A SIMULATION STUDY OF THE TIME-SHARING COMPUTER SYSTEM AT THE NAVAL POSTGRADUATE SCHOOL

by

Ronald Maxwell Goodman

United States
Naval Postgraduate School

THESIS

A SIMULATION STUDY
OF
THE TIME-SHARING COMPUTER SYSTEM
AT
THE NAVAL POSTGRADUATE SCHOOL

by

Ronald Maxwell Goodman

and

Leo Michael Pivonka

June 1969

A Simulation Study of the Time-Sharing Computer System at the Naval Postgraduate School

by

Ronald Maxwell Goodman
Lieutenant, United States Navy
B.A., University of Denver, 1957
M.A., University of Denver, 1959

and

Leo Michael Pivonka
Lieutenant, United States Naval Reserve
B.A., University of Kansas, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

ABSTRACT

A GPSS model of the CP/CMS time-sharing computer system at the Naval Postgraduate School was constructed, and was used in three experiments to investigate the performance of the system under a variety of conditions. In each of the experiments the model generated auto-correlated sequences of observations which were analyzed using techniques adapted from spectral analysis.

An experiment to determine the effect of an increased number of terminals on average response time revealed that the system adequately could support a 25% increase in the number of terminals, and that the number of terminals was limited by the magnetic disk I/O capability. In a second experiment it was found that increasing the number of disks from four to eight would enable the system to support up to 20 terminals. A final experiment involving the examination of a new scheduling algorithm showed no significant changes in average response times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The computer center of the Naval Postgraduate School operates a time-sharing system which currently supports 12 communication terminals. Anticipating a growth in demand for time-sharing services, there was an interest in determining how many additional terminals could be supported without seriously degrading the performance of the system. Further, it is conceivable that the system response may be improved by modifying the important software algorithm used for job scheduling.

A. THE PROBLEM

1. Statement of the Problem

This study was undertaken (1) to estimate the amount of system degradation which would accrue as a result of the additional load; (2) to examine alternatives to the present assignment of system resources; and (3) to test the effects of modification to the main scheduling algorithm.

2. Importance of the Study

The decision to apply an increased load to an established computer system requires some knowledge of what effect that additional load will have on overall system performance. To make such a judgment purely on the basis of speculation or supposition is considered foolhardy at best. Therefore, it is intended that the results of this study will serve as an aid in making more informed decisions regarding proposed modifications to the system.

3. Methods of Analysis

Ideally, an experiment to test the effects of hardware modifications or software changes to a time-sharing computer system would be conducted in a laboratory atmosphere with the actual equipment on hand. Clearly, this would require much in the way of resources not the least of which would be the hardware and/or software used to measure system performance. In most instances, this method of analysis is prohibited by financial considerations alone.

Analytical mathematical models may be considered the ideal analysis tool in some situations. However, the impracticality of this method becomes readily apparent in the event that the system being modeled is as complex as a modern time-sharing computer system. In order to construct an analytical model for which a solution can be obtained it is necessary to make a number of simplifying assumptions. The inaccuracies introduced into the model are difficult to estimate. Thus, while analytical models may serve as useful tools in studying some aspects of computer systems, their usefulness in providing information about a specific complex time-sharing system is questionable and, in cases such as this, computer simulation seems to be the logical approach.

B. DEFINITIONS OF TERMS USED

Definitions of frequently used terms are presented here for the convenience of the reader.

1. Time-sharing system - a remote, multi-access data processing system which allows many users simultaneously to utilize the resources of the system.

2. Core block - a set of 4096 consecutive bytes of main storage, the first byte of which is located at a storage address that is a multiple of 4096.

3. DASD - direct access storage device; for example, 2301 drum or 2311 disk.

4. FIFO - an acronym from First In First Out.

5. Job - this term is used synonymously with "task" throughout this study to denote the processing required as a result of a command from a terminal.

6. Main storage - used synonymously with "core" to refer to storage directly addressable by the central processing unit.

7. Page - 4096 consecutive bytes of program or data which may be located either in main storage or auxiliary storage.

8. Paging - the transfer of pages of information between main storage and on-line auxiliary storage to enable a number of concurrently active programs to share a limited amount of main storage.

