparation, memory protection, and data retrieval and storage.

The central processor does not perform any I/O operations. All I/O operations and system control tasks are handled by the peripheral processors. The central processor has no connection to the peripheral processors except through the central memory under the cognizance of two instructions, the exchange jump instruction and the central program address instruction. The central program address instruction monitors the address of the program currently using the central processor. The exchange jump instruction, which may be issued by a peripheral processor, initiates the program execution in the central processor or interrupts a current program and starts execution of a new program. In either case, the central processor is directed to a central memory file of 16 words which stores information about the new program to be executed. The 16-word file is called the exchange jump package.

The exchange jump feature allows the transfer of control among programs and the operating system; this is the essence of multiprogramming. Before a jump to a different program occurs, the contents of all the 24 registers in the central processor are loaded into an area of central memory. The exchange jump instruction is hardware implemented.

2.2.3 Central Memory

The central memory of the CDC 6400 system at Lehigh University has 65K decimal (base 10) 60 bit words. It is organized into 16 logically independent banks of 4096 words each.

References to the central memory from the PP's and the CPU are made via a common address clearing house called a stunt box. The stunt box accepts addresses under a priority system at a maximum rate of one address every 100 nanoseconds. The address is sent to all banks, and the correct bank, if free, accepts the address and notifies the stunt box of this action. The associated data word is then sent to or retrieved from a central data distributor. The correct bank ignores the address if it is busy processing. If this happens, the stunt box saves the address and reissues it later. The addresses that are saved because of conflicts reside in a hopper mechanism which has the highest priority in issuing addresses. The central processor has the second priority and the peripheral processors have the third priority. Figure 2.2 is a block diagram of the stunt box.

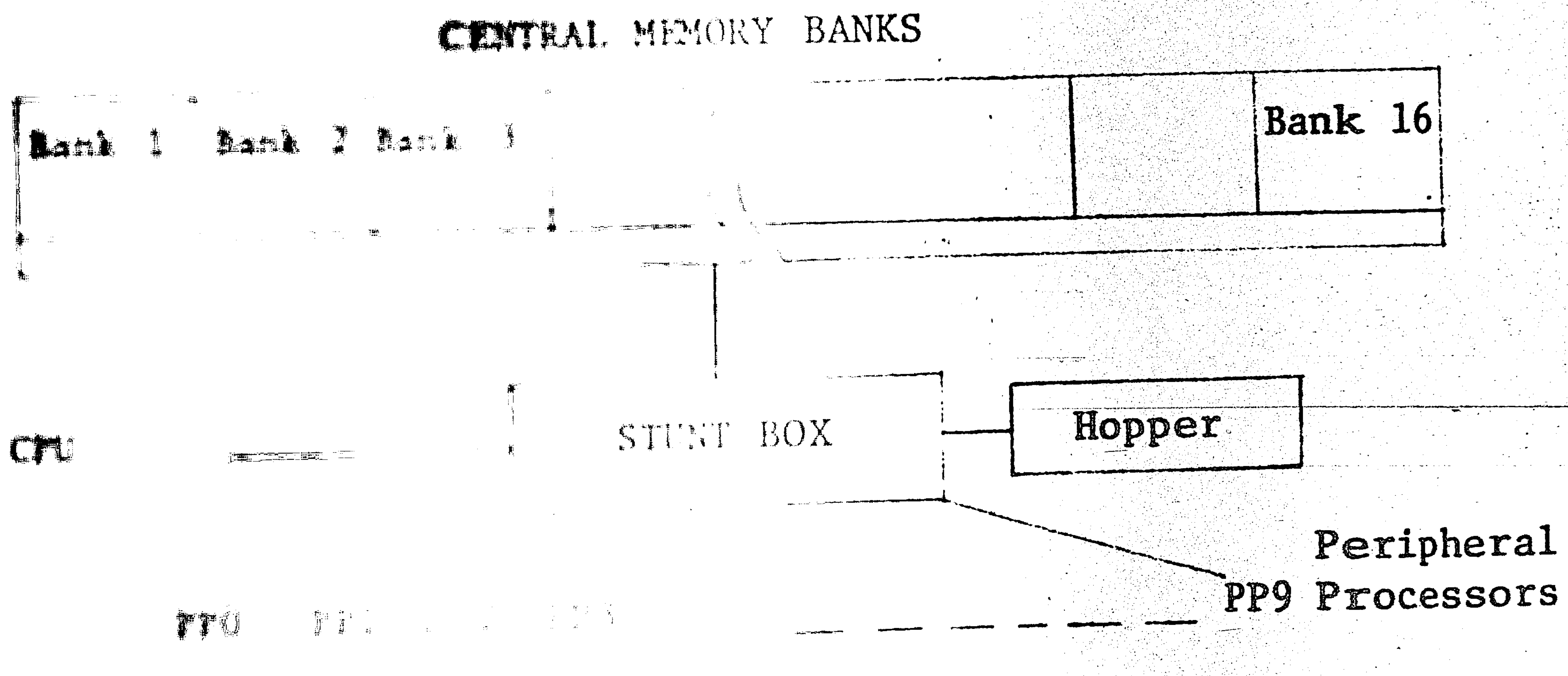


FIGURE 2.2 THE STUNT BOX

In a manner similar to the way the stunt box handles the address, the data distributor handles the actual transfer of the content of the word associated with the address. Data words and addresses are correlated by control information (tags) entered in the stunt box with the address. The tags define the address sender and the origin/destination of the data, and determine whether the address is a Read, Write or Exchange Jump address.

The data distributor has a series of buffer registers which provide temporary storage for words when a conflict occurs. Each group of four banks communicates with the distributor on separate 60-bit read and write paths, but only one word is transferred at a time. Words are transferred every minor cycle or 100 nanoseconds. Figure 2.3 is a simplified block diagram of the data distributor.

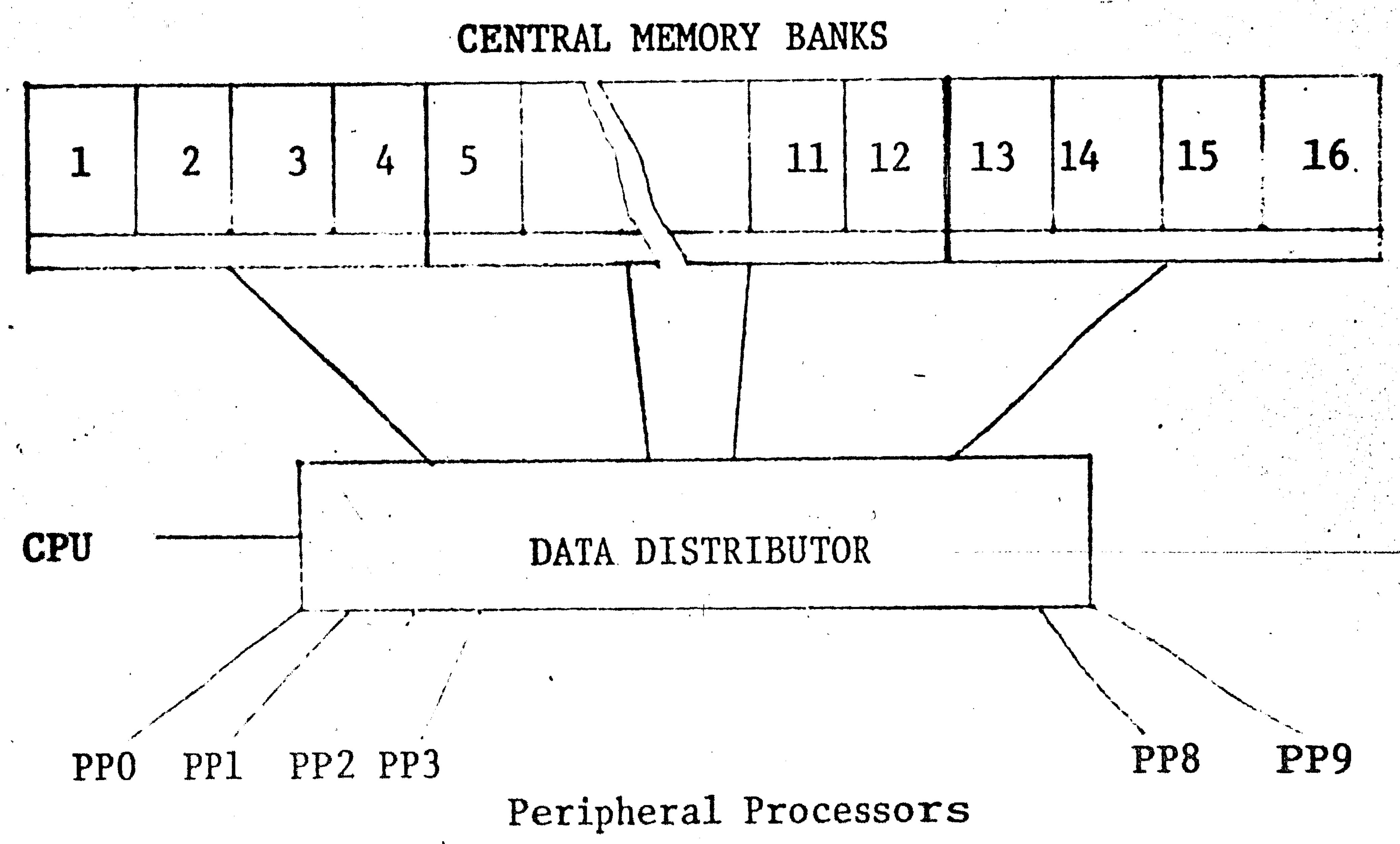


FIGURE 2.3 DATA DISTRIBUTOR

All central processor references to the central memory are made relative to a Reference Address (RA). The reference address defines the lower limit of a program in central memory. The capability to change the reference address permits easy relocation of programs in central memory. This hardware feature aids the software in program loading and dynamic relocation.

2.2.4 Peripheral and Control Processors

The ten peripheral and control processors (PP's) are identical independent stored-program computers. A peripheral processor consists of a control unit and a memory of 4096 12-bit words. The ten PP's share a fast arithmetic unit on a time-division multiplex scheme.

The ten PP's can communicate with external equipment (disks, card readers, printers) via the 12 independent bi-directional I/O channels. Only one piece of equipment can be attached to a channel at any time, but all 12 channels can be activated simultaneously. A peripheral processor may communicate with another peripheral processor over a channel which is selected as output by one and input by the other.

Data is transferred into or out of the system in 12-bit words; each channel has a single register which holds the data word being transferred. Each channel operates at a maximum of one word every 100 nanoseconds. Data being transferred between the central memory and an external device has to pass through the memory of a peripheral processor and the register of a data channel.

Data flows between the central memory and the memory of a PP in blocks of words. When data is being transferred between the memory of a peripheral processor and central memory, five consecutive PP words are assembled to make one 60-bit CM word. Similarly, when data is being transferred from central memory to a peripheral processor, each central memory word has to be broken down into five 12-bit words.

One of the peripheral processors (PP0) monitors and controls the operation of the central processor and the other peripheral processors. The executive monitor program (MTR) resides permanently in PP0 and controls the operation of the whole system. The system display program (DSD) which handles communication between the operator and the operating system resides permanently in PP1. Another peripheral processor is assigned to the VARIAN front-end minicomputer which handles all communication between the system and all remote terminals. The remaining seven PP's form a pool of processors from which the monitor may assign tasks as required.

2.3 SCOPE 3.3 OPERATING SYSTEM

2.3.1 Introduction

SCOPE (Supervisory Control of Program Execution) is a group of programs and subprograms that monitor and control input, compilation, assembly, loading, execution and output of all programs submitted to the computer, and allocate resources to these programs.

The system resources that have to be shared among jobs are the central memory, the central processor, the peripheral processors, data channels, disk controllers, line printers, punches, plotters and paper tape readers/writers. The central memory is shared between system routines, system communication areas, and user jobs. The amount of space available to user jobs is finite, and even though up to seven user jobs may be multiprogrammed, i.e., up to seven jobs may reside in core simultaneously, the sum of their core storage (CM) may not exceed the maximum amount of CM space allowed for user jobs. The maximum space allowed its user jobs at Lehigh is 140,000 (octal, base 8) words. The central processor can be used by only one job at a time; therefore, the jobs in central memory have to wait their turn for the CPU.

The central memory and the central processor are the two main resources that have to be shared by jobs. Queues of jobs waiting to be assigned to these facilities are maintained by the system. Also, a queue is maintained for each of the peripheral devices such as the disk, printers, punch, and so on.

Programs written for execution in the peripheral processors (PP programs) perform all the functions of the operating system. SCOPE consists of a monitor routine (MTR) which monitors and directs the whole operation of the computer system and resides permanently in a PP (PP0); a system display program (DSD) which resides in PP1 and handles all communication between the operator and the computer system. There are a number of PP programs which reside in core or on disk and are called by the monitor (MTR) to perform specific tasks.

A product set that comes with SCOPE 3.3 is INTERCOM (version 3.0). INTERCOM is a system of programs which allows for the submission of jobs to the CDC 6400 computer from remote terminals on a time-shared basis with other remote jobs and locally submitted jobs. Jobs are submitted on one of two bases: either as complete jobs punched on cards and submitted through the card reader of a 200 User Terminal, or as interactive conversational jobs submitted via teletypes (TTY) and cathode-ray-tube (CRT) terminals.

Communications between the remote terminals and the central computer are handled by data sets (TUCK, BELL, MILGO), a VARIAN 620/F minicomputer, a dedicated CDC 6400 I/O channel and a dedicated peripheral processor. The peripheral processor and the I/O channel assigned to INTERCOM cannot be used for any other purposes during the entire time that INTERCOM is running. All other hardware requirements, except the central memory, are on a temporary, or transient basis. INTERCOM uses pool PP's for running various PP programs. External storage devices (disks) are required for storage of INTERCOM system programs and authorization information, as well as for the files generated by the users of INTERCOM.

2.3.2 SCOPE Monitor (MTR)

The SCOPE system functions under the direction of the system monitor (MTR) which is loaded into PP0 at deadstart and remains there for the duration of system execution. Primarily, MTR controls and coordinates system activities to avoid conflicts between various system processors. The tasks performed by MTR include the control of the execution of CPU programs, the assignment of tasks to pool peripheral processors, the allocation of central memory, the reservation and assignment of data channels and peripheral devices, the periodic re-evaluation of control point priorities, and the maintenance of accounting and the system clock. (See Figure 2.4)

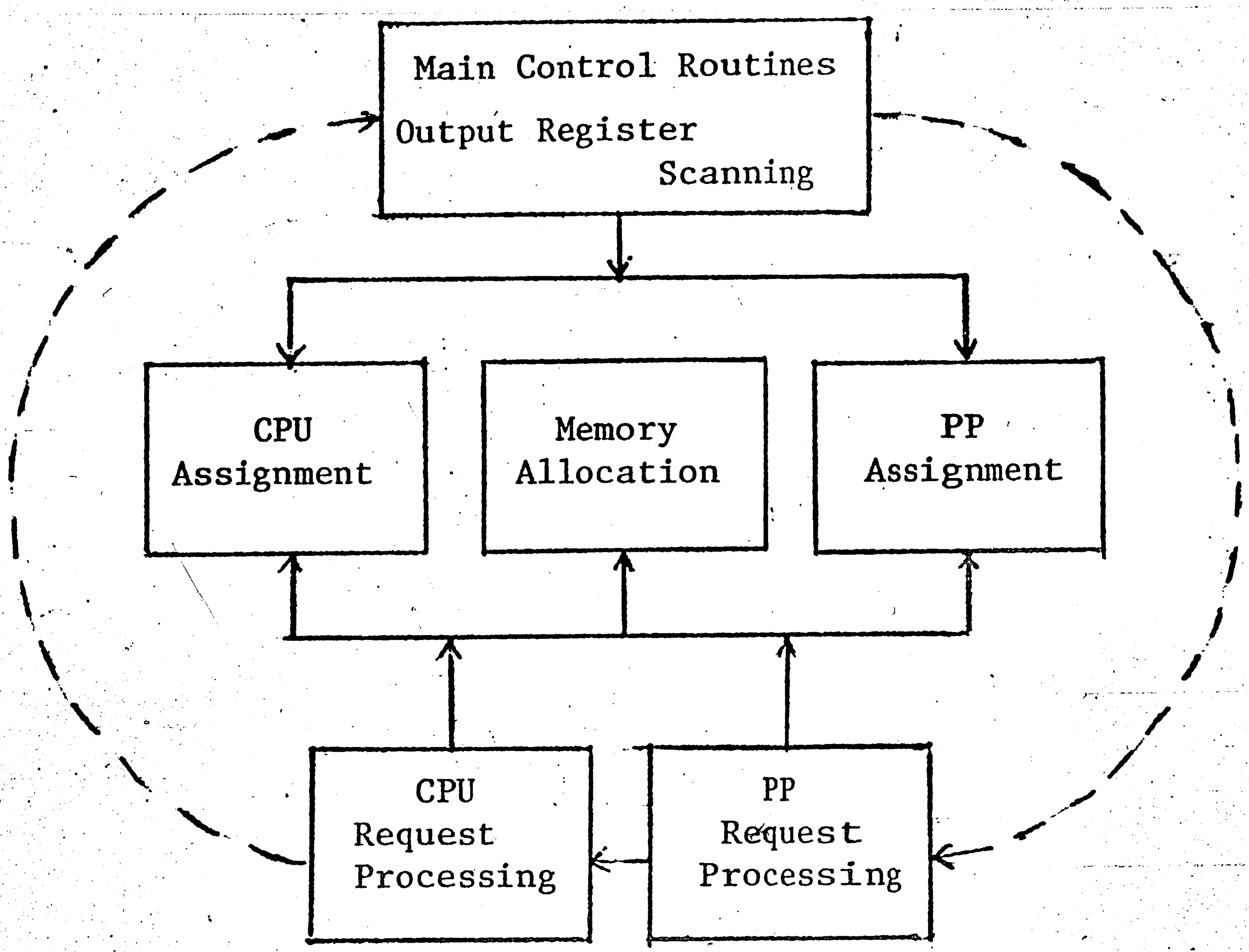


FIGURE 2.4 MTR MAIN LOOP

The monitor routine goes through its loop every 4 milliseconds. During each loop, MTR scans a set of central memory locations that are set by other peripheral processors when they require monitor action. When MTR finds any request or a message, it jumps to an internal subroutine to process the request. The monitor then scans the communication areas of all control points for any requests. Running jobs may request PP's on I/O devices. MTR assigns and releases all peripheral processors and all I/O equipment required by running jobs.

The monitor (MTR) is in control of the system at all times. All the other tasks of the operating system are performed by peripheral processor (PP) routines. The PP routines reside in core or on disk, and are called by the monitor when needed. If one of these routines is needed, MTR puts the name of the routine in the communication area of a PP. The PP picks up the name, locates it in core or on disk, loads a copy of it into its memory and transfers control to it. After the PP completes its task, it signals MTR via the communication area, and waits for the next MTR order.

The SCOPE monitor system works in the "mailbox" fashion. All communication between the monitor (MTR) and the devices and other operations it controls have to go through the communication area in central memory. There are no interrupts.

2.3.3 Central Memory Residence

Portions of the operating system reside in central memory throughout the operation of the system. The portions of the operating system which reside in central memory are the central memory resident (CMR), the library directory, the central memory library of commonly used PP programs, the INTERCOM buffer area, and the record block table (RBT). Figure 2.5 illustrates the allocation of central memory to portions of the operating system and to the user programs.

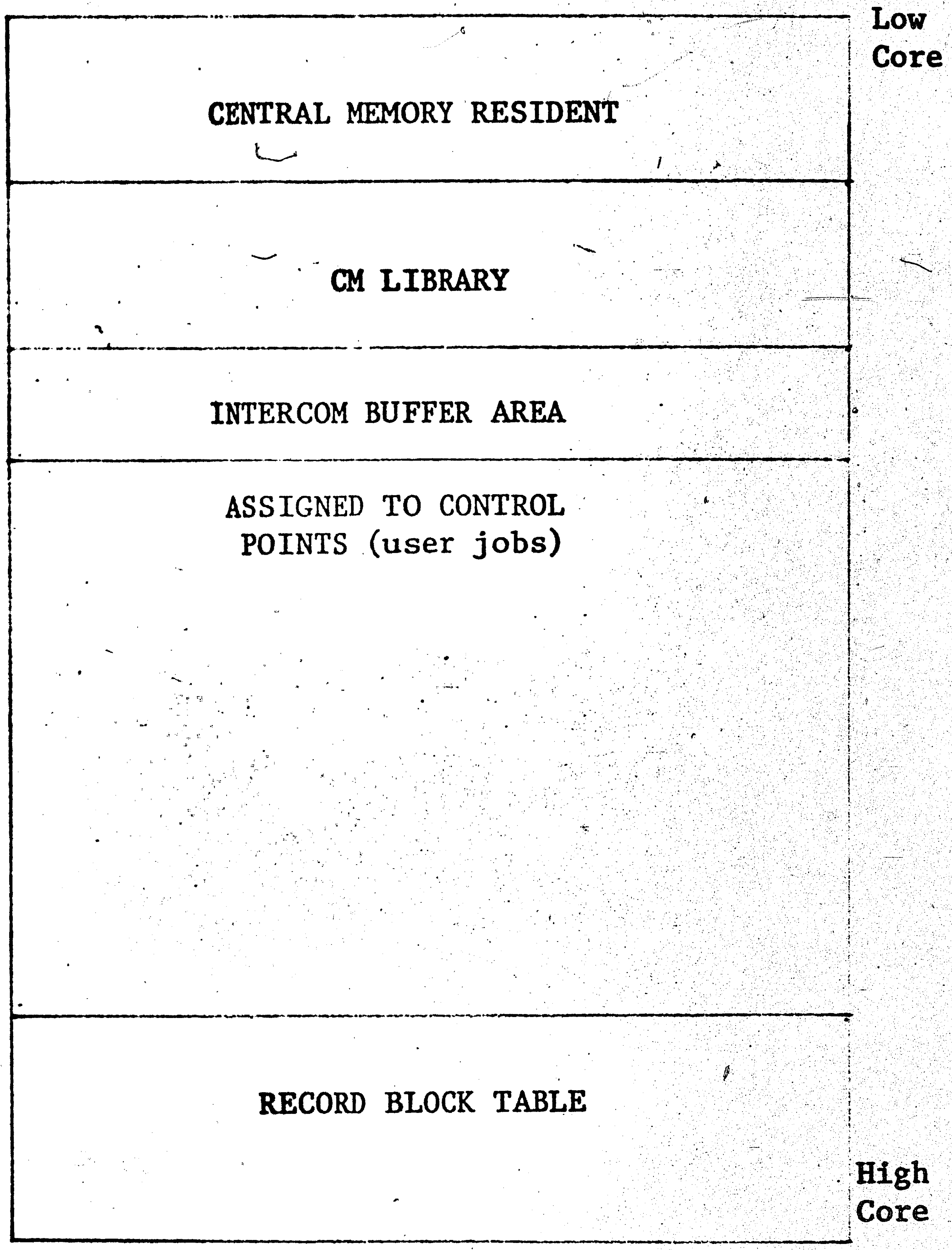


FIGURE 2.5 CENTRAL MEMORY ALLOCATION

The "central memory resident" contains pointers, tables, PP communication areas, and the central processor resident area. The INTERCOM buffer area contains pointers, message buffers for operator communication, user tables for keeping track of connected terminals, and data buffers for data being transmitted to/from terminals.

Blocks of central memory that are not allocated for system use may be utilized by user programs. The user program area is partitioned off into seven variable-sized partitions called control points. An active program has to reside at one of these seven control points. Control points may be assigned to specific classes of jobs. Any of the control points may be assigned to INTERCOM jobs by the operator. A control point so designated may be assigned only to INTERCOM system routines or interactive user jobs. Another control point may be assigned as a buffer area for the storage of system routines and overlays such as JANUS. The remaining control points are used by batch jobs.

Two other areas of central memory may be referred to as control points. An area just below control point one is referred to as control point zero. Control point zero is used to identify all hardware and software resources which are not presently allocated to user jobs or those that are used only by SCOPE. The other system pseudo-control point, control point 8, is used by SCOPE for system utility programs.

2.3.4 Priorities

The selection of a job for central memory allocation and the assignment of the central processor to an active job are based on the priorities of all jobs waiting for the availability of the facility. The priority number of a job in the SCOPE system is stored in twelve-bits; this indicates the relative importance of the job. This number can range from 0 to 7777 octal. For convenience, we shall assume that a priority number consists of four octal digits. The high order digit is called the priority level and the three low order digits are called the priority sublevel. The number of priority levels is an installation parameter (IP.MPR) which is equal to 8 at the Lehigh installation. There are two types of priorities in the SCOPE system, fixed and non-fixed priorities. Fixed priorities remain constant throughout the life of a job in the system, unless changed by operator intervention. Non-fixed priorities are subject to periodic recalculation upward or downward depending on a job's activity within the system and on its elapsed time in the input and output queues. During priority recalculation, only the sublevel (last three digits) is recomputed. The level (first digit) remains constant. Thus, a priority of 2000 octal can never exceed 2777 octal, no matter how long the job stays in the system.

All fixed priorities are higher than any non-fixed priority. The installation parameter (IP.LVF) which is equal to 5 at the Lehigh installation establishes the lowest fixed priority level. Thus any priority less than 5000 is non-fixed and may be recomputed by the SCOPE system.

The priority level of a batch job may be assigned by the user on the job card. The maximum priority level that may be specified on a job card is 4. The operator or the system may assign priorities 5, 6 and 7. All interactive jobs are assigned the fixed priority level of 7. All interactive jobs, therefore, have a priority of 7000 octal. System routines are assigned a priority of 7777 octal.

The scheduling priorities of jobs in the input queue are treated in the following manner. All fixed priorities remain constant. The priority sublevels of all non-fixed priorities are incremented by one (up to their maximum value of 777 octal) periodically. This process is known as aging the input queue. The time period between incrementing sublevels is determined by the installation parameter IP.IQD. Recalculation takes place every 2**(IP.IQD+6) milliseconds. At the Lehigh installation where IP.IQD is equal to 6, recalculation takes place every 16 seconds.

The scheduling priorities of jobs in the output queue are handled in the same fashion, except that the installation parameter IP.OQD determines the time delay between recalculation. The value of IP.OQD at Lehigh is 10 (octal); therefore, recomputation takes place every 64 seconds.

The priority of a control point is the priority of the job residing at that control point. Control point priority is initially established when the PP routine 1RA schedules a job in the input queue by attaching it to one of the seven control points. The initial value of the control point priority is the final value of the job priority in the input queue. For fixed priority jobs, this value has not been altered since the job entered the system. For non-fixed priority jobs, the initial control point priority may be greater than the job card (or default) priority; this can be caused by priority sublevel recalculation which may have occurred while the job waited in the job input queue. The time interval between priority recomputations is determined by the installation parameter IP.CPD. At the Lehigh installation where IP.CPD is equal to 5, priority recomputation takes place every 2.048 seconds. The new sublevel, S(new) is calculated using the following formula (43):

$$S(new) = \frac{\frac{T-CPT}{A} + (W-1) * S(old)}{W}$$

where,

T = 2**(IP.CPD+6), time interval between recalculations

= 2048 milliseconds

CPT = sum of the accumulated milliseconds of CPU time tallied at the control point since the last recalculation

A = 2**(IP.CPD+6-N)

N = 9, number of bits of sublevel

W = 2**2, a weighting factor

S(old) = the old priority sublevel

Upon examination of this formula, one can see that a job which has made use of the CPU during the latest interval will be penalized when recalculation occurs, whereas a waiting program which has not had access to the CPU will be rewarded. Therefore, within a given non-fixed priority level (determined by the non-changing high order digit), all programs will eventually have access to the CPU. However, a 0 level program will never run ahead of a 1 level program regardless of how long the 0 level program has been waiting. This is not a serious problem at the Lehigh installation because the majority of the batch jobs are at the same priority level of 1.

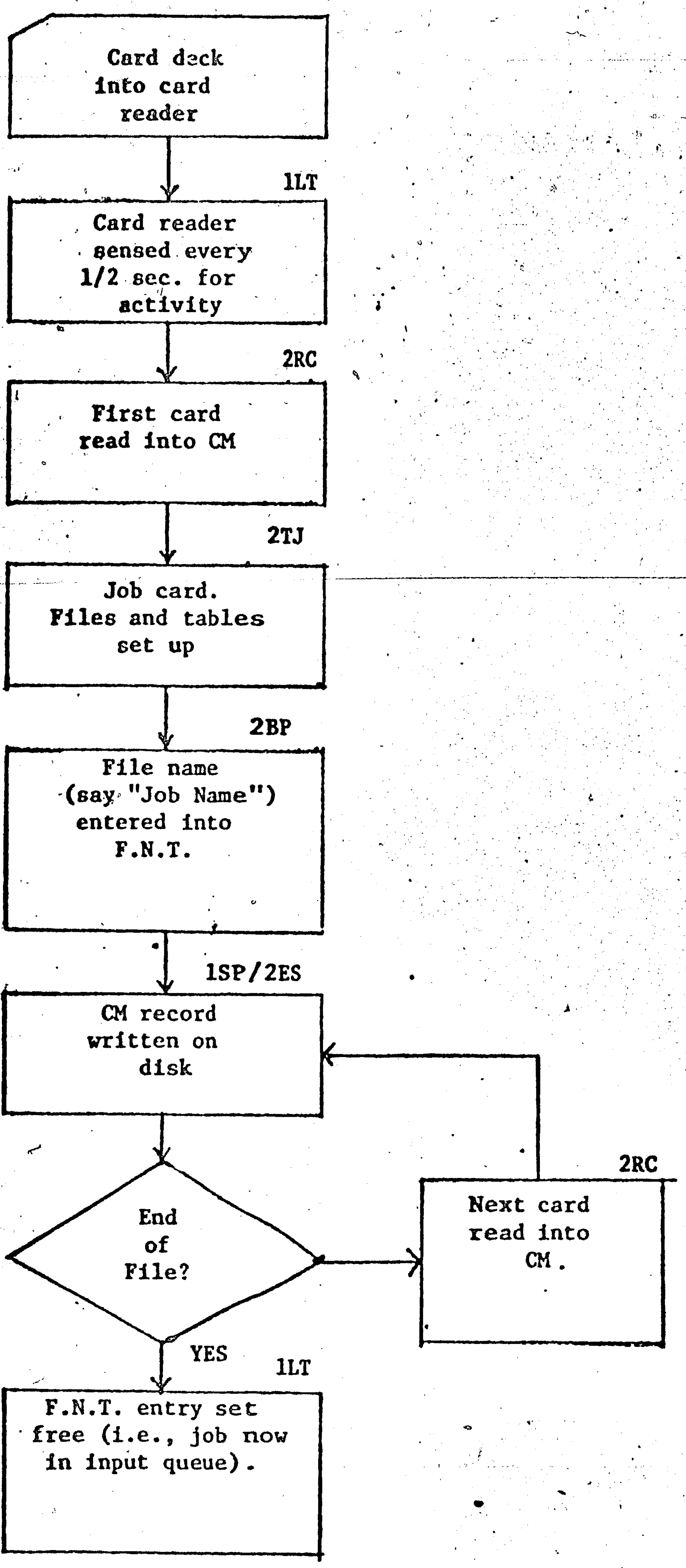
2.3.5 The Processing of Batch Jobs

The operation of the SCOPE system on the CDC 6400 computer was modeled by tracing the flow of jobs through the system. Before the model can be presented, the flow of jobs through the actual system should be described in detail. Three types of jobs are processed by the system: normal batch jobs, remote batch jobs and conversational or interactive jobs. The portion of the operating system that handles interactive jobs is known as INTERCOM. The flow of interactive jobs through the system is described separately in Section 2.3.6.

The SCOPE operating system is a file-oriented system. Two files, designated as INPUT and OUTPUT respectively, are created for every job that enters the system. Other files may be created by the user program or by the operating system for the user job.

A batch job consists of a number of control cards, program deck(s), and data deck(s). A batch job may enter the system via a card reader at the installation, via a card reader at a remote location (remote job entry), or via a magnetic tape drive as card images on magnetic tapes. The system is usually entered through the card reader at the installation. The flow of a typical batch job through the system is now discussed in detail. The flow of jobs through the system is monitored and controlled by the monitor (MTR). The monitor calls on peripheral processors and PP routines to perform tasks for the job. Some of the more important PP routines are mentioned at the appropriate time.

Figure 2.6 is the flow chart of a job entering the system. The operator loads the job card deck into the card reader. Every one-half



Card deck into card reader
1LT
Card reader sensed every 1/2 sec. for activity
2RC
First card read into CM
2TJ
Job card. Files and tables set up
2BP
File name (say "Job Name") entered into F.N.T.
1SP/2ES
CM record written on disk
End of File?
2RC
Next card read into CM.
YES
1LT
F.N.T. entry set free (i.e., job now in input queue).

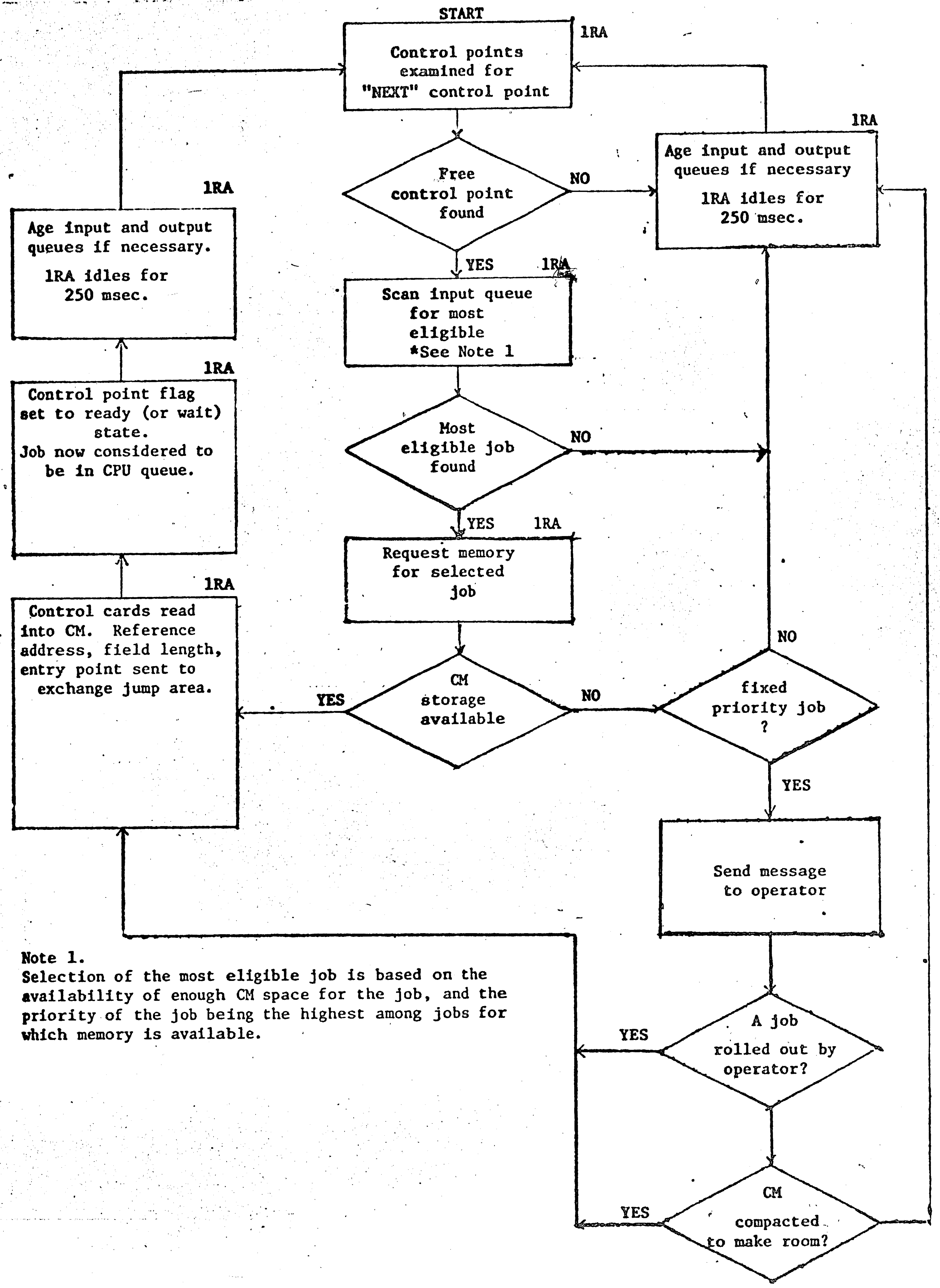
Where:
FNT - File name table
CM - Central memory
1LT - Load job peripheral program
2TJ - Translate job card PP routine
2BP - Buffer parameters PP routine
1SP - Disk I/O
2ES - Enter stack reg. PP routine
2RC - Read cards PP routine

FIGURE 2.6 ENTRY OF BATCH JOB INTO THE SYSTEM

second a PP routine (1LT) checks to see if anything is in the card reader. If a deck is waiting, a PP routine (2RC) reads the first card (which must be a job card) into central memory. The job card contains the job name, the central processor time limit, the total central memory requirement for the job, and the priority level (in octal) for the job. A routine (2RC) then passes control to a PP routine (2TJ) which, along with another routine (2BP) reads the remaining control cards and sets up all necessary files and tables, including the File Name Table (FNT). Finally, two PP routines (2ES, 1SP) write the remaining card images on disk storage until an end-of-file card is encountered. The job is now in the input queue.

In order for a job to be scheduled from the input queue into the active job mix, i.e., into the central memory, three conditions must be met. First, one of the seven control points must be free; a dummy file NEXT attached to a control point indicates that the control point is free. Secondly, the priority of the job must be the highest among all jobs in the input queue. Thirdly, the amount of memory required by the job (as specified on the job card) must be available. Figure 2.7 is a flow chart of the scheduling process.

The actual scheduling is performed and controlled by the PP routine 1RA, the resource allocator (15). 1RA also ages the input and output queues. 1RA is loaded into a peripheral processor every 250 milliseconds. Each time it is loaded, it examines all control points looking for one which has the dummy file NEXT assigned to it. If no "NEXT" control points are present and if the input and output queues are not ready to be aged, 1RA releases control for another 250 milliseconds.



START
1RA
Control points examined for "NEXT" control point
Free control point found
NO
YES
1RA
Age input and output queues if necessary
1RA idles for 250 msec.
1RA
Age input and output queues if necessary.
1RA idles for 250 msec.
Scan input queue for most eligible *See Note 1
1RA
Control point flag set to ready (or wait) state.
Job now considered to be in CPU queue.
Most eligible job found
NO
YES
1RA
Request memory for selected job
1RA
Control cards read into CM. Reference address, field length, entry point sent to exchange jump area.
NO
YES
CM storage available
NO
fixed priority job ?
YES
Send message to operator
YES
A job rolled out by operator?
YES
CM compacted to make room?

Note 1.
Selection of the most eligible job is based on the availability of enough CM space for the job, and the priority of the job being the highest among jobs for which memory is available.

FIGURE 2.7 CM STORAGE ALLOCATION (SCHEDULING)

If an available control point is found, 1RA then examines all jobs in the input queue to determine which job is most eligible to be scheduled. It will either choose the fixed priority job with the highest priority or, in the event there are no fixed priority jobs, the highest priority job which will fit into the available main memory. In the event two jobs have the same priority, the system chooses the job with the higher main memory requirement.

With an available control point and a selected job, 1RA next allocates the control point and requests main memory for the chosen job. It is possible that sufficient main memory may not be available to honor the request; the use of memory may have changed since the last time 1RA checked it. In this case, one of two things will take place. If the selected job has a non-fixed priority, it will be returned to the input queue and another job will be selected. However, if the job has a fixed priority, 1RA will not attempt to find any other jobs for any control points until the selected job can be scheduled. When this condition arises, a message is sent to the operator giving him the option to roll out a running job, or to ignore the message and wait until enough jobs terminate of their own accord to free sufficient main memory for the selected job.

Once the required central memory becomes available, PP routine 1RA performs all the housekeeping essential to start the execution of the job.

When the job is ready for execution, 1RA reassigns itself to the control point zero and continues to look for more "NEXT' control points and identify the next most eligible job in the input queue.

At any given instant of time, each control point is in one of three states: zero status, indicating that it has no need for the CPU; recall status, indicating that it is waiting for a peripheral operation to complete; and the wait status, indicating that it has need for the CPU and is either waiting for it or currently has control of it. The monitor routine MTR always assigns the CPU to the control point with the highest priority and also in the wait (i.e., ready) state. Control point priority sublevels are recalculated by MTR as discussed previously.

A batch job assigned to the central processor continues executing until one of these events occurs: 1) it requests an I/O operation, or 2) it terminates normally or abnormally, or 3) it is rolled out by a higher priority job. In all three cases the CPU is assigned to the highest priority waiting job.

The switching of the CPU from job to job is facilitated by the exchange jump package mentioned in the hardware section of this thesis (Section 2.2.2). Each control point, whether active or not, has 200 octal words reserved for the exchange package. The first 200 locations of every control point contain the exchange package. The exchange package contains the contents of the 8 address registers, the 8 increment registers, and the 8 data registers of the CPU at the time of interruption. It also contains the program address, the reference address, the field length, the monitor address, and other control information. In order to assign the CPU to a control point, the monitor merely has to fill all the actual machine registers with the appropriate values contained in that control point's exchange package.