9. Reaction time - the time interval between the terminal's typing the first letter of a reply and the user's typing the last letter of a new command. This includes the time required to type the reply and the command as well as any additional time the user spends thinking.

10. Response time - the time interval between the user's typing the last letter of a command and the terminal's typing the first letter of the reply [Ref. 10].

11. Virtual computer - a conceptual computer which is made to appear real by the control program. A distinct virtual computer is associated with each remote terminal.

12. Virtual storage - the main storage of a virtual computer. Virtual storage does not necessarily reside in real main storage.

13. Virtual address - an address which references virtual storage and must be translated into a real storage address before it is used.

14. Dynamic address translation - the process of converting virtual addresses into actual main storage addresses during instruction execution.

C. ORGANIZATION OF REMAINDER OF THE PAPER

Section II presents a detailed description of the computer system under consideration. The model of the system is described in Section III which also contains discussion relating to validation and verification. Experimental and statistical considerations, with emphasis on the method of statistical analysis employed in this study, are presented in Section IV. Section V contains a summary of the results of the actual experiments which were performed on the model. Conclusions appear in

Section VI. Following the appendices, which contain statistical data, sample calculations, and flowcharts of the model, appear sample program output and program listings.

II. THE SYSTEM

The time-sharing system discussed in this paper consists of an IBM System/360 Model 67-2 data processing system and its associated software. The software, known as CP/CMS, was developed at the IBM Cambridge Scientific Center and consists of two independent components: (1) the Control Program (CP-67) which manages the resources of the System/360 so that remote users appear to have a dedicated machine at their disposal; and (2) the Cambridge Monitor System (CMS), a conversational operating system which enables users to interact with their programs from their terminals using relatively simple commands.

A. HARDWARE

The hardware configuration of the Model 67-2 system under consideration is diagrammed in Figure 1. A detailed description of the Model 67 may be found in Ref. 5. The card reader, punch, and printer are not shown or discussed in detail since all unit record I/O is spooled on disks by CP and not queued for physical I/O until requested by the user. The number of requests for physical I/O is very small and therefore their effect on system performance is assumed to be negligible.

1. Processor

The Central Processing Unit, CPU, is an IBM Model 2067-2 which contains facilities for (1) instruction decoding and execution; (2) performing arithmetic and logical operations; and (3) addressing up to eight 2365-12 Processor Storages. In order to function more efficiently in a paged time-sharing environment the 2067-2 is equipped with hardware implemented dynamic address translation. If dynamic address