If no control point is in the wait state when MTR is prepared to

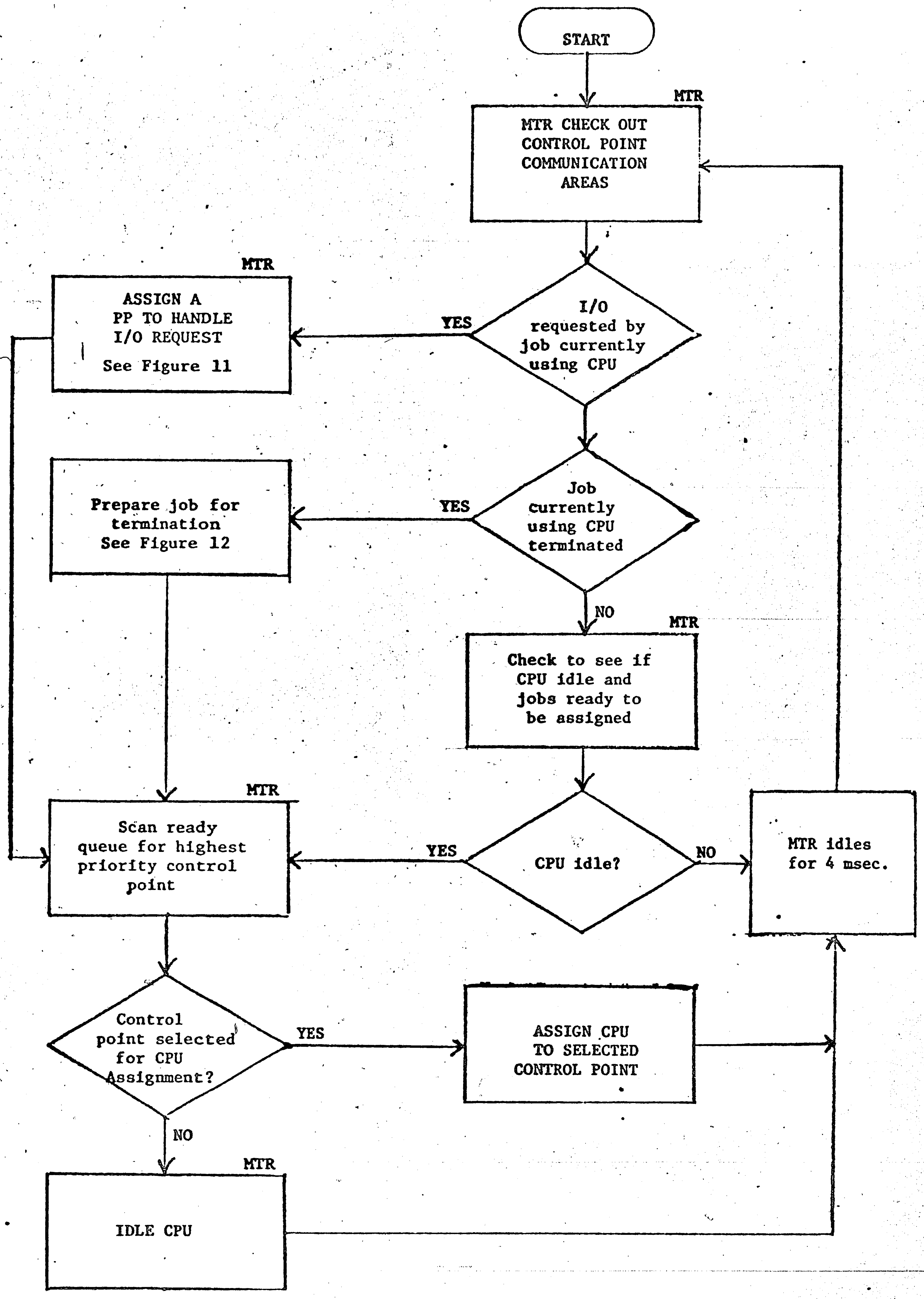

START
MTR
MTR CHECK OUT CONTROL POINT COMMUNICATION AREAS
I/O requested by job currently using CPU
YES
MTR
ASSIGN A PP TO HANDLE I/O REQUEST See Figure 11
Job currently using CPU terminated
YES
Prepare job for termination See Figure 12
NO
MTR
Check to see if CPU idle and jobs ready to be assigned
CPU idle?
YES
NO
MTR
Scan ready queue for highest priority control point
MTR idles for 4 msec.
Control point selected for CPU Assignment?
YES
ASSIGN CPU TO SELECTED CONTROL POINT
NO
MTR
IDLE CPU

FIGURE 2.8 DISPATCHING AND EXECUTION

dispatch the CPU, the CPU becomes idle. MTR will continue to scan the peripheral processor and control point communication areas, in search of any activity requiring attention. It also continues to advance the clock and recalculate priority sublevels as required. Figure 2.8 is a flow chart of the dispatching and execution process.

The execution of a job which has control of the CPU should now be considered. A user job is usually written in a high level language such as FORTRAN or in an assembly language such as COMPASS. This symbolic source program has to be converted into absolute machine language code before actual execution of the job can take place. Typical steps which may occur in this conversion process are:

1) Macro Expansion - The complete text of the source program is created by the substitution of a program text in place of macro calls.
2) Compilation (or Assembly) - The symbolic source text is translated into object modules of machine language code. This process is called assembly if the source text is in assembly language such as COMPASS. The compilation step creates a binary file (machine language code) on mass storage for the user job; it is ready to be linked with other binary files and eventually loaded into main memory.
3) Linkage Editing - A number of separately compiled object modules, and any required library routines, are merged together to form a single program or load module.
4) Loading - The load module is brought into central memory as absolute machine language code.

A specific job may require one (in the case of a program already in the absolute machine code) or more of these job steps. The actual execution of the loaded absolute machine code program is also referred to as a job step.

The control card after the job card causes the first job step to be executed. In the case of compilation, for example, a copy of the compiler is brought to the control point assigned to the job. The compiler utilizes the source program to create a binary file on disk. After completion of that step, the next control card starts the next job step, and so forth until there are no more control statements for the job; the job is terminated.

A job may request a disk I/O operation any time during its execution. All disk requests are entered into a stack, the disk stack. The processing of I/O requests is handled by a group of programs known collectively as the stack processor. In selecting a request from the stack, the stack processor looks for the request that requires the smallest positioning time from the device's current position. The stack processor attempts to minimize the total amount of time devoted to device positioning. The stack processor also uses a priority scheme to ensure that a request involving a large positioning time will not be bypassed indefinitely in favor of requests with smaller positioning times. Figure 9 shows how disk requests are handled.

During execution of a job, a file called OUTPUT is created on disk for the output of the job. Another file, PUNCH, is created for any cards that have to be punched. At job termination, this file can be sent to the appropriate output devices (e.g., card punch, printer).

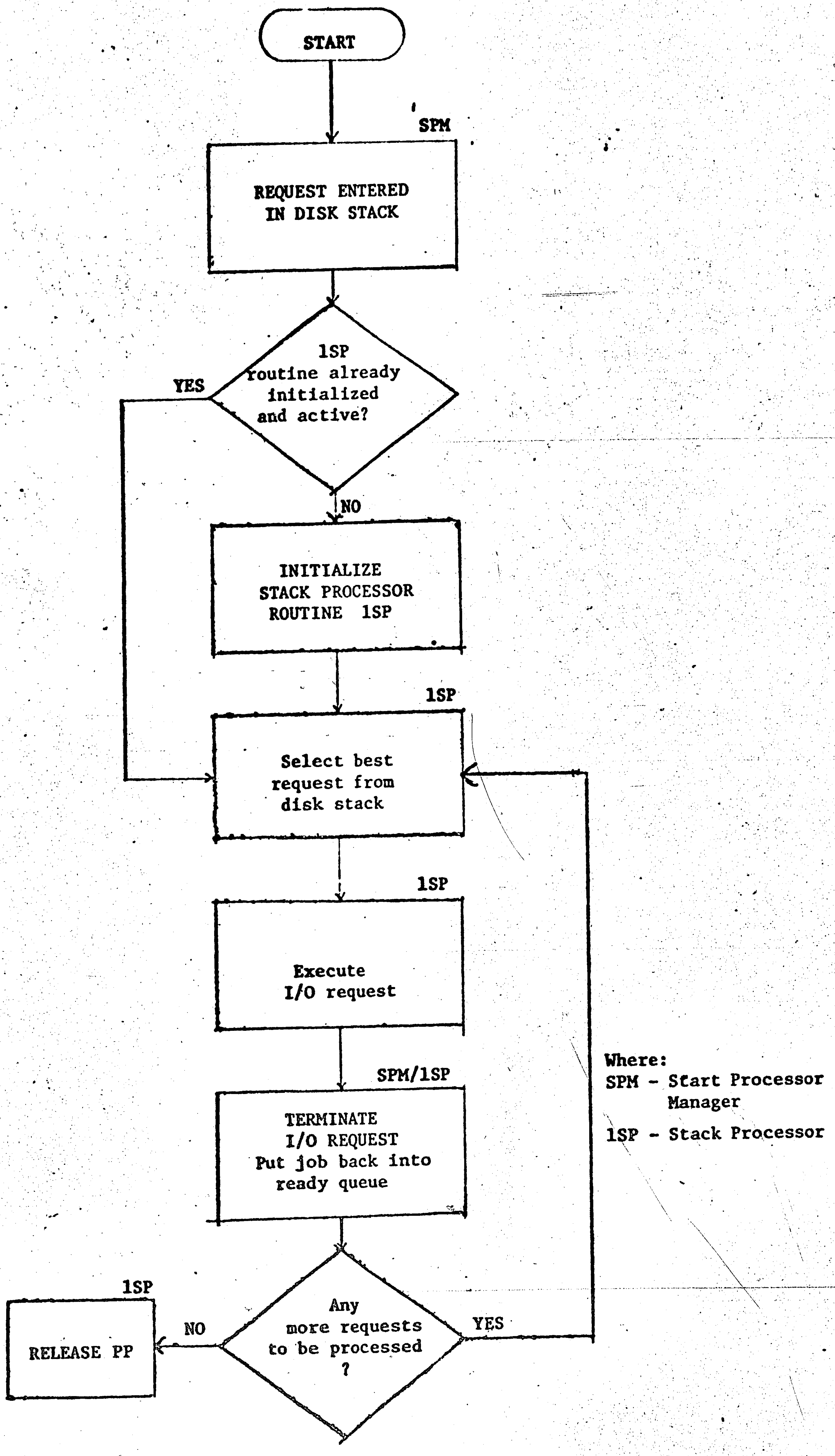

START
SPM
REQUEST ENTERED IN DISK STACK
1SP routine already initialized and active?
YES
NO
INITIALIZE STACK PROCESSOR ROUTINE 1SP
1SP
Select best request from disk stack
1SP
Execute I/O request
SPM/1SP
TERMINATE I/O REQUEST Put job back into ready queue
1SP
RELEASE PP
NO
Any more requests to be processed ?
YES
Where:
SPM - Start Processor Manager
1SP - Stack Processor

FIGURE 2.9 PROCESSING OF DISK I/O

Figure 2.10 shows how a job output is processed after its completion. The PP routine 1AJ checks the program for normal or abnormal termination. If an abnormal end has occurred, routine 1AJ sets the proper flags and calls a routine (DMP) to provide information for debugging; routine 1AJ closes all files for the job. If a normal end of program has occurred, 1AJ checks all error flags. Then 1AJ calls routine 2DF to close all files and enter them into the output queue; next it deallocates all the files and equipment (e.g., tape drives) still assigned to the job. Routine 2DF completes the Day File, clears the control point area and terminates itself.

The PP routine 1ØT is called to begin disposing of the OUTPUT and the PUNCH files. If cards have to be punched, 1ØT calls routine CIØ which reads the PUNCH file and turns it over to routine 1PØ; 1PØ drives the card punch. If printing is necessary (and it is most of the time), 1ØT calls CIØ which reads the OUTPUT file and turns it over to 3ØT which drives the line printer.

The assignment of OUTPUT and PUNCH files to the output devices is handled in much the same way as the assignment of input queue jobs for CM allocation, i.e., by priority. Priorities in the output queue are incremented every 64 seconds to insure that every job is eventually processed. After being copied by the output device(s), the OUTPUT and PUNCH files for the job are deleted from the system.

2.3.6 Processing of Interactive Jobs

The processing of interactive or time sharing jobs in the CDC 6400 computer is controlled by the SCOPE operating system in conjunction with

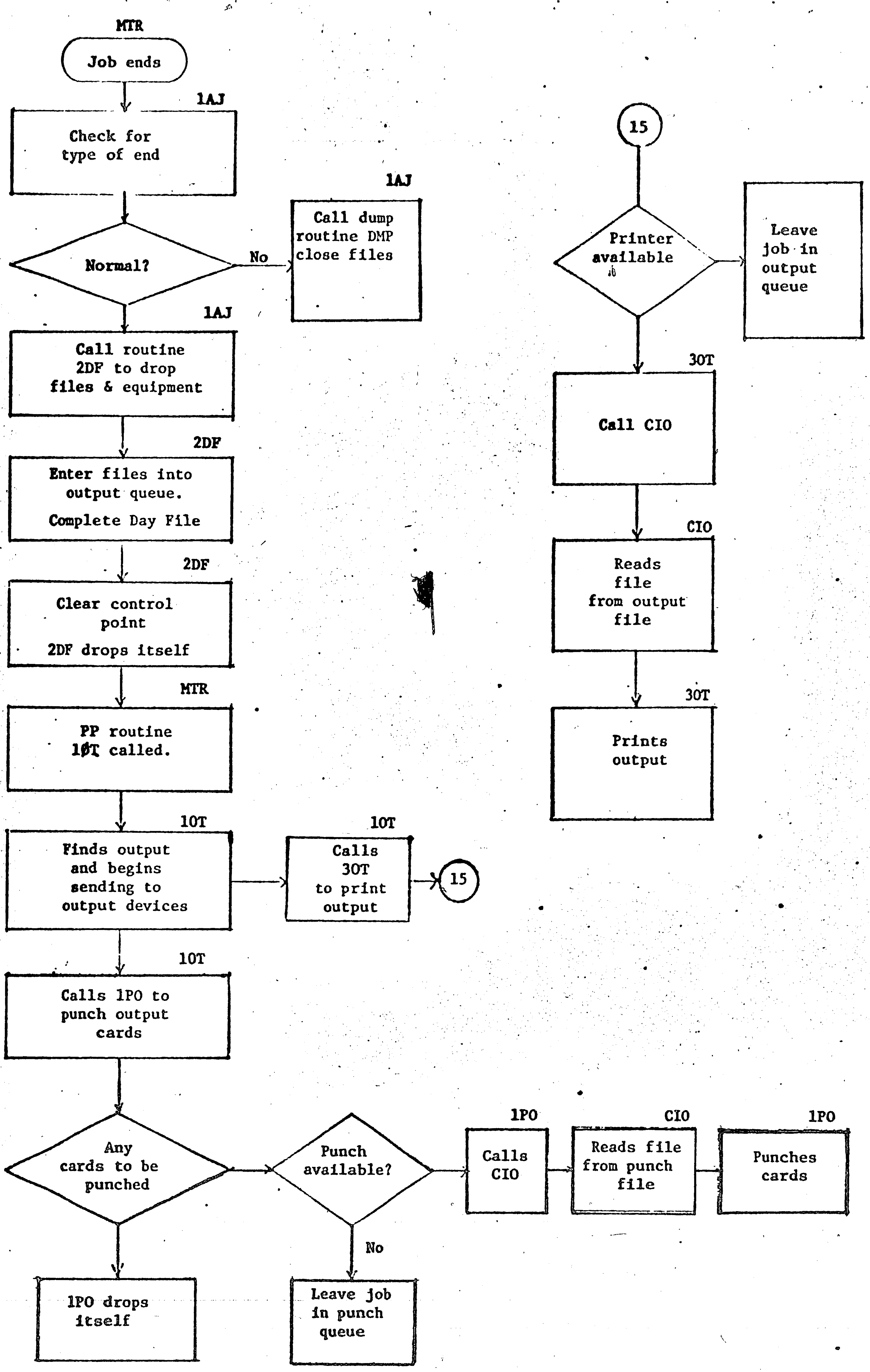

MTR
Job ends
1AJ
Check for type of end
1AJ
Call dump routine DMP close files
Normal?
No
1AJ
Call routine 2DF to drop files & equipment
2DF
Enter files into output queue. Complete Day File
2DF
Clear control point 2DF drops itself
MTR
PP routine 1~~0~~T called.
1OT
Finds output and begins sending to output devices
1OT
Calls 3OT to print output
15
1OT
Calls 1PO to punch output cards
Any cards to be punched
Punch available?
No
1PO drops itself
Leave job in punch queue
1PO
Calls CIO
CIO
Reads file from punch file
1PO
Punches cards
15
Printer available
Leave job in output queue
3OT
Call CIO
CIO
Reads file from output file
3OT
Prints output

FIGURE 2.10 JOB TERMINATION AND OUTPUT PROCESSING

a group of programs called INTERCOM. INTERCOM is a general purpose file-based system which provides the user with facilities for file input, file manipulation, temporary and permanent file storage, editing, compilation, and conversational and non-conversational execution. INTERCOM has a delayed output facility that enables a user to request that the output of a non-conversational job be stored on a special file. The output can be retrieved at a later time. INTERCOM also permits the submission of batch jobs from remote locations (remote job entry). These facilities enable a user to submit a wide variety of jobs from remote locations.

As a system of programs, INTERCOM has five major parts. These are the multiplexor driver, the scheduler, the quantum calculator/MUJ servicer, the buffer manager/file processor, and the utility programs.

The multiplexor driver is a PP program which resides permanently in a peripheral processor during the time INTERCOM is running. The multiplexor driver can service up to two multiplexors on a single data channel.

The scheduler is responsible for the initiation of the execution of all commands, and the assignment of INTERCOM control points to user jobs. The scheduler is also called to swap out a job which has used up its quantum. The INTERCOM scheduler is always in competition with the batch scheduler. The INTERCOM scheduler has priority over the batch scheduler.

The quantum calculator/MUJ servicer is responsible for calling the main scheduler routine 1SJ to swap out users who have exceeded their quantums. The quantum calculator/MUJ servicer, which operates at control point zero, also handles the communication between multi-user jobs, such as the EDITOR, and the INTERCOM system. It also schedules multi-user-jobs [MUJ's]. MUJ's are programs which can be used simultaneously by several users on a time-shared basis.

The buffer manager/file processor is responsible for the management of the INTERCOM buffer area, the initiation of users entering the system, and the transmitting of data files to and from user terminals.

The utility programs are routines which carry out INTERCOM commands specified by the user.

Communication between the user and the INTERCOM system is in the form of commands. A command causes either a system program or a user program stored on disk to be activated. The program has to share the central memory and the central processor on a time-sharing basis with programs initiated by other INTERCOM users as well as with the background batch jobs.

2.4 SYSTEM HARDWARE AND SOFTWARE EVALUATION

The CDC 6400 and the other computer systems in the CDC 6000 series are fast multi-computer systems designed mainly for the scientific environment. These systems are simple and highly modular in structure. They can be upgraded easily and efficiently. They have some outstanding design features such as the exchange jump package and a hardware relative addressing scheme which aids multiprogramming. The use of peripheral processors to perform system functions virtually eliminates CPU overhead.

The CDC 6000 series computers do have both hardware and software limitations. These problems are discussed next, beginning with hardware limitations.

2.4.1 Hardware Limitations

The CDC 6000 series computers do have several hardware limitations. One serious limitation is that there are no hardware interrupts. This results in delays in the recognition of the completion of tasks and causes an increase in system overhead time. A potentially serious problem that can result from the lack of interrupts is a peripheral processor run-away. Peripheral processors are supposed to notify the monitor when they complete an assigned task. If, because of software errors, a PP fails to notify the monitor about the completion of a task, then the PP is lost until the next system deadstart.

The central memory is not directly accessible to I/O devices. All data transferred from the disks and the peripheral devices have to be buffered through the small core of a peripheral processor. This imposes a limit of 5000 characters on the size of a record that can be transferred at a time. It also makes it difficult to transfer data at a high rate of speed.

The internal representation of data in the Binary Coded Decimal (BCD) format differs from the standard BCD format used for magnetic tapes. Data in the standard BCD format must be converted by a peripheral processor to the internal display code format. Not only does this place unnecessary burdens on the peripheral processors, it also reduces the rate of data transmission.

Data transfers between central memory and the processors (CPU, PP's) are made via a single path, the data distributor. The limit of 4 on the number of words that can be transferred at a time can result in memory reference conflicts and a degradation of the performance of the system.

Another serious limitation is the limit of 10 on the number of peripheral processors that can be accommodated on the system. A number of peripheral processors are usually involved in processing a request for a user job. For example, at least two PP's are involved in reading a deck of cards for a job; one reads the cards into a CM buffer, another stores the contents of the CM buffer on disk. During heavy system usage, the peripheral processor pool can become the main system bottleneck. The peripheral processor situation can become very acute in a system where some of the PP's are assigned for extended periods of time to such tasks as telecommunications and file auditing. The use of extended core storage (ECS) will eliminate some PP tasks and partially alleviate the PP problem.

A feature not available in the CDC 6000 series computers is core parity checking. The number of errors resulting directly from the failure to check the parity of memory is not known. It can be said, however, that such errors can occur without detection.

2.4.2 Software Limitations

A potential problem of serious consequence for the Chippewa Operating System and, therefore of SCOPE, is high priority degradation. An article written by Dave Stevens (47) discussed the symptoms of high priority degradation and how special scheduling algorithms can be used to avoid it. High priority degradation appears when one or more control points are occupied by high priority jobs requiring more central memory than is currently available. The high priority jobs are unable to execute because they cannot get any core, and the smaller low priority jobs in the input queue are unable to reach a control point. High priority degradation can also result from the hogging of the central processor by a high priority, computer-bound job while other I/O bound jobs are waiting to execute.

One solution to this problem suggested by Stevens was the dynamic assignment of priorities based on the degree to which the job is I/O bound, rather than the amount of compute time since the last recomputation. He also suggested a reduction of operator interaction.

In the SCOPE system, tape assignments are not made until the programs have been assigned control points and have been loaded in the central memory. The control point, therefore, remains idle while the operator is mounting the required tapes. The system should require all system resources to be available before the program is loaded into central memory. The Lehigh installation controls this deficiency by allowing the operators to read tape jobs only if enough tape drives are available. The operators are required to mount the tapes before the jobs are read.

This chapter has discussed the hardware and software of the CDC 6400 computer system. The outstanding features of the system and the hardware and software limitations have also been discussed. The next chapter describes the simulation model which was developed to imitate the system. The design of the simulation model and the implementation of the model are discussed.

CHAPTER 3

SIMULATION MODEL OF THE CDC 6400

3.1 INTRODUCTION

The primary objective of this project was to develop a reasonably valid simulation model which could be used to predict the performance of the CDC 6400 computer system. The secondary objective was to use the model to study the effect of changes in configuration, workload and resource allocation algorithms on the performance of the system. The key measures of performance selected for this study were job turnaround time, interactive user response time, job throughput, equipment utilization, and control point dwell. Control point dwell is the total length of time a job resides at a control point in central memory.

A critical issue in the development of a simulation model for a computer system is the level of detail at which the model is to operate. The level of detail is crucial because not only does it determine the execution speed of the model, but it also determines the kind of data that is needed to run the model. Two contrasting levels of detail are frequently considered; the extremely detailed level is known as the micro level, and the extremely aggregate level is referred to as the macro level.

At the micro level, the effects of individual machine-language or source-language instructions are simulated. The units of work or transactions identified in a micro level model may be instructions. Such minutiae as channel speeds, sizes of instruction registers, buffer sizes, memory cycle time, and single instruction execution times are of vital importance in a micro level model. Micro level models require large amounts of computer time. They usually operate, at best, two or three times faster than the actual system. This means that to simulate an hour of system operation an analyst has to run his model for about 30 minutes.

Micro level models are very useful in the study of hardware and software design problems. They can be used to study the effect of hardware and software modifications. For the measures of performance selected for this study, however, the micro level of detail is inappropriate.

At the macro level, the processing of complete jobs is simulated. The unit of work or transaction identified in this model is the total job. The macro level model ignores the job steps or tasks within the job. It also ignores the individual hardware components. The macro level model is fast in terms of computer running time. The macro level as defined here was rejected for this study because the resulting model would not have been realistic compared to the actual system. Dynamic field length changes during program execution, which are a very important aspect of the LUCC system, would have been ignored in a macro level model.

The level of detail chosen was a compromise between the two extremes. A fairly complex model of interactive and batch job flows was used, so that important measures such as interactive user response time, the number and duration of roll-outs, turnaround time, etc.

could be determined. Such minutiae as the 4 msec. cycle time of the monitor (MTR), positioning time in disk access, the 100 nanosecond memory cycle time, etc., were either ignored completely or were incorporated in grosser measures.

A model written at this level of detail runs fairly fast, usually about 50 times faster than the actual system. Most of the information needed to run such a model can be obtained from the daily accounting file known as the "Day File." The Day File accumulates data on individual jobs, such as the number of cards read, the amount of central processor and peripheral processor times used by the job, the total time a job spends at a control point (referred to in this thesis as control point dwell), the amount of time the job spends in the various queues, and the total time the job spends in the system.

The next section discusses the characteristics of the workload at the Lehigh University Computer Center.

3.2 CHARACTERISTICS OF THE WORKLOAD

The performance of a computer system depends not only on the hardware and the operating system, but also on the workload processed by the system. The simulation of a computer system involves the development of two models: a model of the computer system itself, and a model of the workload. The workload model provides the input to the system model.

Two main approaches used in characterizing workloads are the "trace-driven" approach and the "statistically-driven" approach. In the trace-driven approach, attributes of actual jobs are monitored and stored on tape or on disk. This raw data is used to drive the system model. In the statistically driven approach, job attributes and system service parameters are generated from statistical distributions. The statistical distributions are either empirical distributions or statistical approximations to the empirical distributions. The job attributes generated from the distributions are used to drive the system model.

The statistically-driven approach was used for a number of reasons. This analyst was not a member of the Systems Programming group at the Computer Center, and had no authority to insert software and hardware probes in the system. These probes are necessary for the creation of trace data tapes or files. The major reason for the selection of the statistically-driven approach was that, at the time the project was started, the Computer Center was working on a Day File analysis program. This program, developed by Richard Cichelli of the LUCC Systems Group, analizes the Day File and generates important statistics on the workload for each day. These statistics are contained in an internal report known as the "LUCC Daily Operating Statistics" (9). Data from a number of these reports was used to drive the simulation model. Information on the workload contained in this section was also obtained from the daily reports.

The Lehigh University Computer Center (LUCC) environment is a typical university environment. The characteristics of the job mix vary from day to day. Table 3.1 shows statistics for batch jobs on six selected days at LUCC. A number of observations can be made from the table.

TABLE 3.1

STATISTICS FOR BATCH JOBS ON SIX SELECTED DAYS

	DATE					
	1 3/13/73	2 3/15/73	3 3/20/73	4 3/23/73	5 3/26/73	6 4/3/73
No. Jobs	1586	1419	772	872	1318	1482
% of Student Jobs	60.0%	65.0%	30.3%	36.3%	58.5%	59.3%
Mean CPU Time (sec)	10.48	8.55	19.44	13.28	10.37	9.08
Mean PP Time (sec)	11.82	7.83	36.8	18.3	15.85	18.17
Mean CPU Space (decimal)	20,332.7	21,585.8	16,980.9	16,079.9	17,234.7	20,489.9
Mean Print Lines	837.6	671.48	1709.6	994.1	827.68	928.7
Mean Punched Cards	247.4	370.18	284.6	132.2	240.92	355.9
Time in System (sec)	216.24	1083.1	581.37	398.32	314.93	384.4
Control Point Dwell (sec)	56.80	47.88	154.3	94.46	66.64	81.52

First of all, in terms of the number of jobs processed by the system and the mean values of the various job characteristics, no two days are alike. Some of the data in Table 3.1 (data sets 3 and 4) was collected during a vacation. The average number of jobs per day dropped from about 1500 during the regular semester to about 800 during the break. The percentage of instructional jobs dropped from 60 percent to about 33 percent. The total number of hours of system operation is usually from 14 to 12. The system usually does not operate trouble-free. Close examination of Table 3.1 reveals an abnormality on the 15th of March; the average time in the system was 1083.1 seconds while the average control point dwell was 47.9 seconds. This abnormality could be due to a printer failure. Incidentally, the average time in the system generated by the Day File analysis program is not, as one would expect, the turnaround time; it does not include waiting time in the input queue (see Table 3.1 - time in system).

Most of the jobs submitted at LUCC can be put into one of two categories: instructional and non-instructional jobs. Instructional jobs are those submitted by students to fulfill course requirements. These are generally small jobs which require small to moderate amounts of peripheral processor and central processor time. An average of 4 seconds of CPU time for this class of jobs is common for most days.

Non-instructional jobs include sponsored and unsponsored research jobs, administrative jobs, jobs from a few industrial users, and jobs submitted by the staff of LUCC. The characteristics of the jobs in this class vary widely; the average CPU time for this class is 20 seconds. Statistics collected on March 26, 1973 are broken down into these two job classes and illustrated by Table 3.2. Table 3.2 shows the mean values of some of the job characteristics for each class and for the overall job mix.

TABLE 3.2

JOB STATISTICS FOR THE TWO CLASSES OF JOBS*

Variable	Overall Mean	Instructional Mean	Non-Instructional Mean
CPU Time (sec)	10.3	3.6	20.5
PP Time (sec)	15.8	6.6	29.0
CM Space (decimal words)	17,234	15,000	21,000
Lines Printed	827	493	1,300
Cards Punched	240.9	169.0	342.0
No. of Jobs	1,318	774	544
Control Point Dwell (sec)	66.6	28.9	119.5

*Data was collected on 3/26/73

Statistics on INTERCOM usage are also provided by the Day File analysis report. Statistics on INTERCOM usage for six selected days are presented in Table 3.3. These six days are the ones whose batch job characteristics were presented in Table 3.1

TABLE 3.3

STATISTICS ON INTERCOM USAGE

	3/13/73	3/15/73	3/20/73	3/23/73	3/26/73	4/3/73
Total # of sessions	175	202	143	141	213	228
Mean CPU Time (sec)	4.44	5.33	3.74	4.76	3.79	7.21
Mean PP Time (sec)	20.19	19.39	24.12	23.10	19.57	25.74
Time in System (sec)	133.2	1202.5	1564.0	1327.3	1227.9	1350.7

In a statistically driven simulation model, the statistical distributions of job characteristics are of great importance. The distribution of core size is shown in Figure 3.1. Two distinct peaks can be identified in the distribution, one around 3K decimal words (K = 1024) and another around 11K decimal words.

The distribution of CPU time is shown in Figure 3.2. A single peak value occurs around 0.7 seconds. The average CPU time for batch jobs on that day was 9.2 seconds. The distribution is very positively skewed (suggesting an exponential or gamma distribution). Several previous studies have been conducted, indicating that, generally, the distribution of CPU time per job can be approximated by an exponential distribution (17, 25). An exponential distribution was generated to represent the distribution of the CPU time, using the LEAPS program. The curve fitting feature of LEAPS (Lehigh Amalgamated Package for Statistics) uses the least squares polynomial method to fit a curve to a set of data. The cumulative frequency curves for the empirical distribution and for the exponential approximation are shown in Figure 3.3. The Kolmogorov-Smirnow goodness-of-fit test was used to test the degree of agreement between the empirical distribution and the generated exponential distribution. The maximum deviation D obtained for the test was 0.1011. With a selected level of significance (α) of 0.05 and a sample size of 120, the critical value of D is 0.1242. Since D is 0.1242, we must conclude that the exponential distribution is a satisfactory fit.

The Erlang distribution is a very useful distribution. The Erlang

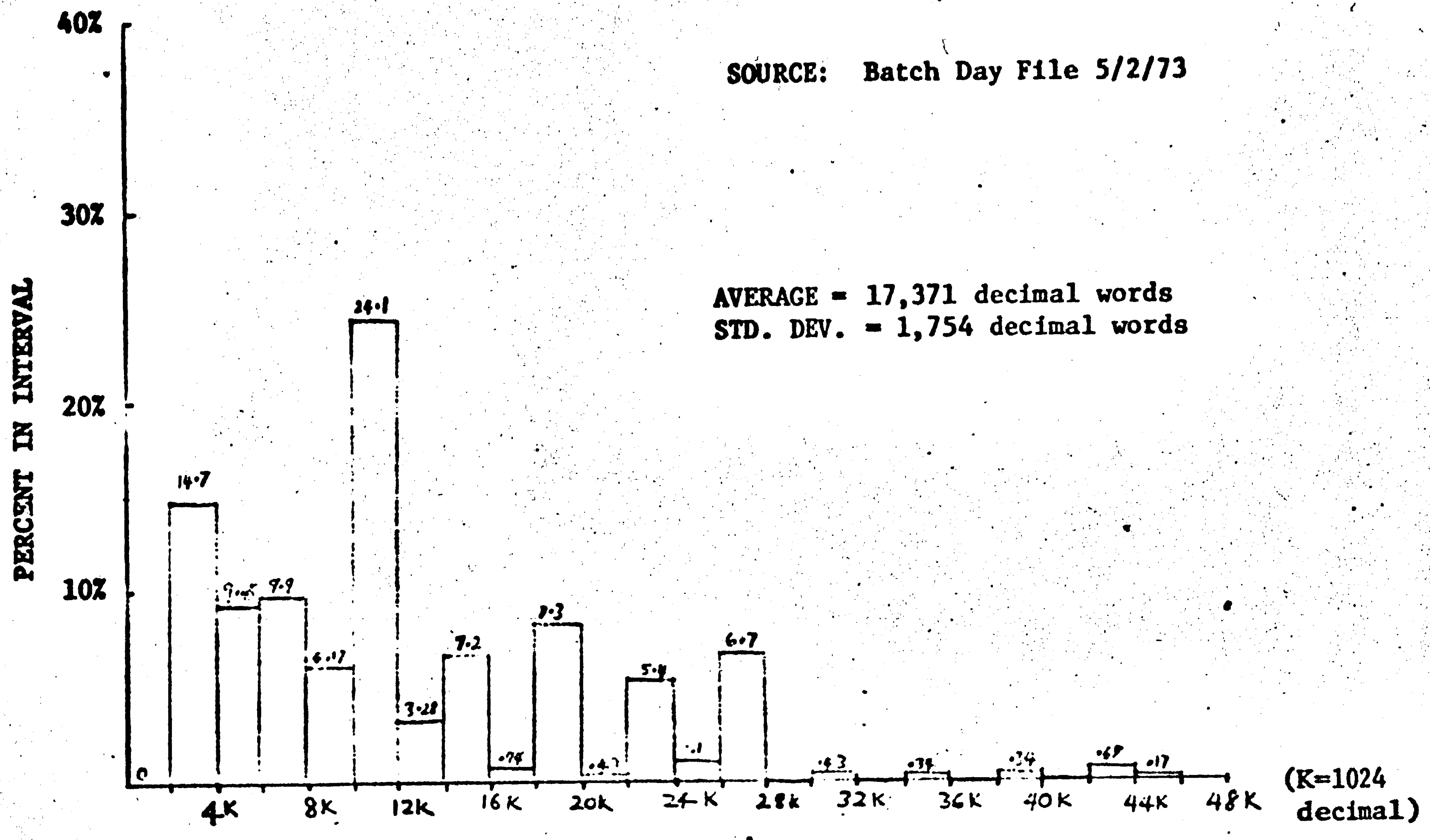


FIGURE 3.1 DISTRIBUTION OF CM CORE SIZE DEMAND PER JOB

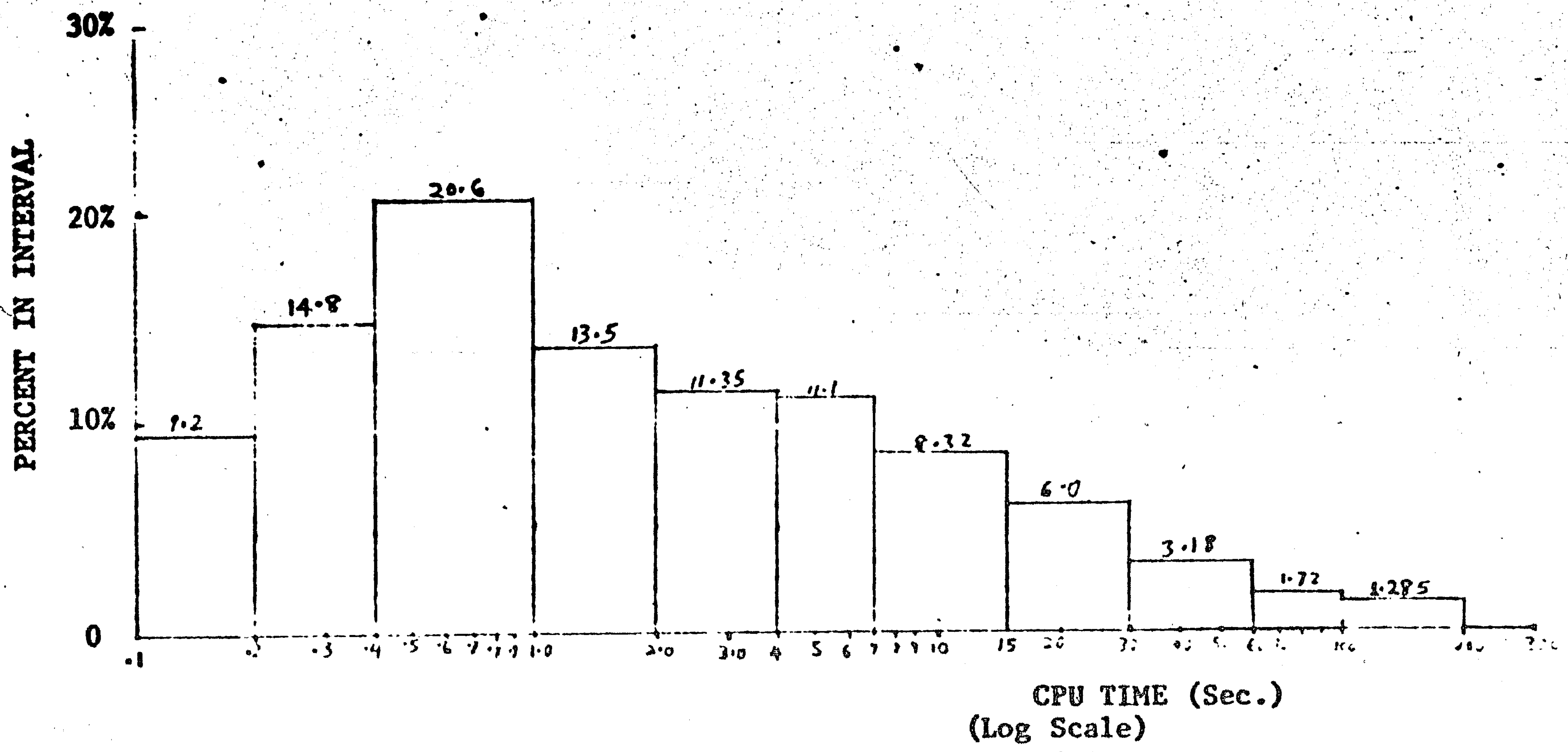


FIGURE 3.2 DISTRIBUTION OF CPU TIME DEMAND PER JOB

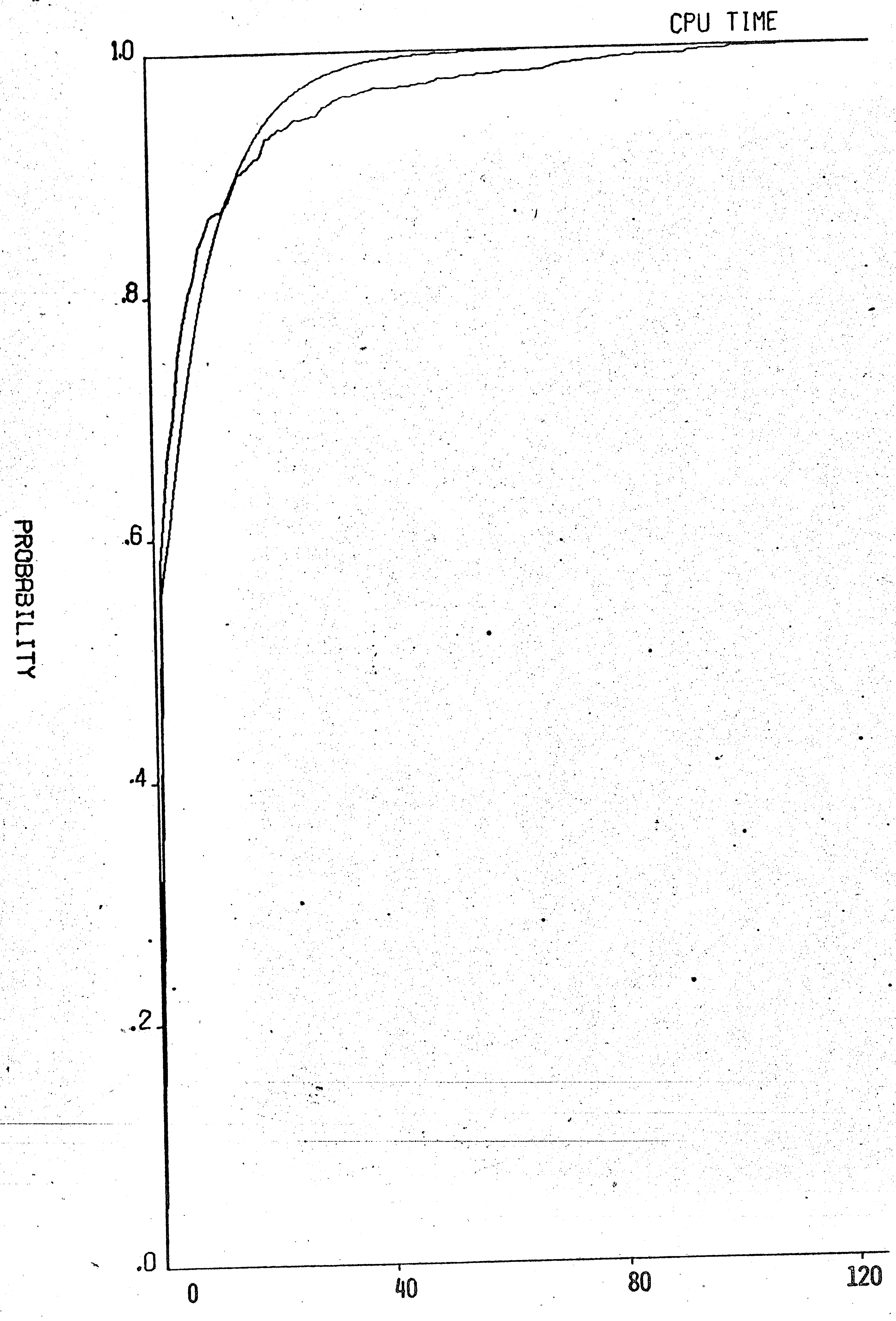


FIGURE 3.3 CUMULATIVE DISTRIBUTION OF CPU TIME

distribution, which is a special case of the Gamma distribution with an integer parameter, can represent several other distributions. The Erlang distribution reduces to the exponential distribution when the parameter K is equal to one. The Erlang also approximates the normal distribution when K is large. Thus, distributions of varying skewness can be generated by simply varying the K parameter of the Erlang.

The Erlang distribution was used in the model to generate several job characteristics. The generation of job characteristics is discussed in section 3.3.3. The next section deals with the design and structure of the simulation model.

3.3 DESIGN AND STRUCTURE OF THE SIMULATION MODEL

In developing the simulation model at the specified level of detail, a number of assumptions and simplifications had to be made. The first assumption was that the operation of the system was fully automatic, and that no operator intervention occurred. This assumption was made because the behavior of an operator is not fully predictable. The data used as input to the model and also as a means of validating the model was collected under conditions other than the fully automatic conditions.

A continuous operation of the system was assumed. The system frequently operates without any malfunction; therefore, this is not an unreasonable assumption. Another assumption made was that no system overhead was performed by the central processor. All the system overhead activities were performed by the peripheral processors. The time it takes to switch the CPU from one program to another was assumed to be negligible. This assumption is not unreasonable since the system is organized in such a way that the CPU performs a negligible amount of overhead.

In order for the model to serve as a tool for investigating the behavior of this complex multiprogramming time-sharing system, all the important features of the system were modeled. Some of the more important features in the model are discussed next.

3.3.1 Features of the Model

Several features were incorporated in the model to make the model a realistic representation of the real system.

A significant feature of this simulation model was the consolidation of both the system's batch and interactive operations into a single model. This is one difference between this model and prior simulation models reported in the literature. Most prior simulation studies have dealt with either the batch or the interactive aspect of the system's operation; they have not combined both aspects. The interaction between batch and interactive jobs in the CDC 6400 is too important to be ignored.