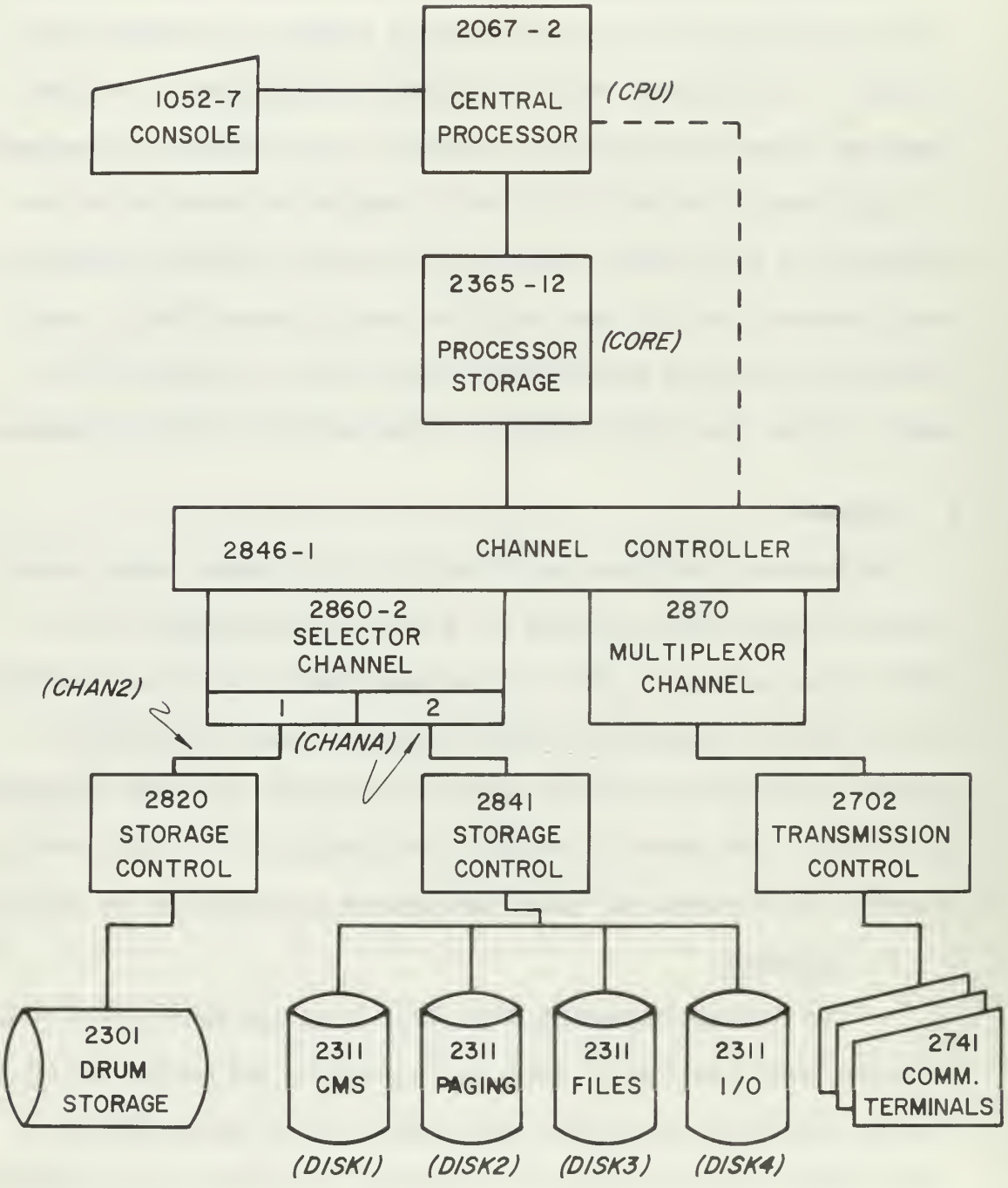


FIGURE 1

CP/CMS SYSTEM CONFIGURATION

translation fails because the referenced virtual page is not core resident a special interrupt, known as a page-translation exception, is recognized.

2. Main Storage

The main storage for the system consists of one IBM 2365 Model 12 Processor Storage Unit which is directly addressable by the CPU and by the 2846 Channel Controller. The 2365 has a basic 750 nanosecond storage cycle and accesses eight bytes (64 bits) in parallel. Each 2365 contains two independently accessible storage arrays of 128K bytes each. Since these two arrays can be accessed simultaneously an average cycle time of 375 nanoseconds is theoretically possible. The 2365 has a total capacity of 256K bytes or 64 pages.

3. Channels

a. Channel Controller

The IBM 2846 Channel Controller provides the communication interface between the CPU and the channels, and also provides the data paths for control information and data transfers between main storage and the channels. It is the IBM 2870 Multiplexor Channel and the two IBM 2860 Selector Channels which actually execute the channel command word packages from main storage and, thus, control all I/O operations for their attached devices. In addition to their control functions the channels also provide the data path between their attached I/O units and the channel controller.

b. Selector Channel 1

Selector Channel 1 is dedicated to the IBM 2301 Drum Storage Unit. The 2301 is composed of 200 addressable tracks on which data is recorded in parallel groups of four bits. Each addressable track has four fixed-position read/write heads so that an entire track of 20,483

bytes can be read in one revolution of the drum. The drum makes a complete revolution each 17.5 milliseconds, giving a data transfer rate of 1,200K bytes per second. The total capacity of the drum is 4.09 million bytes, or approximately 1,000 pages.

c. Selector Channel 2

Linked to this selector channel are four IBM 2311 Disk Storage Drives. Each disk contains 200 cylinders with ten tracks per cylinder. The access mechanism is constructed so that the ten read/write heads can access the ten tracks in any one cylinder without repositioning. Data is recorded serially by bits along a track. Therefore, it requires a full rotation of the disk (25 milliseconds) to read one 3,625 byte track. The total storage capacity of a 2311 disk is 7.25 million bytes, or about 1,770 pages. Timing characteristics, shown in Table I, were obtained from Ref. 6 which contains a more detailed description of both devices.