A result of the interaction between batch and interactive jobs is the roll-out of batch jobs by higher priority interactive jobs. The roll-out of a batch job can also occur when a batch job requests more memory during execution and sufficient memory is unavailable. The roll-out/roll-in feature of the system was incorporated in the model.

A special feature built into the operating system at the Lehigh installation is a facility that can be used by the operator to specify the minimum amount of time a batch job must have resided at a control point before it can be rolled out by a higher priority interactive job. This minimum dwell time is an input parameter to the model.

Time-slicing for interactive jobs was also incorporated in the model. The time-slice can either be specified at the beginning of the simulation run, or it can be computed for each job during the run.

In the SCOPE system at LUCC, the operator can specify preferred job classes for different times of the day. Thus, the operator may specify that only jobs whose CM core requirements are less than 70K (octal) words and whose CPU time limits are less than 200 seconds may be processed between 9 a.m. and 12 noon. Since this feature can significantly affect overall performance of the system, it was incorporated in the model.

There is a facility in the model for assigning control points to INTERCOM jobs, express batch jobs, normal batch jobs, and system routines such as JANUS. The control point assignments are part of the input data to the model. The ability to make different control point assignments in the model permits the effect of different assignments to be studied.

The effects of modifications in scheduling and dispatching algorithms can also be studied very easily because of the modularity of the program. Key algorithms are relatively isolated from the rest of the coding so that changes can easily be programmed and incorporated. The workload and the system configuration can be specified and modified very easily.

Before the structure of the simulation model is presented, the representation of batch, remote batch, and interactive jobs, and the generation of their attributes are discussed.

3.3.2 Job Representation

A time-sharing user session is defined as the time interval between a user log-in and log-out. During this time interval, the user may issue several commands to the system and wait for the response of the system to each of his commands. Each command begins a different cycle in the user/system interaction. The basic cycle of user/system interaction entails the user issuing a command, waiting for a response from the system, and reviewing the response; it is known as a transaction. A user session consists of a series of tasks, and each task in turn consists of a series of transactions.

A task is defined as a series of transactions initiated by commands of the same type. Five types of tasks are distinguished in the model. These are:

1) File manipulation
2) Program input and editing
3) Program running and debugging
4) Program compilation and assembly
5) Miscellaneous

File manipulation involves the printing of the contents of a file, the combination of files, the splitting of a file, the deletion of files, the copying of files, and similar operations. The file manipulation task usually consists of a single transaction and requires little CPU time. The program input and editing task involves the generation and modification of disk files that contain the source program written by the user. The program input and editing task usually consists of several transactions and very little CPU time. In the program running and debugging task, binary-coded files are loaded, linked and executed. The program running task involves a moderate number of transactions and a comparatively large CPU time per transaction. The fourth type of task is that of program compilation or assembly. In the compilation task, machine language files are created from source language files. This task involves one transaction and a relatively large amount of CPU time. Commands not falling in any of the above four categories are put in a fifth category, miscellaneous.

A transaction consists of command input, command execution and user evaluation of output. For the purposes of the model, a transaction can be considered as consisting of two states: the "console" state and the "working" state. In the console state, the user is either thinking, typing in an input line or reviewing an output. In the working state, the system has a program from the user and is working on it. A transaction begins with the console state and changes into the working state when the user types in an input line or command.

The duration of the console portion of a transaction depends on the activity performed by the user during the console session. During the console part of a transaction, the user can either be thinking, typing in a command or typing in an input line. During execution, a program may generate another command and cause another program to be loaded. A program-generated command is considered as a new transaction with a zero console portion. An executing program can enter an output-wait condition if the program attempts to add to an already full output buffer. An output-wait condition initiates the console part of the next transaction.

The time-sharing part of the computer system was simulated by tracing the initiation of transactions by users and the transition of transactions from the console state to the working state.

A batch job is represented in the model as a sequence of job-steps or tasks. A job-step is a computation unit to which system resources, such as memory and CPU time, are allocated. In the SCOPE system, job-steps are initiated by control cards. The main types of job-steps or tasks distinguished in the model are compilation or assembly, program execution, and file manipulation. The lowest level of a batch job identified in the model is a job-step or task.

The processing of batch jobs was simulated by tracing the initiation of jobs, the initiation of tasks within jobs, and the flow of tasks from service station to service station through the system. The service stations are input devices such as the card reader, the central processor, disk I/O processors, and output devices such as printers, and card punches.

3.3.3 Job Generation

The processing of a batch job in the model begins with its arrival at the card reader. The average number of arrivals per hour, which is an input parameter, determines the interarrival time. Interarrival times are generated from a statistical distribution which is an input parameter. Interarrival times are usually exponentially distributed; hence, an exponential distribution was used to generate arrivals.

On arrival at the card reader, a number of job characteristics are generated from statistical distributions which represent the workload.

The job characteristics include:

a) the number of cards to be read,

b) the number of cards to be punched,

c) the number of lines to be printed,

d) central memory space required,

e) total CPU execution time,

f) the priority level of the job,

g) the job class (instruction, non-instructional),

h) number of job steps or tasks,

i) job number (assigned sequentially).

All these job characteristics are attached to the job as attributes. The full list of attributes assigned to each job is shown in Appendix A.

Several types of distributions can be used in the model to generate job characteristics: the normal distribution, the exponential distribution, the Poisson distribution, and the Erlang distribution. As mentioned earlier, with the change of a single parameter, the Erlang distribution can be used to represent several distributions of varying skewness.

The modeling of an interactive job begins with the log-in event. The number of arrivals or log-ins per hour is a data input to the model. An exponential interarrival rate is typically assumed. At log-in, a random number is generated to determine whether the job is a remote batch or an interactive job. If the job is a remote batch job, batch job characteristics are generated by the same routine that generates job attributes for normal batch jobs. A class number of 3 is assigned to remote batch jobs.

If the job is interactive, session characteristics such as the total amount of CPU time to be used during the session and the total PP time, are generated from distributions whose parameters are specified at the beginning of the simulation. The nature of the first task is determined by drawing a random number. The number of transactions in the first task is generated from a Poisson distribution whose parameters are input to the model. The console portion of the first transaction is then scheduled. The duration of the console session is generated from a distribution whose parameters are specified in the input data. The generation of transaction attributes such as CPU time and memory requirements occurs when the working part of a transaction begins. The characteristics of a user session and the current transaction are attached to the user job as attributes. Appendix A shows the attributes attached to each job.

3.3.4 <u>The Structure of the Simulator</u>

The simulation program (SIMLUCC) developed in this project was programmed in the GASP II simulation language. GASP II is an event oriented simulation language which provides a convenient structure upon which a wide variety of simulators could be constructed. It consists of a number of specialized routines and an executive routine which schedules events and advances the simulation clock. The specialized routines perform the functions of system state initialization, data collection, information storage and retrieval, statistical computation, random number generation, and statistical process generation.

In an event-oriented simulation system, the model is concerned only with the initiation or the termination of activities or operations. The initiation and termination of operations, which are referred to as events, mark the important transition points in the processing of an item. The life of an item in a system can therefore be thought of as consisting of a series of events, each event corresponding to a transition point in the life of the item.

At the occurrence of each event, the time and identity of the next event for the item are determined. The time of the next event is determined by adding the calculated duration of the next activity to the current simulation clock time. The generated event is then stored in the event file. The event file is a list of events that have been scheduled but which have not already occurred. Items in the event file are ordered according to the time of occurrence, in an ascending order.

When an event occurs, the next event is scheduled only if the facility is available to process the item. If the facility is unavailable, the item is entered in the queue for the facility. When the facility becomes available, it accesses its queue and processes the highest priority item in the queue.

The advancement of an item through the system is determined by a number of attributes associated with the item. In the SIMLUCC program, some of the attributes of items or jobs are generated on arrival. Others are accumulated as the job progresses through the system. Some of the attributes contain flags which are used to determine the path of the job through the system. These flags are necessary because three types of jobs are modeled: normal batch, remote batch, and interactive jobs. Appendix A shows the assignment of attributes.

An item progresses through the model by moving from file to file, until the terminating event occurs. At the occurrence of the terminating event, all necessary statistics are collected, and the item is removed from the system.

This is the basic structure of the model. In regards to file structure, the model consists of one event file and eleven other files. Eight of the files are queues, and the rest are scratch files. File assignments are presented in Appendix A.

Apart from the GASP II routines, the SIMLUCC program consists of a main program and forty-eight subroutines. Some of the subroutines are event routines which are responsible for the advancement of jobs through the system. The other routines are specialized routines which are called by the event routines to perform specific tasks. Some of the tasks performed by these specialized routines are the scheduling of batch jobs, the scheduling of interactive jobs, the dispatching of active jobs, the termination of a currently running job, the rolling-in of a rolled-out job, and the generation of job characteristics.

The advancement of jobs through the model is accomplished by event routines. Figures 3.4 and 3.5 respectively show the flow of batch and INTERCOM jobs in the model. Figures 3.6 and 3.7 respectively show the event routines associated with the advancement of batch and interactive jobs through the model. The flow of jobs through the model will now be described as each job is processed by the appropriate event routines. The flow of batch jobs will first be described. In the

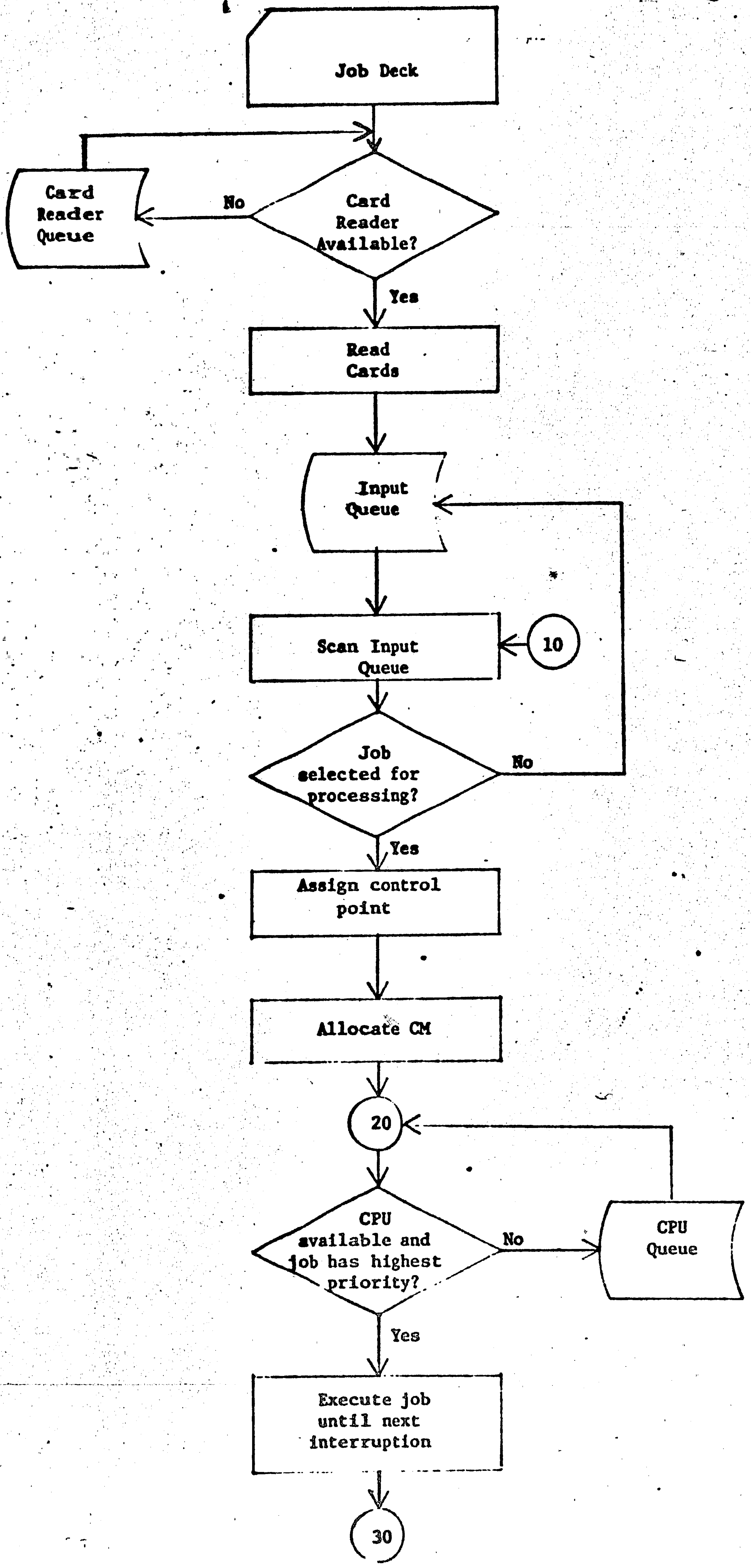


FIGURE 3.4 BATCH JOB FLOW IN THE MODEL

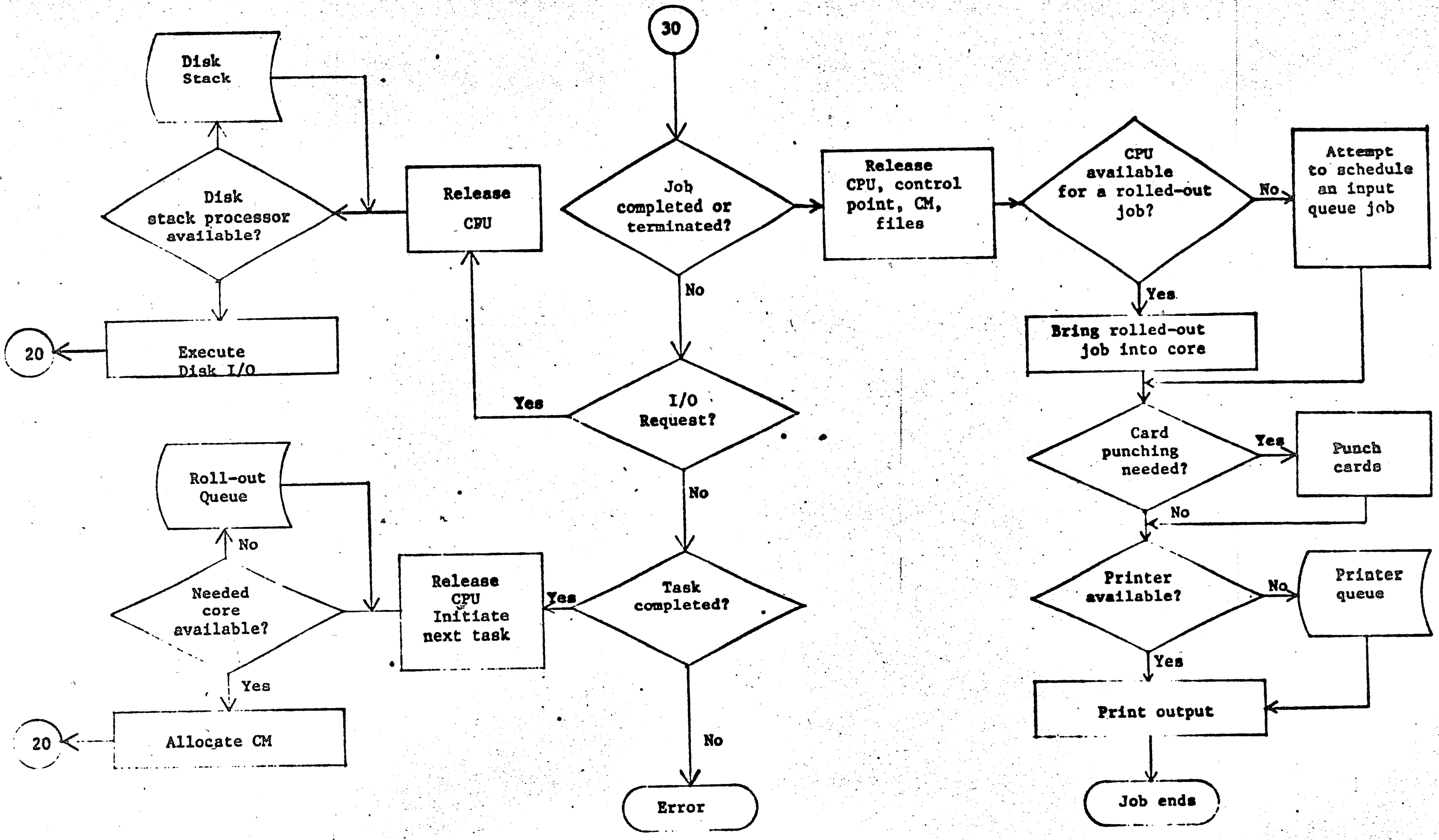


FIGURE 3.4 BATCH JOB FLOW IN THE MODEL

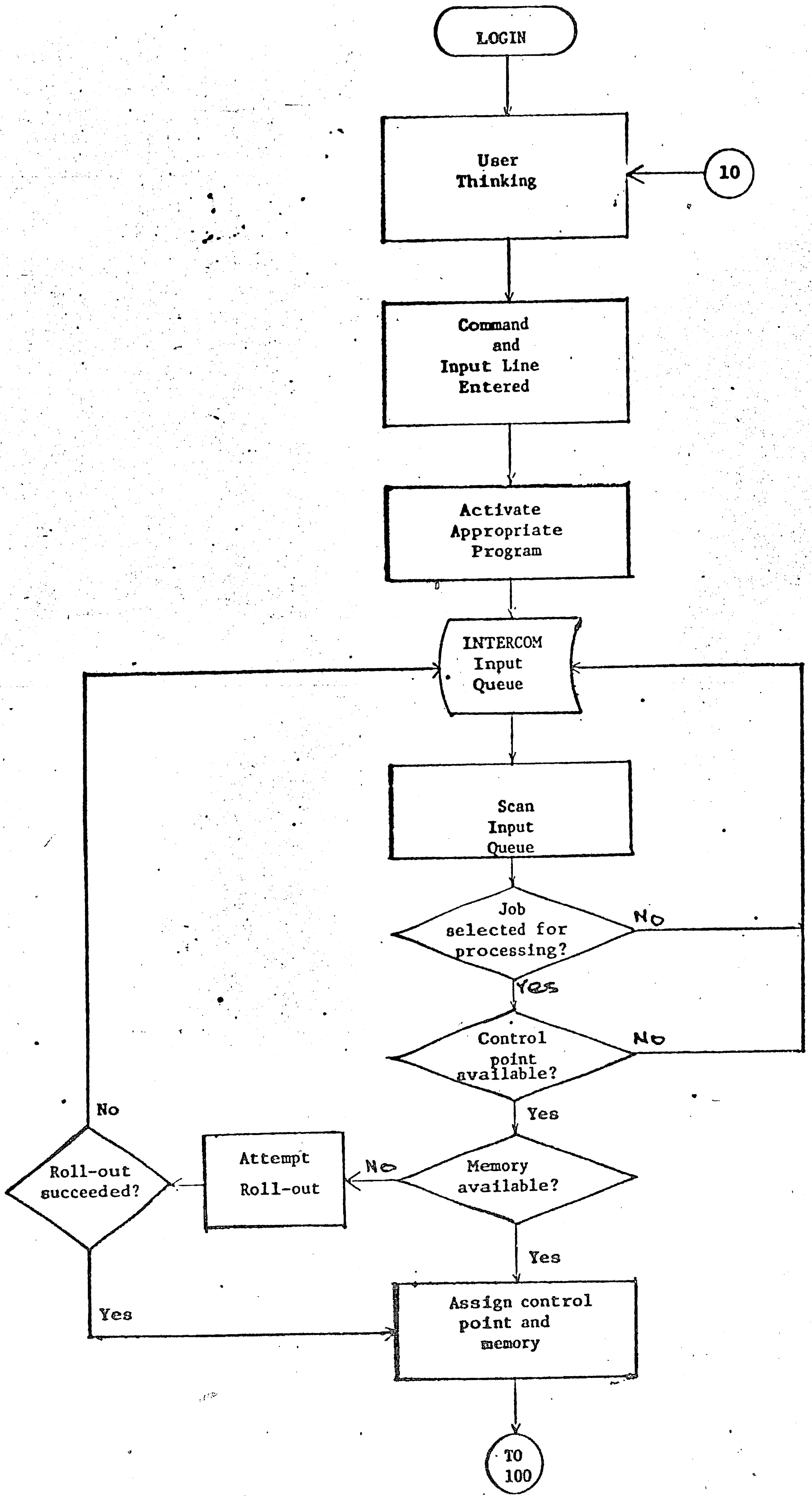


FIGURE 3.5 INTERACTIVE JOB FLOW IN MODEL

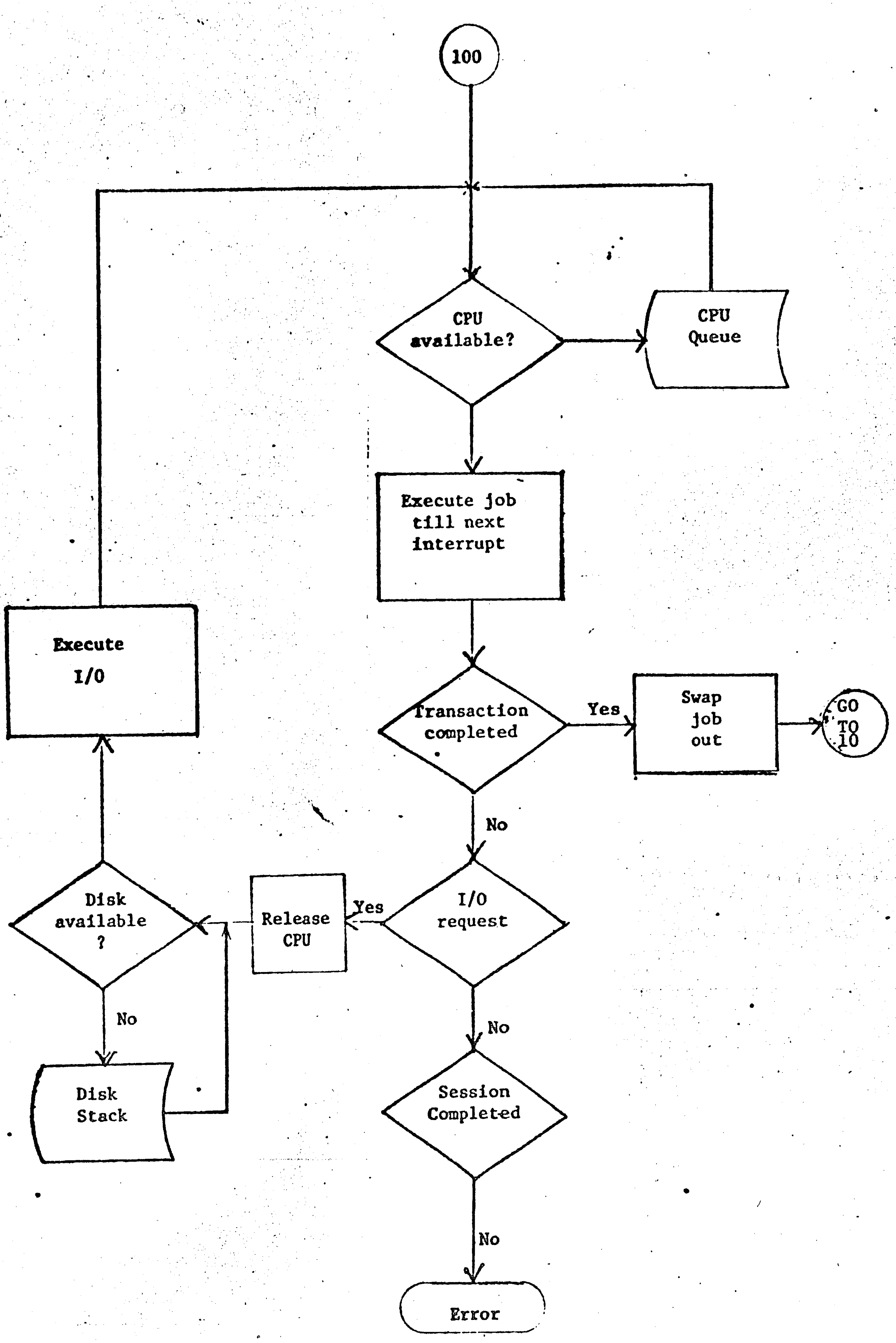


FIGURE 3.5 INTERACTIVE JOB FLOW IN MODEL

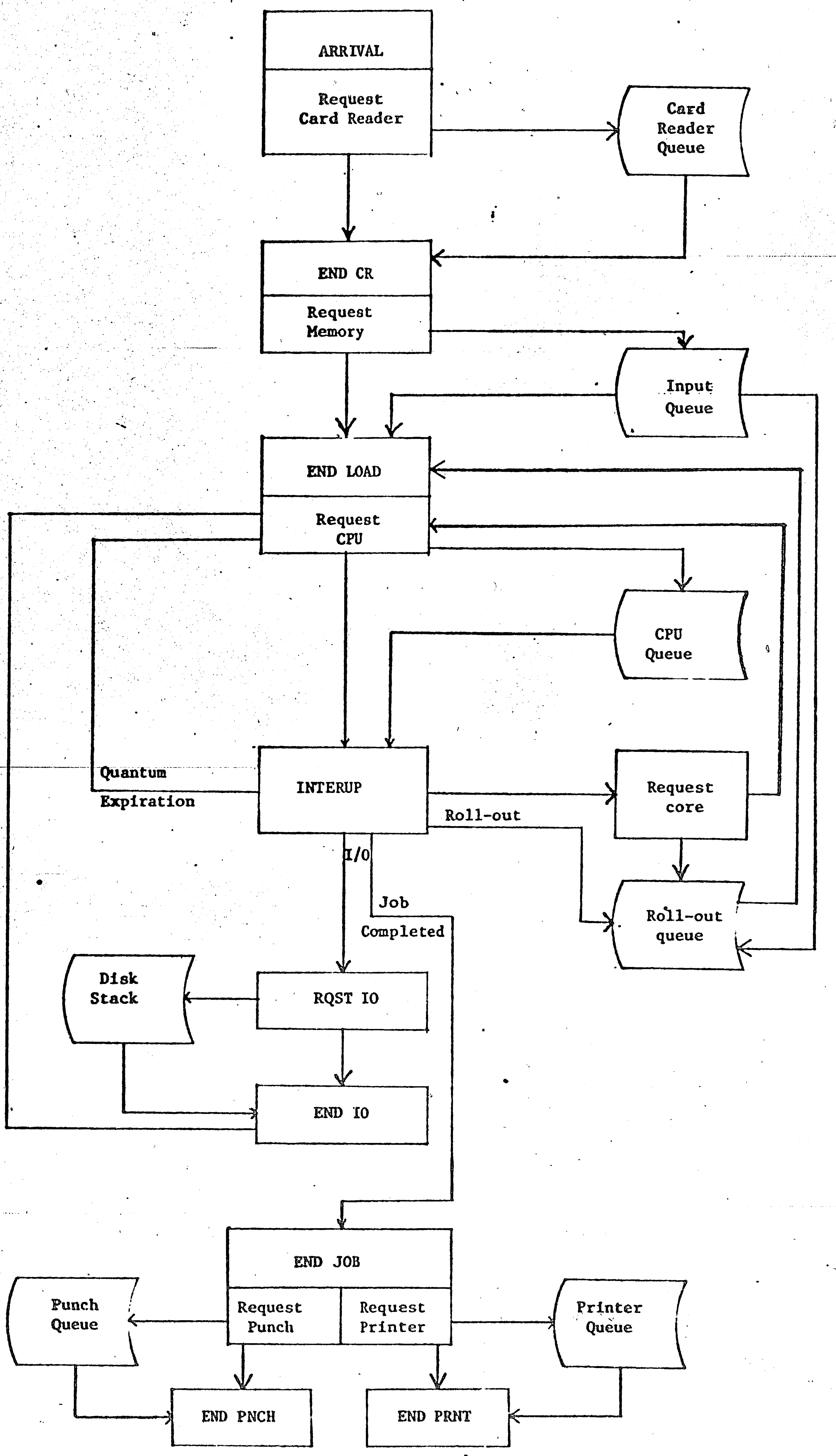


FIGURE 3.6 EVENT ROUTINES ASSOCIATED WITH BATCH JOB PROCESSING

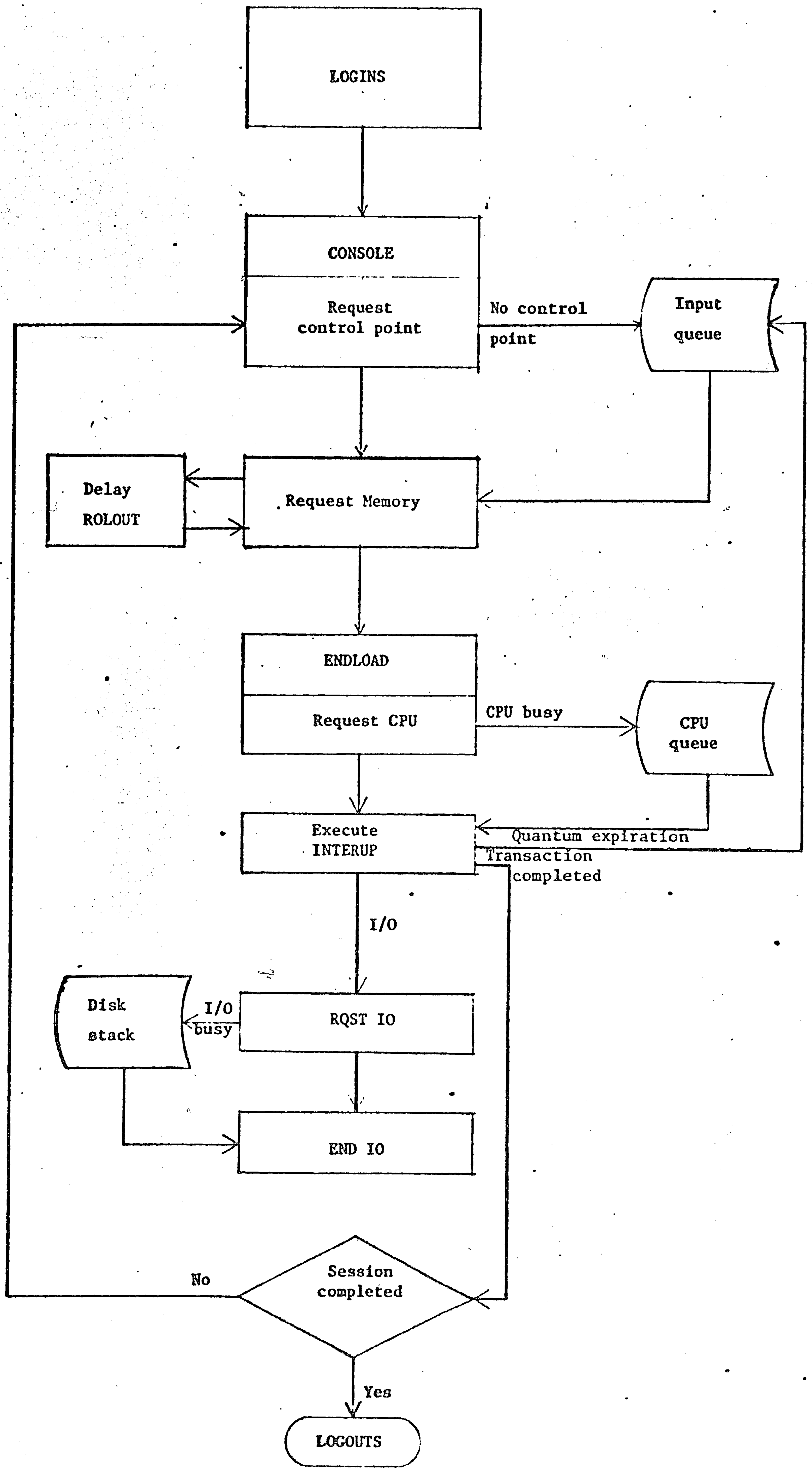


FIGURE 3.7 EVENT ROUTINES ASSOCIATED WITH INTERACTIVE JOB PROCESSING

description of the flow of interactive jobs, any routines that may have been described in the batch job flow will not be described.

EVENT ROUTINE

ARRIVAL DESCRIPTION

The arrival event marks the entry of a new job. The ARRIVAL routine calls the job generator which generates characteristics for the new job. The card reader is then checked to see if it is available. If the card reader is available, the ARRIVAL routine schedules an end-of-card-reading event (ENDCR) for the job. If the card reader is busy, the ARRIVAL routine puts the new job in the card reader queue. The next arrival is then generated.

ENDCR

The occurrence of event ENDCR indicates the completion of the card reading activity. The first job-step or task is ready to begin. The CM core requirements and the CPU time for the first task are generated. The CPU time for the task is computed by dividing the total CPU time for the job by the number of tasks. The time of occurrence of the next I/O is then generated, and the job is put in the input queue. A routine ASSIGNB is then called to attempt to assign control points to jobs in the input queue. When ASSIGNB finds an available control point, it calls the scheduler which attempts to allocate core to the highest priority job in the input queue for which sufficient CM space is available. The scheduler SCHEDLB schedules an end of loading event (ENDLOAD) for every job that it allocates core to. The ENDCR event now checks the card reader queue to see if any job is waiting to be read. If a job is waiting, the routine schedules an ENDCR event for the oldest job in the queue. If the card reader queue is empty, the card reader remains idle.

ENDLOAD

At the occurrence of an end of loading event (ENDLOAD), the job is put in the CPU queue, the active job mix. If the CPU is idle, the dispatching routine DSPATCH is called to dispatch the highest priority job in the CPU queue.

INTERUP

The INTERUP event marks the end of a CPU execution interval or CPU burst. The end of a CPU execution interval is referred to in this thesis as an interruption. Five types of interruptions are identified in the model: I/O request, time-sharing quantum expiration, interactive transaction or batch task completion, interactive session completion and batch job completion. The type of interruption is specified when the DSPATCH routine schedules the INTERUP event. If the interruption is an I/O request, the request-input/output event (RQSTIO) is scheduled. If the interruption is a quantum expiration, the interactive job is swapped out and the oldest job in the INTERCOM swap area is swapped in. The scheduling of intercom jobs is discussed under CONSOLE event. If the interruption is the completion of an interactive transaction, the job is swapped out and another interactive job is swapped in. A CONSOLE event for the next transaction is then scheduled for the job which was swapped out. If the interruption is the completion of a batch task, the CPU time and the core requirements of the next task are generated. If the required core space is less than or equal to the core space for the previous task, the job is put in the CPU queue. If the new core requirement is greater than the space for the previous task, then additional core has to be requested. If the requested core space is available, it is assigned to the job. If sufficient space is unavailable, the job is rolled out to make room for another job for which sufficient space is available. Rolled out jobs have higher priority over input queue jobs when memory space becomes available and a job has to be allocated memory.

When a batch job terminates execution, the end of job event (ENDJOB) is scheduled for the job. When an interactive session is completed, a log-out event (LOGOUTS) is scheduled for the session. The CPU is then assigned to the highest priority job in the CPU queue.

RQSTIO

When a job requests an I/O operation, it takes some time for the monitor to recognize the request and assign a peripheral processor to process the request, and finally for the peripheral processor to process the request. The purpose of the request-I/O event is to take care of the time delay. The RQSTIO routine assigns a disk unit to the job by drawing a random number. The busy state of the disk is then checked. If the disk is busy, the request is put in the disk stack. If the disk is idle, the end of I/O event is scheduled for the job.

ENDIO

After the completion of the I/O operation, the job is ready to resume computation. The job is placed in the CPU queue. The busy state of the CPU is checked. If the CPU is busy, the job remains in the CPU queue. If the CPU is idle, the dispatcher is called to dispatch the highest priority job. The disk stack is then checked. If the disk stack is empty or none of the jobs requires the services of the particular disk, the disk is set to idle; otherwise an end-of-I/O event is scheduled.

ENDJOB

When a job terminates, it takes some time for the monitor to assign a peripheral processor to perform the termination procedure, i.e., to release control point, close files, etc. The ENDJOB routine takes care of the time delay. The routine checks to see if there are any cards to be punched. If the card punch is needed by the job, the busy state of the punch is checked. If the punch is busy, the job is put in the punch queue. If the punch is idle, an end of punching event (ENDPNCH) is scheduled for the job. The busy state of the printer is then checked. If the printer is busy, the job is put in the printer queue; otherwise, an end-of-printing event is scheduled.

ENDPRINT

The job leaves the system by the printer. Some statistics are gathered here. The printer queue is checked. If the queue is empty, the printer is set to idle. If jobs are waiting to be processed, the highest priority job is assigned to the printer. An end-of-printing event is scheduled.

ENDPNCH

Some jobs leave by the printer as well as by the punch. The time a job leaves the system is determined by whichever occurs last, the end of punching or the end of printing. Some statistics are collected here. The punch queue is examined. If the queue is empty, the punch is set to idle. If jobs are waiting to be processed, the highest priority job is selected and assigned to the punch. An end of punching event is scheduled.

LOGINS

The arrival of an INTERCOM user at a terminal is indicated by the log-in event (LOGINS). Routine LOGINS draws a random number to determine whether the job is remote batch or interactive. If the job is interactive, session characteristics such as the total CPU time for the session and the total PP time are generated. The first console session is then scheduled. If the job is remote batch, the LOGINS routine calls the job generator which generates characteristics for the job. An end of card reading event is scheduled for the remote batch job. The next log-in event is then scheduled.

CONSOLE

The CONSOLE event marks the end of the console portion of a transaction. The job characteristics necessary for the execution of the transaction are generated. The occurrence of the next I/O operation is generated and stored in one of the attributes. The transaction is put in the INTERCOM input queue, and a control point requested. If a control point is not available, the transaction remains in the input queue. If a control point is available, it is assigned to the oldest job in the input queue. Memory is then requested. If memory is available, an end of loading event (ENDLOAD) is scheduled. If memory is not available, a roll-out is attempted. If no job can be rolled out, another roll-out attempt (ROLOUT) is scheduled. A rolled-out batch job goes into the roll-out queue.

ROLOUT

The ROLOUT event is scheduled if roll-out attempts are unsuccessful. This routine merely calls the scheduler to schedule an INTERCOM job. The scheduler does not have to roll-out a job if memory has become available since the ROLOUT event was scheduled.

LOGOUTS

The log-out event (LOGOUTS) marks the end of a user session. Appropriate statistics are collected at this point.

UPDATES

The UPDATES routine updates job priorities in the input queue, output queue and the CPU queue. UPDATES reschedules itself. The time intervals between successive priority updates of the input queue, the output queue and the CPU queue are input parameters.

ENDSIM

The ENDSIM routine terminates the simulation. Final statistics are collected on busy states of all the facilities before the simulation is terminated.

OTHER ROUTINES

SCHEDLT

The SCHEDLT routine schedules time-sharing jobs. The routine is called only when there is a control point available. It selects the oldest job in the INTERCOM input queue (or swap area) and attempts to assign memory to it. If sufficient memory is not available, it attempts a roll-out; and if no job can be rolled out, it schedules the ROLOUT event which attempts to schedule the job at a later time. The SCHEDLT routine schedules ENDLOAD event when a job is assigned memory.

SCHEDLB

The SCHEDLB routine schedules batch jobs. This routine is called only if a control point is available. SCHEDLB scans the input queue from the highest priority job down. The highest priority job not exceeding available memory is assigned memory space. The SCHEDLT routine schedules the end of loading event (ENDLOAD) if the job is assigned memory.

DSPATCH

The DSPATCH routine is the dispatcher. It is called only when the CPU is idle. It removes the highest priority job in the CPU queue and schedules the closest interruption.

The general structure of the simulation model and the flow of batch and interactive jobs through the model have been described in this section. The functions of the event routines and a number of specialized routines have also been presented. The development of the model at the task and transaction levels imposes a number of requirements on the data necessary to run the model. The input data required to drive the model is described in the next section.

3.3.5 Input Data Requirements

The two types of data distinguished in the model are GASP and non-GASP data. GASP data are those which are read by the GASP routines, and non-GASP data are all other data supplied to the model. The non-GASP data cards are referred to as model confiuration cards; they include hardware configuration and rate cards, software parameter cards, and control point assignment cards. The model configuration cards are described in Appendix B.

There are eight types of GASP data cards. The GASP data cards are described in detail by Pritsker and Kiviat (41), and will not be described in this thesis. The distribution parameter cards are described in Appendix C.

The simulation of batch and interactive jobs at the task and command or transaction levels respectively requires that data on tasks and transactions be supplied to the model. Data supplied to the model was obtained from the Day File analysis report, "LUCC Daily Operating Statistics" (9). The data on batch jobs obtained from the Day File analysis report, even though at the job level rather than at the task level, could be used without any modification. Tasks are identified in the model because they place different demands on the system's resources. The element which determines task requirements in the model is central memory space. If the CM core requirement of a job does not change during execution, the job can be considered as consisting of only one task, even though it may consist of compilation, execution and file manipulation processes. A task can now be redefined as the computational work between successive memory requests. This redefinition of a task allows the Day

File data to be used to run the model. The average number of tasks or memory requests per job has to be supplied for the two classes of batch jobs, instructional and non-instructional jobs. Remote batch jobs are considered to be non-instructional.

The data on INTERCOM jobs supplied by the Day File report was at the session level, rather than at the command or transaction level. Detailed statistics on interactive transactions could not be obtained. This was a significant handicap in the project. The regular Day Files could not provide data at the command level; software and hardware monitors would be needed to generate data at that level of detail. Sherr (44) has performed extensive statistical analysis of interactive jobs at the transaction level on the CTSS [Compatible Time-Sharing System] at MIT. The characteristics of transactions on the INTERCOM system were believed to be fairly similar to those on the CTSS. The time required for a system to respond to a user command is perhaps the most important performance measure for a time-sharing system. The response time cannot be predicted if the CPU execution time of a transaction is not known. Data collected on transactions on the CTSS system was used as input to the model because of the lack of such data on the INTERCOM system. The primary purpose of incorporating the time-sharing part of the system was not to determine the response time in absolute terms, but rather to determine the effects of changes in system configuration and software parameters on the performance of the system. The use of CTSS data, therefore, should not seriously affect the validity of the model. The data taken from Sherr's book (44) is shown in Table 3.4. Statistics obtained from the SCOPE

TABLE 3.4

Data on Transactions

	File Manipu-lation	Program Input & Editing	Program Running & Debugging	Assembly or Compilation	Misc.
Transaction Probability	1.3	6.3	3.0	1.4	3.4
Avg. Duration of Console Portion of Transaction (sec)	52.	33.	38.	25.	29.
Avg. CPU Time per Transaction (sec)	1,1	0.4	1.5	3.4	0.6
Avg. Transactions per Task	2.8	10.7	5.8	1.7	6.3

Day Files at the session level were also used. For example, the total CPU time for a session was used as the cut-off for that session.

This chapter has dealt with the assumptions made in the development of the model, the features of the model, the representation and generation of jobs, the structure of the simulator, and the input data requirements of the model. The validation of the model and the results of the experiments conducted with the model are discussed in the next chapter.