It should be noted that the timing figures given in Table I do not reflect the time which a read or write request must wait for a device or channel to become free. For instance, a request to read or write a record on one of the 2311 disks must wait for both the channel and the access mechanism, or arm, to become free. The channel then may initiate a seek for that arm. The seek time is the time required for the arm to be positioned at the appropriate cylinder. During this seek time the channel is free to service other requests or transmit data from the other three disks. After the seek is completed, there may be another delay until the channel again is unoccupied. Once the channel is free, there is a rotational delay to position the home address (record zero) of the track under the read/write head and an

TABLE I

AUXILIARY STORAGE TIMING CHARACTERISTICS

UNIT AND FUNCTION	MEAN	SPREAD
2301 Drum		
Rotational delay	8.75 msec.	0 - 17.5 msec.
Transmission time/page	3.0 msec.	-
Total	11.75 msec.	3 - 20.5 msec.
2311 Disk		
Seek time	74.25 msec.	25 - 135 msec.
Rotational delay:		
to home address	12.5 msec.	0 - 25 msec.
to record	12.5 msec	0 - 25 msec.
Transmission time/page	26.0 msec.	-
Total	125.25 msec.	25 - 216 msec.

additional rotational delay to position the desired record under the head. The final delay is the time actually required to read or write the record, known as transmission time. The lengths of the waiting times mentioned are functions of the rate of reference to the disks and have been the subject of both simulation studies [Ref. 10] and queueing analyses [Refs. 1 and 10].

d. Multiplexor Channel

The IBM 2870 Multiplexor Channel supports the low data rate I/O devices of the system. It concurrently can sustain I/O operations on several I/O devices by interleaving bytes of data associated with different devices and routing them to or from the proper locations in main storage. The I/O devices of most interest in this study are the 12 IBM 2741 Communication Terminals which interface with the multiplexor channel through an IBM 2702 Transmission Control Unit. These terminals are IBM Selectric typewriters with added electronic controls for communication with the computer system. In the time-sharing environment the terminals become the operator's consoles for the virtual machines, and from them the users can control the execution of their programs.

4. DASD Allocation

In the present system configuration the drum is used entirely for paging and can accommodate virtual storages for 13 to 15 users depending upon blocking factors and the number of system pages which are made drum resident. Disk number one is the system disk and contains about 500 pages of system programs including compilers, assemblers and macro-libraries. Disk two is reserved for paging of any virtual storages which cannot fit on the drum. Disks three and four are used to store user files, to spool I/O, and to store scratch files.

B. SOFTWARE

The two major components of the CP/CMS time-sharing software system are the Control Program, CP-67, and the monitor system, CMS. CP creates the time-sharing environment enabling many users to simultaneously perform work, while CMS produces the conversational atmosphere which enables users to interact with their programs. Each component is capable of operating independently on an appropriate System/360 configuration. Thus, CP could be used without CMS to construct a non-conversational time-sharing system; and, similarly, CMS could be used without CP to produce a conversational operating system without time-sharing capabilities. When the two are used together, in order to take full advantage of the features of both, CMS operates under the supervision of CP. Since CP is responsible for creating and managing the time-sharing environment its structure and operation are considered in more detail. A complete description of CP-67 and CMS is available in Ref. 9.

CP constructs and maintains virtual computers which are indistinguishable from real computers to both the user and his programs. Whenever a user activates a terminal, CP creates for his personal use a virtual computer system of predefined configuration. Since the systems are virtual their configurations need not correspond to either the real system or each other. In the system studied, the normal virtual configuration included a Model 65 CPU, 256K bytes of main storage, three disk drives, and a console, the user's 2741 terminal. To each user his virtual system appears real and he uses it as if it were real. Actually, the resources of the real computer system must be shared among the virtual computer systems of all concurrently active users. Thus, CP must schedule and allocate real resources to the various virtual computers.

1. Execution Control

One resource which must be shared is real CPU time. The main dispatcher and control routine, DISPATCH, of CP is responsible for allocating the execution time in the CPU among the virtual computers. After the processing of each interrupt, control is passed to DISPATCH which charges time used within the control program to the appropriate user and performs other "housekeeping" functions. DISPATCH then determines if the interrupted user has run to the end of his time slice and, if he has, lowers his priority one level before starting the highest priority runnable user. If the user has not completed his allotted time slice and there are no higher priority runnable users, he then is restarted with the remaining portion of his initial time slice. However, if a higher priority runnable user is found, that user's time slice is computed and he is started.

The time slice assigned each user task is determined, before it is started, on the basis of priority as indicated in Table II.

TABLE II

TIME SLICE AS A FUNCTION OF PRIORITY

PRIORITY	TIME SLICE
10, 9, 8	50 msec.
7, 6, 5	100 msec.
4, 3	150 msec.
2, 1	200 msec.

Users are placed at the highest of the ten priority levels whenever they complete a terminal interaction. When a user is started, his

time slice, or the remaining portion, is loaded in the interval timer, so that a timer interrupt signals the completion of a time slice. The purpose of lowering the priority of a user who completes a time slice is to prevent "compute bound" users from monopolizing the CPU. However, to ensure that such users do not fail to run at all, each 15 seconds DISPATCH locates all runnable users who have not run during the preceding 15 seconds and raises them one priority level.

2. Dynamic Adjustment of Paging Load

In addition to execution control, DISPATCH is responsible for the dynamic adjustment of the system paging load. Each time DISPATCH is entered the following information is recorded:

1. Time the CPU was idle with outstanding page requests (page-blocked idle time).
2. Time spent executing CP.
3. Time spent executing problem programs.

Every 60 seconds the ratio of page-blocked idle time plus CP time plus problem time to page-blocked idle time is computed. This ratio provides a measure of the system paging load. If this ratio is greater than five, indicating a low paging load, the value of PAGMUT (maximum number of concurrent paging operations allowed) is increased by one. If however, this ratio is less than two, the paging load is undesirably high and, therefore, PAGMUT is decreased by one. This adjustment is subject to the limitation that PAGMUT must always be greater than or equal to two.

3. Main Storage Management

Another resource of the real system which must be shared among the virtual systems is main storage. To facilitate this process the 256K bytes of main storage are divided into 64 equal 4K blocks, each

of which can hold one page of program or data. Seventeen of these core blocks are permanently occupied by CP-67 itself, and another seven are filled with the nucleus of CMS, leaving 40 blocks available for sharing among virtual machines. These core blocks are shared using the technique known as paging.

If a user program (any program other than the supervisor, CP) references a page which is not currently core resident, dynamic address translation will fail and a page-translation interrupt will occur. This paging interrupt results in a call to PAGTRANS, the page handling routine. PAGTRANS first determines if a new paging operation can be initiated. The value of PAGMUT is compared with PAGARP, the number of current paging operations. If PAGARP is greater than or equal to PAGMUT no new paging operations may be initiated and the page request is added to the queue of pending page requests, PAGMUTQ. (Servicing this queue is one of DISPATCH's "housekeeping chores".) Control is then returned to DISPATCH. If, however, PAGARP is less than PAGMUT the value of PAGARP is incremented and the paging operation is begun.

PAGTRANS then must locate an available core block into which the required page may be read. This is accomplished by examining the core table which contains an entry for each of the 64 core blocks indicating the status of the page currently occupying that block. Blocks are selected for paging subject to the following constraints:

1. No block which is locked (i.e., affected by a pending I/O operation) may be selected.

2. No block which is "in transit" (i.e., affected by a pending paging operation) may be selected.

3. No block having a FIFO flag set may be selected. FIFO flags are set whenever a page is read into a block. When all FIFO flags are found to be set they are all reset immediately.

Blocks meeting the constraints listed above are selected in the following order of priority:

1. Non-valid (i.e., currently empty)
2. Unused and unchanged
3. Unused and changed
4. Used and unchanged
5. Used and changed.

When an available core block is located, the transit, valid and FIFO flags are set in the corresponding core table entry.

If the changed flag is set in the selected core table entry, the page currently occupying that core block must be written into its direct access storage device location before the new page can be read. This process is known as page "swapping". PAGTRANS determines the appropriate DASD address, generates an I/O task block to write the page, and adds this block to the queue of I/O task requests. After this is completed, or if it were unnecessary, PAGTRANS creates an I/O task block to read the requested virtual page from its DASD location and queues this block for execution. Control then is returned to DISPATCH.