CHAPTER 4

4.1 VALIDATION

The validity of a simulation is a measure of the extent to which the model represents the real system. The validity of a model depends not on its ability to precisely duplicate the real system, but on the extent to which it satisfies its design objectives. The primary design objective of this simulation model was that it should be able to predict with a 90 percent level of confidence the performance of the system in terms of turnaround time, response time, control point dwell, and CPU utilization.

A computer simulation program may fail to satisfy its design objectives either because of coding errors or logical errors. Coding errors are easier to detect because they usually prevent the program from running to completion. Logical errors are very difficult to detect. Many error checking statements were built into the program to check for the reasonableness of the values of key variables. Special error processing routines were called to indicate the nature of any erroneous conditions that may have occurred. "WRITE" statements were used to monitor the operation of the model.

The validity of a simulation model is difficult to ascertain. Three approaches may be used to assure the validity of a simulation model. Validity may be built into a model by the use of parameters, it may be assured by comparing run results with those obtained from observation of the real system, or it may be checked by soliciting expert opinion of simulation run results (34, page 33).

Validity can be improved by making the model parametric. The parameters are variables that denote the state of the environment and the underlying characteristics of the system. The use of parameters permits the modification of the model characteristics to match the characteristics of the real system and the relationship of the system with its environment. The parametric approach was used in the development of the model. Three parameters were introduced to represent the state of the system in terms of the availability of peripheral processors. The three parameters used in the model were a loading delay, a delay in the recognition of I/O requests, and a delay in the recognition of the termination of a job. These three delay parameters were described in more detail in Section 3.3.4. These parameters can be adjusted to reflect the availability of peripheral processors.

The second validation approach involves the comparison of simulated results with actual measurements. This approach, which can be used only when actual measurements are available, was used to validate the model. The actual workloads for four different days were used to drive the model. Two of the days selected occurred during a semester break; the other two were during the regular semester. Thus, the model was tested with a wide range of workloads.

The validation was done with an awareness of the assumptions made during the development of the model. The two most important assumptions were the absence of system malfunction and the fully automatic operation of the system. The data used to validate the model was collected under the conditions of constant operator surveillance and intervention.

Whether or not there were any system problems on the four selected days could not be ascertained. The effect of hardware and software problems, if any, could not be determined. Therefore, in validating the model, two key variables which were unlikely to be affected greatly by system malfunction or by a moderate amount of operator intervention were analysed. T-tests were conducted on the two performance variables, control point dwell and CPU utilization. Control point dwell is defined as the total time a job spends at a control point. CPU utilization is the fraction of time when the CPU is busy. The actual performance data obtained from the Day File analysis report was compared with the results of simulation runs. The T-test was used to test the hypothesis that the means of control point dwell and CPU utilization obtained from the simulator were the same as the actual means.

Actual and simulated values of the performance indicators are presented in Table 4.1. The model was run for 5400 seconds of simulation time. The t-statistics for the average control point dwell and CPU utilization are also shown in Table 4.1. For a large degree of freedom and a confidence interval of 90 percent, the critical value of t for the T-test is 1.645. This means that, if the hypothesis that the two means are the same is true, there is a probability of 0.90 that the observed value of t will fall in the range -1.645 to +1.645. If the observed value of t for this test is less than 1.645, the true means are equal. Observation of Table 5 reveals that the t statistics for all the four runs are less than 1.645. The hypothesis that the true means are equal therefore cannot be rejected.

TABLE 4.1

ACTUAL VS. SIMULATED VALUES OF PERFORMANCE INDICATORS

	Run #1			Run #2		
	Simulation	Day File	T-test t	Simulation	Day File	T-test t
Average Control Point Dwell (sec)	149.7	154.3	0.1011	91.2	94.4	.123
Control Point Dwell, instructional (sec)	62.9	92.0		32.3	48	
Control Point Dwell, non-instructional (sec)	175.0	180.5		122.2	120.5	
CPU Utilization (percent)	42.8%	43.0	0.2851	35%	34%	1.48
Utilization of Printer 1 (%)	86%	81%		60%	50%	
Utilization of Printer 2 (%)	83%	81%		42%	50%	
Utilization of Punch (%)	9%	10.5%		7%	3.8%	
Intercom Response Time (sec)	4.2	-		4.3	-	
Average CPU Time per Intercom Session (sec)	3.9	3.78		5.29	4.96	
Average CPU Time per Batch Job (sec)	21.6	19.5		12.4	13.2	
Average CPU Time, instructional jobs (sec)	9.7	10.5		4.33	4.69	
Average CPU Time, non-instructional jobs (sec)	25.0	23.3		16.74	18.05	
Simulation Run Time (sec)	5400			5400		

Critical value of t = 1.645

Run #1: Data collected on 3/20/73

Run #2: Data collected on 3/23/73

TABLE 4.1

ACTUAL VS. SIMULATED VALUES OF PERFORMANCE INDICATORS

	Run #3			Run #4		
	Simulation	Day File	T-test t	Simulation	Day File	T-test t
Average Control Point Dwell (sec)	61.0	66.0	0.291	88.3	98.8	0.2005
Control Point Dwell, instructional (sec)	29.5	29.0		30.1	40.2	
Control Point Dwell, non-instructional (sec)	120.1	126.0		178.8	183.0	
CPU Utilization (percent)	28.2	28.6	0.594	23.6	24.8	1.484
Utilization of Printer 1 (%)	57.2	50.0		47.0	45.0	
Utilization of Printer 2 (%)	36.5	50.0		34.0	45.0	
Utilization of Punch (%)	15.0	11.0		9.4	9.2	
Intercom Response Time (sec)	4.22	-		4.85	-	
Average CPU Time per Intercom Session (sec)	3.39	3.99		3.55	3.15	
Average CPU Time per Batch Job (sec)	8.81	10.37		8.36	9.08	
Average CPU Time, instructional jobs (sec)	3.68	3.5		3.06	3.07	
Average CPU Time, non-instructional jobs (sec)	19.0	20.0		17.26	17.8	
Simulation Run Time (sec)	5400			5400		

Critical value of $t = 1.645$

Run #3: Data collected on 3/26/73

Run #4: Data collected on 4/11/73

The F-test was not used in the validation of the model because the empirical distributions for most of the job characteristics were not easily estimated from the Day File analysis report; only the means and the standard deviations were obtained.

4.2 RESULTS OF EXPERIMENTATION

An objective of this project was to study the effect of software parameter changes on the performance of the system. Two of the most important software parameters in the SCOPE 3.3 system are quantum and minimum control point dwell for roll-out. In order to determine the effect of these two software parameters on the performance of the system, a factor analysis experiment was conducted using the analysis of variance technique. The results of the experiment are presented below in Section 4.2.1.

A question of great interest is how the number of control points available to interactive, express and normal batch jobs affects the performance of the system. A separate experiment was conducted to determine the effect of control point availability on key performance measures. The results of this experiment are presented in Section 4.2.2.

4.2.1 Effect of Quantum and Minimum Dwell on System Performance

The analysis of variance can be used to determine whether or not certain factors or their interactions affect the performance of a system. In this experiment, the effects of two factors, quantum and minimum control point dwell for a roll-out, on the performance of the system were analysed. Two levels were selected for quantum, 0.5 and 2.0 seconds.

Three levels were selected for minimum dwell, 0.0, 8.0 and 30.0 seconds. The effects of these two factors and their interation on turnaround time, response time, and control point dwell were analysed using the analysis of variance technique.

Table 4.2 shows the mean values of turnaround time, response time, control point dwell, CPU utilization, and throughput for six combinations of quantum and minimum dwell values. Table 4.2 shows that there is a high degree of interaction among the two factors. When quantum is two, the minimum control point dwell has a linear effect on all the performance variables. Turnaround time and control point dwell decrease with increased minimum dwell; response time increases with increased minimum dwell. When quantum is 0.5 seconds, however, the effect of minimum control point dwell on the performance variables is highly non-linear (see Table 4.2). This observed non-linearity is not unreasonable. In a recent article on the state of art in operating system performance, Lynch (31) stated that the response of an operating system to various workloads and/or software parameters tends to be highly non-linear because of the high degree of interaction among the modules of the operating system.

The LEAPS (Lehigh amalgamated package for statistics) program was used to perform the analysis of variance. The F-ratios and the means obtained in the analysis of variance are presented in Table 4.3. The F-ratio indicates the degree to which a factor affects the performance of the system. The analysis of variance was performed for turnaround time, response time, and control point dwell. For turnaround time, the F-ratios for quantum, minimum dwell, and the interaction of the two factors

TABLE 4.2

SUMMARY OF FACTOR ANALYSIS

Run #	Quantum	Minimum Control Point Dwell (for Roll-out)	Turnaround Time	Response Time	Control Point Dwell	CPU Utilization	Batch Throughput
1	0.5 sec	0.0	381.1	6.37	70.87	29.09	100.05
2	0.5 sec	8.0	338.0	7.54	68.49	29.13	97.4
3	0.5 sec	30.0	506.3	6.86	85.39	32.69	98.0
4	2.0 sec	0.0	412.0	5.31	87.0	27.65	100.05
5	2.0 sec	8.0	361.0	6.19	75.79	28.41	99.4
6	2.0 sec	30.0	256.0	6.3	62.4	28.51	100.05

TABLE 4.3

RESULTS OF THE ANALYSIS OF VARIANCE

FACTOR A = Quantum

FACTOR B = Minimum Control Point Dwell

F - RATIO

	Factor A	Factor B	Interaction of A&B
Turnaround Time	4.6	0.75	9.74
Response Time	5.0	2.08	0.16
Control Point Dwell	0.0	0.71	7.63

MEANS FOR FACTOR A

	Factor A = 2.0	Factor B = 0.5
Turnaround Time (sec)	344.3	409.6
Response Time (sec)	5.78	6.85
Control Point Dwell (sec)	75.0	75.2

TABLE 4.3 (cont'd)

MEANS FOR FACTOR B

	Factor B = 0.0	Factor B = 8.0	Factor B = 30.0
Turnaround Time (sec)	397.7	352.6	380.5
Response Time (sec)	5.75	6.94	6.25
Control Point Dwell (sec)	78.7	72.6	73.9

MEANS FOR INTERACTION (AB)

	Factor A	Factor B 0.0	B 8.0	B 30.0
Turnaround Time (sec)	2.0	413.6	366.4	257.8
	0.5	331.9	338.8	508.2
Response Time (sec)	2.0	5.31	6.21	5.81
	0.5	6.19	7.67	6.69
Control Point Dwell (sec)	2.0	87.0	76.1	61.9
	0.5	70.5	69.2	85.9

were 4.6, 0.75 and 9.70 respectively. For a significance level alpha (α) of 0.10 and 145 degrees of freedom (numerator and denominator) the critical value of F is 1.20. This means that any factor whose F-ratio is greater than 1.2 significantly affects the performance measure. For turnaround time, the F-ratios for quantum and the interaction between quantum and minimum dwell were greater than 1.2. It can be concluded that quantum and the interaction between quantum and minimum dwell significantly affect the turnaround time.

A similar analysis shows that both the quantum and the minimum dwell significantly affect the response time of interactive jobs. The analysis also shows that the interaction between the two factors does not affect the response time.

Analysis also shows that neither quantum nor minimum dwell affects the batch control point dwell significantly; the interaction between the two factors, however, significantly affects control point dwell.

4.2.2 Control Point Availability Experiments

In order to determine the effect on system performance of varying the number of control points available to batch and interactive jobs, four experiments were conducted. In the first experiment, the number of INTERCOM control points was fixed at two, the number of express control points was fixed at one, and the number of control points available to normal batch jobs was varied from one to four. Four runs were made; the results are summarized in Table 4.4.

TABLE 4.4

SYSTEM PERFORMANCE SUMMARY, VARYING BATCH CONTROL POINT AVAILABILITY

	Number of Normal Control Points			
	1	2	3	4
Throughput (per hr)	89.0	94.0	99.5	99.6
Turnaround Time (sec)	677.2	618.5	361.9	315.3
Control Point Dwell (sec)	58.3	76.8	75.79	82.5
Response Time (sec)	5.43	5.73	6.19	6.0
CM Occupancy (%)	41.6	50.5	49.1	48.4
CPU Utilization (%)	27.65	30.3	28.41	31.4
Utilization of Printer 1 (%)	63.8	50.5	60.28	58.6
Utilization of Printer 2 (%)	41.3	41.3	40.26	44.9
Utilization of Control Point 1 (%)	12.1	11.8	7.4	8.25
Utilization of Control Point 2 (%)	70.6	65.4	59.2	59.21
Utilization of Control Point 3 (%)	1.3	1.9	0.3	0.25
Utilization of Control Point 4 (%)	78.1	75.3	65.35	61.7
Utilization of Control Point 5 (%)	-	69.8	51.48	46.3
Utilization of Control Point 6 (%)	-	-	37.00	34.11
Utilization of Control Point 7 (%)	-	-	-	20.3
Waiting Time in Input Queue (sec)	497.5	451.9	153.7	104.0
Waiting Time in CPU Queue (sec)	0.07	0.90	.12	0.13
Waiting Time in Printer Queue (sec)	10.5	5.8	8.8	8.8

INTERCOM Control Points are 1 and 3

Express Control Point is 2

Normal Batch Control Points are any of 4, 5, 6, 7

Simulation Run Time = 5400 seconds

Reducing the number of control points available to batch jobs greatly altered the performance of the system. One has to be very much aware of the objective of the system before reducing the number of control points. The results show that a reduction in the number of control points from four to one decreased and therefore optimized the control point dwell. However, the reduction also doubled the turnaround time and decreased the throughput. Examination of Table 4.4 also shows that a reduction in the number of control points available to batch jobs improved INTERCOM response time. The advisability of reducing the number of batch control points therefore depends on the objective one has in mind. If one is trying to optimize throughput and turnaround time, then a reduction is not advisable. If, however, one is interested in improving the INTERCOM response time, then a reduction in the number of batch control points will give the desired effect. One has to bear in mind that the improvement in response time (10%) was far outweighed by the deterioration in turnaround time (115%).

The first and second experiments were conducted with INTERCOM usage at a moderate level, about 20 sessions per hour. In the second experiment, the number of express and normal batch control points was kept fixed at one and two respectively. Three simulation runs were made with the number of INTERCOM control points varying from one to three. The results of the three runs are summarized in Table 4.5. At low levels of INTERCOM usage, increasing the number of INTERCOM control points from two to three had little effect on the total performance of the system. Decreasing the number of INTERCOM control points from two to one, however, drastically

TABLE 4.5

SYSTEM PERFORMANCE SUMMARY, VARYING INTERCOM CONTROL POINT AVAILABILITY

	No. of INTERCOM Control Points		
	1	2	3
INTERCOM Throughput (per hr)	16	20	22
Batch Throughput (per hr)	99.0	99.0	99.0
Turnaround Time (sec)	372.0	618.5	636.9
Control Point Dwell (sec)	76.46	76.86	78.5
Response Time (sec)	6.73	5.73	5.73
CM Occupancy (%)	51.38	51.5	51.6
CPU Utilization (%)	30.08	30.32	30.32
Utilization of Printer 1 (%)	57.95	50.5	50.5
Utilization of Printer 2 (%)	41.52	41.4	41.4
Utilization of Control Point 1 (%)	11.59	11.85	11.85
Utilization of Control Point 2 (%)	58.73	65.45	65.45
Utilization of Control Point 3 (%)	-	1.92	1.92
Utilization of Control Point 4 (%)	71.9	75.32	0.0
Utilization of Control Point 5 (%)	63.1	69.8	75.32
Utilization of Control Point 6 (%)	-	-	69.8
Utilization of Control Point 7 (%)	-	-	-
Waiting Time in Input Queue (sec)	167.74	451.9	451.9
Waiting Time in CPU Queue (sec)	0.11	0.09	0.09
Waiting Time in Printer Queue (sec)	10.95	5.86	5.86

INTERCOM Control Points are any of 1, 3, 4

Express Control Point is 2

Normal Batch Control Points are 5, 6

Simulation Run Time = 5400 seconds

affected the performance of the system; the response time for time-sharing jobs deteriorated; average turnaround time for batch jobs improved significantly.

This experiment was conducted with INTERCOM usage at an "average" level of about 20 sessions per hour. The average was obtained by dividing the total number of user sessions by the number of hours of system operation. In reality, there is no average INTERCOM usage; INTERCOM usage is either high or low. In order to determine, more realistically, the effect of INTERCOM control point availability on the performance of the system, another experiment was conducted with INTERCOM usage at a level of about 40 sessions per hour.

The results of this experiment are presented in Table 4.6. The results show that interactive response time improved with an increase in the number of control points available to INTERCOM jobs. Increasing the number of INTERCOM control points, from one to two and from two to three, had a non-linear effect on batch job turnaround time and control point dwell. Both turnaround time and control point dwell were worst when the number of INTERCOM control points was two. The results of this experiment indicate that the optimal strategy during periods of high INTERCOM usage would be to assign three control points to INTERCOM, one to express batch jobs, and two to normal batch jobs.

The above strategy makes use of all the seven control points (one for system routines such as JANUS). Another strategy that makes use of all the available control points is to assign two control points to INTERCOM jobs, one to express jobs and three to normal batch jobs. This strategy and the one indicated in this experiment to be optimal were compared in another experiment. The performance of the system for the two control point assignment strategies is shown in Table 4.7.

The results of Table 4.7 demonstrate that there is a trade-off between interactive response time and batch turnaround time. The choice of a control point assignment strategy depends on the performance measures which are being optimized. If one is attempting to improve interactive response time, then the first strategy, two normal batch and three INTERCOM control points, is the preferred approach. If batch turnaround time is the performance measure being optimized, then the second strategy, three batch and two INTERCOM control points, is desirable.

The control point availability experiments lead to a number of conclusions. The first is that, generally, the greater the number of control points available to a class of jobs, the better the performance of the system for that class of jobs. This means that all the seven control points available to jobs should be used at all times. The second conclusion is that the proper definition of system objective is very important. The choice of the wrong performance measure can result in suboptimization and a degradation of the total performance of the system.

TABLE 4.6

SYSTEM PERFORMANCE SUMMARY, VARYING INTERCOM CONTROL POINT ASSIGNMENTS

	No. of INTERCOM Control Points (N)		
	N=1	N=2	N=3
INTERCOM Throughput (per hr)	37	36	38
Batch Throughput (per hr)	89	88	90.5
Turnaround Time (sec)	567.8	1039.9	469.3
Control Point Dwell (sec)	77.9	94.1	78.4
Response Time (sec)	6.6	6.27	5.71
CM Occupancy (%)	57.12	57.92	50.7
CPU Utilization (%)	31.79	34.9	29.29
Utilization of Printer 1 (%)	59.9	64.1	57.1
Utilization of Printer 2 (%)	38.4	41.8	39.0
Utilization of Control Point 1 (%)	23.4	20.78	21.33
Utilization of Control Point 2 (%)	65.9	80.9	65.52
Utilization of Control Point 3 (%)	-	6.1	5.28
Utilization of Control Point 4 (%)	73.5	84.0	74.96
Utilization of Control Point 5 (%)	69.6	79.2	63.04
Utilization of Control Point 6 (%)	-	-	.05
Utilization of Control Point 7 (%)	-	-	-
Waiting Time in Input Queue (sec)	360.8	815.2	267.4
Roll-outs per Hour	57	67	66
Average Time in Roll-out Queue (sec)	34.08	39.36	33.5
Waiting Time in Printer Queue (sec)	8.11	8.8	7.15

INTERCOM Control Points are any of 1, 3, 4

Express Control Point is 2

Normal Batch Control Points are 5, 6

Simulation Run Time = 5400 seconds

TABLE 4.7

COMPARISON OF TWO CONTROL POINT ASSIGNMENTS

	No. of Control Points	
	2 Batch 3 INTERCOM	3 Batch 2 INTERCOM
INTERCOM Throughput (per hr)	38	38
Batch Throughput (per hr)	90.5	99.5
Turnaround Time (sec)	469.3	364.7
Control Point Dwell (sec)	78.4	78.0
Response Time (sec)	5.71	6.25
CM Occupancy (%)	50.7	55.7
CPU Utilization (%)	29.29	33.2
Utilization of Printer 1 (%)	57.1	58.2
Utilization of Printer 2 (%)	39.0	43.2
Utilization of Control Point 1 (%)	21.33	16.8
Utilization of Control Point 2 (%)	65.52	57.5
Utilization of Control Point 3 (%)	5.28	3.0
Utilization of Control Point 4 (%)	74.96	68.9
Utilization of Control Point 5 (%)	63.04	54.0
Utilization of Control Point 6 (%)	0.5	40.8
Utilization of Control Point 7 (%)	-	-
Waiting Time in Input Queue (sec)	267.4	160.5
Roll-outs per Hour	66	70
Average Time in Roll-out Queue (sec)	33.5	30.11
Waiting Time in Printer Queue (sec)	7.15	8.7

Simulation Run Time = 5400 seconds

CHAPTER 5

SUMMARY AND CONCLUSIONS

The objectives of the project were to develop a comprehensive and valid simulation model of the CDC 6400 under the SCOPE 3.3/INTERCOM 3.0 operating system, and to use the model to study how the performance of the system is affected by changes in the workload, system configuration, and software parameters.

The first chapter in this thesis outlined the three approaches used in the evaluation of computer systems. The approaches discussed were monitoring, analytical evaluation, and simulation. The current state of the art in the simulation of computer systems was also presented.

Information on the CDC SCOPE operating system is not widely available. The second chapter therefore described both the hardware and the software of the CDC 6400 system in fairly great detail. The remainder of the thesis was devoted to the description of the model which was developed for the CDC system, and the experiments which were conducted with the model.

One main design criteria was that the simulation model had to be able to predict the performance of the system with a 90 percent level of confidence. The key performance measures used in the evaluation study were batch turnaround time, interactive response time, batch control point dwell, and CPU utilization. This choice of performance measures necessitated the modeling of interactive jobs at the command level and batch jobs at the task level.

Most of the features of the computer system were incorporated in the model. Perhaps the most significant aspect of this simulation model was that it incorporated both batch and interactive job processing in the same model. The interactions between the processing of batch and interactive jobs could therefore be studied. Roll-in/roll-out which is a result of the interaction between batch and interactive jobs and also of dynamic memory requests was incorporated into the model. The roll-in/roll-out feature included the use of a variable, minimum control point dwell for a roll-out, to prevent a batch job from being rolled out until it had resided at a control point for a specified amount of time.

The only aspect of the CDC 6400 system that was not modeled directly was the pool of peripheral processors. The effects of the availability of pool PP's were modeled by the use of three delay parameters. These parameters which represented the state of the system in terms of PP availability improved the validity of the model. An iterative procedure was used to obtain a combination of values of the three parameters.

In validating the model, four runs were made with different workloads. The results of the simulation runs were compared, using the T-test, with the actual performance of the system as obtained from the Day File analysis report, LUCC Daily Operating Statistics. All four validation runs passed the T-test.

A number of experiments were conducted with the model. The effect of reducing the number of control points available to batch and interactive jobs was studied. It was found that, in general, the greater the number of control points available to batch jobs, the better the turnaround time. Likewise, the greater the number of control points available to interactive jobs, the better the response time. It is therefore better to keep ~~all~~ the control points active. The results also showed that a proper definition of overall system objective is very important. For example, if one were to choose control point dwell as the performance measure to be optimized, one would use a strategy which would be optimal for control point dwell, but which would drastically degrade turnaround time and not improve response time.

An analysis of variance was conducted to determine the effect of quantum and minimum control point dwell for roll-out on the performance of the system. It was found that, in general, an increase in the minimum control point dwell improved turnaround time and degraded interactive response time. The effect of quantum was non-linear; more studies would be needed to establish an optimal value for this important variable.

SUGGESTIONS FOR FURTHER WORK

An operating system consists of subsystems such as job initiators, schedulers, dispatchers, disk request processors and job terminators. A substantial amount of information has been accumulated on the behavior of subsystems of operating systems. The interactions among the subsystems are, however, very poorly understood. A great deal of work has to be done in the area of operating system subsystem interactions.

Response time is perhaps the most important measure of effectiveness for a time-sharing system. To be able to predict response time, a simulation model has to model time-sharing jobs at the command level. A model at the command level requires input data at the transaction level. Such data is unavailable on the SCOPE system. The development of a system to gather data on INTERCOM usage at the command level will be very useful in any detailed evaluation of the time-sharing part of the system.

GLOSSARY OF TERMS

ACTIVE JOB - a job in central memory which is ready to use the central processor

BUFFER - a temporary storage device used to compensate for a difference in the rate of flow from one device to another

CENTRAL MEMORY (CM) - fast access core memory used to store programs which are in execution

CENTRAL PROCESSING UNIT (CPU) - the unit of a computer system which contains the arithmetic and logical units

CONTROL POINT - an area of central memory partitioned off by SCOPE for a user job

CPU BURST - time interval between two successive input/output requests

DATA CHANNEL - specialized processing unit that transfers data bidirectionally between an I/O device and the memory of a peripheral processor

DAYFILE - the system job accounting file which records control cards as they are processed and system messages for jobs

DISK STACK - a queue of disk I/O requests waiting to be processed by the stack processor

DISK STORAGE - storage device on which information is recorded on the magnetizable surface of a rotating disk

DISPATCHING - the process of selecting which of the jobs in central memory (i.e., in the active job mix) will use the central processing unit next

EXECUTION - a program is said to be in execution at a specified processor such as the CPU if the program is in control of that processor and the processor is executing instructions belonging to the program

EXTENDED CORE STORAGE (ECS) - a core storage device which may be connected directly to the central memory. Data are transferred bidirectionally between ECS and CM at nearly CM speed under the control of the CPU; programs cannot execute in ECS.

INTERACTIVE JOB - a job submitted via a remote console or teletype where there is an alternation of job output and user input. Also referred to as a conversational job.

INTERCOM - a system of programs which allows jobs to be submitted to a CDC 6000 series computer on a time-shared basis with other remote jobs and locally submitted jobs

INTERRUPT - a hardware facility that causes the central processor to suspend execution, save its status, and transfer control to an interrupt handler

JOB - a unit of computational work which is independent of all other jobs concurrently in the system

JOB STEP - a unit of work of a job. A job is made up of one or more job steps such as compilation, loading, execution.

LOG-IN - the initiation of a terminal user session

LOG-OUT - the termination of a terminal user session

LUCC - Lehigh University Computer Center

MULTIPROCESSING - two or more central processing units simultaneously execute parts of the same program or different programs

MULTIPROGRAMMING	the concurrent residence of two or more programs in the central memory
MULTI-USER JOB	a program that can be used concurrently by several user jobs. Also referred to as a reentrant program.
OPERATING SYSTEM	an integrated set of control programs and processing programs designed to maximize the use of a computer system's resources and to reduce the complexity involved in preparing a program for execution on the computer
OVERHEAD	time spent by the system (CPU) in performing system functions such as scheduling which cannot be charged to a specific user
PERIPHERAL PROCESSOR (PP)	a small special-purpose computer that performs I/O and supervisory functions
PP PROGRAM (ROUTINE)	a program written for execution on a peripheral processor
PREEMPTION	the termination of a currently executing low-priority job and the assignment of the CPU to a higher priority job
PRIORITY NUMBER	a number assigned to a job to indicate the relative urgency or importance of the job
QUANTUM	the length of time which a job is allowed to run before the CPU is taken away from it. Also referred to as time slice.
REENTRANT ROUTINE	a routine or program that can be used concurrently by several user jobs
REMOTE BATCH JOB	a job submitted from a remote location via a 200-USER Terminal into the computer for processing; the results are received at the printer and/or punch at that same remote location. Also known as Remote Job Entry.

RESPONSE TIME - the time between the initiation of a request (issue of a command) at a terminal and the beginning of the response from the system

ROLL-IN - the moving of a rolled-out job back into central memory

ROLL-OUT - the moving of a partially completed job from central memory onto disk to make room for a higher priority job

SCHEDULING - the process of selecting the next job to be loaded into main memory and actually preparing the job for inclusion in the active job mix

SESSION - time interval between user log-in and log-out at a remote terminal

SIMULATION - a numerical technique for conducting experiments on a digital computer; this technique involves certain types of mathematical and logical models that describe the behavior of business, economic, social, biological, physical or chemical systems (or some component thereof) over a period of time

STACK PROCESSOR - a group of PP routines which process disk I/O requests

TASK (Batch job processing) - a unit of computation to which system resources are allocated

TASK (Interactive job processing) - a number of transactions which are initiated by commands of the same type

THROUGHPUT - amount of jobs completed in a specified time interval

TIME-SHARING - the simultaneous interaction with a computer system by several users, each of whom is unaware of the presence of the others

TIME SLICE - see quantum

TRANSACTION - the basic cycle of thinking, issuing a command, waiting for a response, and receiving the response

TURNAROUND TIME - the total time a user job spends in the system. The time lag between the submission of a card deck input and the receipt of the output.

BIBLIOGRAPHY

Abbreviations used in these references:

ACM - Association of Computing Machinery

AFIPS - American Federation of Information Processing Societies

FJCC - Fall Joint Computer Conference

SJCC - Spring Joint Computer Conference

References in alphabetical order:

1. Allen, G. A., "Extended Core Storage for the Control Data 64 6600 Systems," AFIPS Conference Proceedings, Vol. 30, (SJCC 1967), pp. 729-734.

2. Barrett, W. A., Report on Extended Core Storage for LUCC. Lehigh University, August 9, 1972.

3. Bonner, A. J., "Using System Monitor Output to Improve Performance," IBM Systems Journal, Vol. 8, No. 4, 1967, pp. 290-298.

4. Browne, J. C., et. al., "The Interaction of Multiprogram Job Scheduling and CPU Scheduling," AFIPS Conference Proceedings, Vol. 41 (FJCC 1972), pp. 13-21.

5. Bryan, G. E., "JOSS: 20,000 hours at a Console," AFIPS Conference Proceedings, Vol. 31, (FJCC 1967), pp. 769-777.

6. Bull, G. M., and Packham, S. F. G., Time-Sharing Systems, McGraw-Hill, London, 1971.

7. Cardenas, A. F., et. al., Computer Science, Wiley-Interscience, New York, 1972.

8. Cheng, P. S., "Trace-driven System Modelling," IBM Systems Journal, Vol. 8, No. 4, 1969, pp. 280-289.

9. Cichelli, Richard, LUCC Daily Operating Statistics, Lehigh University Computer Center.

10. Clayton, B. B., et. al., "An Operating System and Programming Systems for the 6600," AFIPS Conference Proceedings, Vol. 26, part II (FJCC 1964), pp. 41-57.

11. Cohen, L. J., Operating System Analysis and Design, Spartan Books, New York, 1970.

12. Control Data Corp., SCOPE 3.3 System Programmer's Reference Manual, CDC, Publication No. 60307800A, Mineapolis, Minnesota, 1971.

13. Control Data Corp., 6000 Series Computer Systems Reference Manual, CDC, Publication No. 60100000, Minneapolis, Minnesota, 1972.

14. Control Data Corp., SCOPE Reference Manual, 6000 Version 3.3, CDC, Publication No. 60305200B, Minneapolis, Minnesota, 1971.

15. Control Data Corp., Internal Maintenance System, 6000 SCOPE 3.3, CDC, Minneapolis, Minnesota.

16. Control Data Corp., Internal Maintenance System INTERCOM 3.0, CDC, Minneapolis, Minnesota.

17. DeCagama, A., "A Methodology for Computer Model Building," AFIPS Conference Proceedings, Vol. 41, (FJCC 1972), pp. 299-310.

18. Dickinson, R. V., and Orr, W. K., "System Ten - A new approach to multiprogramming," AFIPS Conference Proceedings, Vol. 37, (FJCC 1970), pp. 181-186.

19. Enslow, Philip H., Operating Systems, Monitors, Executives and Supervisors, American University, 4th Revision, 1970.

20. Feinroth, Y., et. al., "Telecommunications using a front-end minicomputer," Communications of ACM, Vol. 16, No. 3, March 1973, pp. 153-160.

21. Fox, D., and Kessler, J. L., "Experiments in software modelling," AFIPS Conference Proceedings, Vol. 31, (FJCC 1967), pp. 429-436.

22. Hansen, P. B., "The nucleus of a multiprogramming system," Communications of ACM, Vol. 13, No. 4, April 1970, pp. 238-250.

23. Harrison, M. C., and Schwartz, J. T., "SHARER, a time sharing system for the CDC 6600," Communications of ACM, Vol. 16, No. 3, March 1963, pp. 153-160.

24. Herman, Donald J., "SCERT, a computer evaluation tool," Datamation, Vol. 13, No. 2, Feb. 1967, pp. 26-28.

25. Hunt, E., et. al., "Who are the users? - An analysis of computer use in a University Computing Center," AFIPS Conference Proceedings, Vol. 38, (SJCC 1971), pp. 231-238.

26. Hutchinson, G. K., "A Computer Center Simulation Project," Communications of ACM, Vol. 8, No. 9, Sept. 1965, pp. 559-563.

27. Katzan, Harry, "Operating System Architecture," AFIPS Conference Proceedings, Vol. 36, (SJCC 1970), pp. 106-117.

28. Katzan, Harry, Advanced Programming, Van Nostrand Reinhold Co., New York, 1970.

29. Kimbleton, Stephen R., "The role of computer system models in performance evaluation," Communications of ACM, Vol. 15, July 1972, pp. 586-590.

30. Lucas, Henry C., "Performance Evaluation and Monitoring," Computer Survey, Vol. 3, No. 3, Sept. 1971, pp. 79-91.

31. Lynch, W. C., "Operating System Performance," Communications of ACM, Vol. 15, July 1972, pp. 579-586.

32. MacDougall, M. H., "Computer System Simulation: an introduction," Computing Surveys, Vol. 2, No. 3, Sept. 1970, pp. 191-205.

33. MacDougall, M.H., "Simulation of an ECS-based Operating System," AFIPS Conference Proceedings, Vol. 30, (SJCC 1967), pp. 735-741.

34. Maisel, H., and Gnugnoli, G., Simulation of Discrete Stochastic Systems, Science Research Associates, Inc., Chicago, 1972.

35. Mallach, E. G., "Simulating the Viatron 2140/2150 on the IBM 360", Simulation, Vol. 16, No. 3, March 1971, pp. 130-136.

36. Nielson, Norman R., "The Simulation of Time-Sharing Systems," Communications of ACM, Vol. 10, July 1967, pp. 397-412.

37. Noe, J. D., and Nutt, G. J., "Validation of a Trace-Driven CDC 6400 Simulation," AFIPS Conference Proceedings, Vol. 40, (SJCC 1972), pp. 749-757.

38. Noetzel, A. S., "The design of a meta-system," AFIPS Conference Proceedings, Vol. 38, (SJCC 1971), pp. 415-423.

39. Norland, K. E., and Bulgren, W. C., "A simulation model of GECOS III," Proceedings of ACM National Conference, 1971, pp. 596-611.

40. Nutt, G. J., SIM 6000 Program Description and Use, University of Washington Computer Center Science Group, Technical Report No. 71-07-05.

41. Pritsker, A. A. B., and Kiviat, P. J., Simulation with GASP II, Prentice Hall, Englewood Cliffs, New Jersey 1969.

42. Schwetman, H. D., A Study of Resource Utilization and Performance Evaluation of Large Scale Computer Systems, Doctoral Dissertation, University of Texas at Austin, 1970.

43. Sherman, S., et. al., "Trace-driven modelling and analysis of CPU scheduling in a multiprogramming system," Communications of ACM, Vol. 15, No. 12, Dec. 1972, pp. 1063-1069.

44. Sherr, A. L., An Analysis of Time-shared Computer Systems, MIT Press, Cambridge, Mass., 1967.

45. Shiveley, M. W., The micro-scheduling of computer jobs, Doctoral Dissertation, Lehigh University, 1971.

46. Smith, J. L., "An analysis of time-sharing computer systems using Markov models," AFIPS Conference Proceedings, Vol. 28,(SJCC 1966), pp. 87-95.

47. Steven, D., "On Overcoming High Priority Paralysis in Multiprogramming Systems: A Case History," Communications of ACM, Vol. 2, No. 8, Aug. 1968, p. 539.

48. Thornton, J. E., "Parallel Operation in the Control Data 6600," AFIPS Conference Proceedings, Vol. 26, (SJCC 1964), pp. 33-39.

49. Watson, R. W., Time-sharing System Design Concepts, McGraw-Hill, New York, 1970.

50. Wirth, N., "Program development stepwise refinement," Communications of ACM, Vol. 14, No. 4, April 1971, pp. 221-226.

APPENDIX A

FILE AND ATTRIBUTE ASSIGNMENTS

1. FILE ASSIGNMENTS IN THE MODEL

FILE	
1	Event file
2	Card reader queue
3.	Batch input queue
4	INTERCOM input queue, swap area
5	CPU queue, active job mix
6	Disk stack
7	Roll-out queue
8	Printer queue
9	Punch queue
10	Temporary scratch file
11	Temporary scratch file
12	Temporary scratch file

2. STRUCTURE OF EVENT FILE (File 1)

ATTRIBUTE	
1	Scheduled time of event
2	Event code

The following event codes are used

CODE	
1	ARRIVAL
2	ENDCR
3	INTERUP
4	ENDIO
5	ENDPRNT
6	ENDPNCH
7	LOGINS
8	CONSOLE
9	LOGOUTS
10	UPDATES
11	ENDLOAD
12	RQSTIO
13	ENDJOB
14	ROLOUT
15	ENDSIM

3. ATTRIBUTE ASSIGNMENT

1	Scheduled Time of Event
2	Event code
3	Time unit entered system

3. ATTRIBUTE ASSIGNMENT (continued)

4	Time unit entered queue
5	Central Memory Space
6	Peripheral processor time
7	Number of cards read
8	Number of cards punched
9	Number of lines printed
10	Number of magnetic tape drives
11	Central processor time
12	Time central processor was assigned
13	CP time currently used
14	CP time before next disk I/O
15	Total CPU time currently used in session
16	Total CPU time for the session
17	Number of transactions in task left
18	Type of task
19	Type of interrupt
20	Job number
21	Job type or class
22	Job priority class
23	Current job priority
24	Control point assigned to job
25	Time control point was assigned
26	Printer number (batch)
27	Disk assigned to job
28	Time INTERCOM transaction was initiated
29	CPU time used since last priority recalculation
30	I/O time currently used

APPENDIX B

MODEL CONFIGURATION CARDS

1. TYPE A CARD

Field	Description	Variable
1	Number of card readers	NCARD
2	Number of line printers	NLINE
3	Number of punches	NPUNCH
4	Number of tape drives	NTAPE
5	Number of control points	NPOINT
6	Number of remote batch terminals	NRBATCH
7	Number of teletypes	NTELE
8	Number of priority classes	NPCL
9	Number of disks	NDISK
10	Number of disk controllers	NDISC
11	Number of plotters	NPLOT
12	Number of paper tape readers	NPAR
13	Number of paper tape punches	NPAP

FORMAT IS: FORMAT (5x, 10I5, 3I5).

2. TYPE B CARD

Field	Description	Variable
1	Maximum amount of CM	CMMAX
2	Minimum amount of CM requested by batch job	CMMINB
3	Minimum amount of CM requested by interactive job	CMMINT
4.	Maximum CM for express job	CMEXP
5	Maximum CP time for express job	CPEXP
6	Time increment for input queue aging	
7	Time increment for output queue aging	
8	% of PP time used for disk I/O	PCENT
9	Threshold for CPU dispatching	THRESH
10	Minimum control point dwell before roll-out	ROLLMIN

FORMAT IS: FORMAT (5x, 4F10.4, 7F5.2)

3. TYPE C CARD

Description	Variable
No. of INTERCOM control points	MINTER
No. of express control points	NEXP
No. of normal control points	NNORM
No. of system control points	NJANUS
INTERCOM control points	TERM(5)
Express control points	EXP(5)
Normal control points	SNORM(5)
System control points	SJANU(5)

Card	Variables	Format
1	NINTER, NEXP, NNORM, NJANUS	(5x,4I5)
2	TERM(J), J=1, NINTER	(5x,5I5)
3	EXP(J), J=1, NEXP	(5x,5I5)
4	SNORM(J), J=1, NNORM	(5x,5I5)
5	SJANU(J), J=1, NJANUS	(5x,5I5)

4. TYPE D CARD

Respective unit record per minute rates for:

a. Each individual card reader, followed by
b. Each individual line printer, followed by
c. Each individual card punch, followed by
d. Each individual remote and reader.

FORMAT IS: FORMAT (8F10.4)

5. TYPE E CARD

Description	Variable
Batch job rate (per hour)	RJOB
INTERCOM job rate (per hour)	RTJOB

FORMAT IS: FORMAT (2F10.4)

APPENDIX C

DISTRIBUTION PARAMETER CARDS

Card No.		Distribution
1	Cards read, class 1	Erlang
2	Cards read, classes 2 and 3	Erlang
3	Lines printed, class 1	Erlang
4	Lines printed, classes 2 and 3	Erlang
5	Memory space for INTERCOM	Erlang
6	No. of transactions in Task 1	Poisson
7	No. of transactions in Task 2	Poisson
8	No. of transactions in Task 3	Poisson
9	No. of transactions in Task 4	Poisson
10	No. of transactions in Task 5	Poisson
11	Console portion duration in Task 1	Erlang
12	Console portion duration in Task 2	Erlang
13	Console portion duration in Task 3	Erlang
14	Console portion duration in Task 4	Erlang
15	Console portion duration in Task 5	Erlang
16	CPU time per transaction, Task 1	Erlang
17	CPU time per transaction, Task 2	Erlang
18	CPU time per transaction, Task 3	Erlang
19	CPU time per transaction, Task 4	Erlang
20	CPU time per transaction, Task 5	Erlang
21	INTERCOM swapping delay	Erlang
22	Roll-in delay	Erlang
23	CM loading delay	Erlang
24	I/O request delay	Erlang
25	Job ending delay	Erlang
26	INTERCOM session CPU time	Erlang
27	INTERCOM session PP time	Erlang
28	Cards punched, class 1	Erlang
29	Cards punched, classes 2 and 3	Erlang
30	Memory space, class 1	Erlang
31	Memory space, classes 2 and 3	Erlang
32	CPU time, class 1	Erlang
33	CPU time, classes 2 and 3	Erlang
34	PP time, class 1	Erlang
35	PP time, classes 2 and 3	Erlang
36	CPU burst	Erlang
37	I/O duration, class 1	Erlang
38	I/O duration, classes 2 and 3	Erlang
39	I/O duration, INTERCOM	Erlang

BIBLIOGRAPHICAL NOTE

Robert Kofi Baafi was born in Akim Oda, Ghana, on June 12, 1946. He attended Adisadel Secondary School in Cape Coast, Ghana. He went to the University of Connecticut in 1967 on a scholarship from the African-American Institute, New York. He received a Bachelor of Science degree in Electrical Engineering in 1971. While at the University of Connecticut, Mr. Baafi worked part-time with the New England Research Application Center, one of a number of information retrieval centers set up by NASA to disseminate scientific and technological information.

Since his graduation from UCONN, the author has pursued the M. Sc. degree in Management Information Systems in the Industrial Engineering Department at Lehigh University. This degree will officially be granted in October 1973.

Mr. Baafi married the former Pamela Dean Eaton of Storrs, Connecticut, in August 1970. He is a member of Tau Beta Pi and Eta Kappa Nu.