When the requested page is read into its core block the transit flag is cleared from the corresponding core table entry and the requesting user again is made runnable.

4. I/O Management

All virtual computer I/O operations must be translated into real I/O operations and scheduled by CP. Since CP/CMS is a disk file oriented system and because all unit record I/O is spooled on disk by CP, virtually all input and output may be considered to be disk I/O.

All I/O instructions are privileged; that is, they may be executed only by a supervisory program. Consequently, any attempt by a virtual machine to execute an I/O instruction leads to a privileged operation interrupt and a transfer of control to CP. CP translates virtual device addresses into real device addresses, virtual storage addresses into real storage addresses, and virtual channel command lists into real channel command lists. It also locks the affected page(s) into core, places the requesting virtual computer in an IOWAIT status, queues an appropriate I/O task block for execution, and returns control to DISPATCH. When CP receives an interrupt indicating the completion of an I/O operation it unlocks the affected page(s), makes the requesting user's virtual machine runnable, and returns control to DISPATCH.

III. THE MODEL

A. MODEL DESIGN

A flexible GPSS model was constructed within which changes in system configuration could easily be incorporated. To accomplish this required a rather large number of functions and variables, the use of which resulted in a corresponding increase in the execution time of the program. However, it is felt that the benefits which arise as a consequence of being more general far outweigh the disadvantage of somewhat longer execution times. This is particularly true when an experiment is being conducted which involves the examination of a number of alternatives. In this case it is desirable that changes be accomplished merely by modifying some parameters.

The model also was constructed in a modular fashion. Those routines which simulate DISPATCH, PAGTRANS, and IOEXEC, for example, are essentially separate sections. Thus, to reflect changes in the paging algorithm it is necessary to modify or replace only that section of code which deals with paging.

Depending upon the system configuration and the job mix, an execution ratio of between one-to-one and one-to-three was obtained. Thus, one minute of 360/67 time was needed to simulate one to three minutes of time on the system being modeled. Using results provided by Nielsen, and others, in similar studies it is judged that this ratio is not unacceptable [Ref. 11]. To assist in reducing overall execution times, user chains were provided in appropriate locations throughout the program. In addition, an experiment was performed to determine precisely how much

time could be saved by significantly reducing the amount of coding in the paging section, one of the lengthier sections in the program. This is discussed more fully under "level of detail", later in this section.

B. GENERAL DESCRIPTION

A GPSS program consisting of 249 blocks, 47 variables, and ten functions comprises the model of the time-sharing system. The program consists of the following sections:

1. Control information
2. Function description
3. Variable description
4. Timing, accumulation of statistics, output, and simulation of DISPATCH routines
5. Pre-execution
6. Execution
7. Post-execution
8. Miscellaneous routines
9. Input/Output
10. Paging.

The operation of each of these parts is discussed in detail later. Functions and variables describing the distributions of (1) execution time; (2) program size; and (3) user reaction time were obtained from a study by Scherr at the Massachusetts Institute of Technology [Ref. 12]. In obtaining these empirical distributions, Scherr used an extremely large sample size which lends some measure of credence to the statistics. The supposition that these distributions were valid for use in this study was based upon the consideration that one would not expect the characteristic requirements of programs written at M.I.T. to be vastly different from those written at this school, since both are educational environments where the majority of programmers are students.

C. JOB DESCRIPTION

Jobs resulting from terminal interactions are represented by "transactions", temporary GPSS entities. Jobs are completely described by means of their parameters, which are explained in Table III.

Apart from those created for special purposes, such as interrupt processing and timing, the number of transactions, and hence the number of jobs active in the system, is equal to the number of terminals. That is, it was assumed that all terminals had "logged on" and were in use. Clearly, this situation will not always be prevalent in the real system; however, it was decided to err on the side of conservatism and portray the system under the most demanding conditions in order to ensure that any conclusions about system performance were based upon "worst case" estimates.

Jobs have exponentially distributed execution times. For all runs, 40% were designated as "large" jobs with a mean execution time of 2,000 milliseconds and 60% were considered "small" jobs having a mean execution time of 25 milliseconds. This results in the hyper-exponential distribution which has been used in other studies of this type. [Ref. 13]. At most, two GPSS cards need be modified to alter either the job mix or the mean execution times.

D. PAGE ACCOUNTING

Two 40 word arrays were established to keep track of every page in real core. These arrays, with typical contents, are illustrated in Figure 2. Array A, halfword savevalues 1-40, contains an up-to-date list of all pages currently in real core, exclusive of the 24 blocks assigned to the core resident portions of CP/CMS. Page numbers identify

TABLE III

JOB PARAMETER DESCRIPTION

PARAMETER	DESCRIPTION
1	Indicates the time remaining before the next scheduled request for a page.
2	Indicates the time remaining before the next scheduled request for I/O.
3	Contains the time remaining in the time slice currently allotted.
4	Initially, contains the total CPU time required for completion of the job. This number is decremented by an appropriate amount each time the job gains control of the CPU.
5	Contains the length of a time slice which is determined on the basis of priority.
6	Contains the remaining time the job was scheduled to have control of the CPU when interrupted. This number is used to update parameters one through four to reflect the current status of the job.
7	Contains the terminal number.
8-9	Pointers for arrays and logic switches.
10	Points to one of the first four parameters and, by indirect addressing, is used to determine the amount of time a job will spend in the CPU (ADVANCE P*10). This parameter also is used as an index upon completion of a unit of processing to select the correct destination for the job.
11	Contains either a one or a zero and is used to adjust the priority of a job to its correct level.
12	Used to record the correct priority of a job.

TABLE III (continued)

PARAMETER	DESCRIPTION
13	A pointer to one of two variables which, initially, are used to select the amount of CPU time a job will require.
14	A pointer used to designate the I/O channel for the job.
15	A pointer to a savevalue location.
16	Contains either a one or a two. Used to indicate the number of times a job will loop during the paging operation.
17	Indicates the program size of a job.
18	A pointer to the core status table.
19	Contains the number of the page which currently is required by the job.
20	A counter used in looping to reset the FIFO indicators.
21	A pointer to the core status table.
22	A pointer to one of two variables. Used to select the time to next page request.
23	A pointer to one of two variables. Used to select the time to next I/O request.
24	Contains the number of the disk being referenced.
25	A pointer to the core status table.
26	Contains either a one or a two which indicates in IOWAT whether the job is doing I/O or paging from a disk.

HALFWORD SAVEVALUE LOCATION	CONTENT	HALFWORD SAVEVALUE LOCATION	CONTENT
1	1205	41	10001
2	113	42	00011
3	707	43	10101
4	1209	44	00011
.	.	.	.
.	.	.	.
.	.	.	.
39	1202	79	10011
40	801	80	10001
	A		B

FIGURE 2

CORE STATUS TABLES

both the user and a particular page for that user. As an example, the entry 1205 refers to the fifth page for the user on terminal 12. It was assumed that the user's requirements would not exceed 99 pages; otherwise fullword savevalue arrays would have been required. Array B, halfword savevalues 41-80, contains the status of the associated page in array A; that is, a sequence of zeroes and ones whose location signifies a particular indicator and whose value represents whether or not the indicator has been set (0-reset, 1-set). The indicators, from left to right, are:

1. Locked. A one in this position indicates the page has been locked in core, which implies that this particular core location may not be used for paging at this time.

2. Transit. A one signifies that the page is in transit from a DASD. This block of core may not be used for paging until the transit indicator is reset.

3. FIFO. FIFO indicators are set when a page is read into that core block. Once all FIFO indicators have been set, they are reset immediately. No core block having the FIFO flag set may be selected for paging.

4. Changed/Unchanged. This indicator is used to indicate whether or not a particular page has been altered during execution. If this indicator is set, the page must be written on its DASD before another page can be read into its core block. A reset indicator permits a read-only operation since an exact copy of the old page already resides on its DASD.

5. Valid/Non-valid. A reset indicator means the core block is unoccupied.

E. DETAILED MODEL DESCRIPTION.

A detailed explanation of the individual GPSS instructions is contained in the GPSS User's Manual [Ref. 7].

1. Control Information

This section of code is comprised of the RMULT card and the EQU cards. The RMULT card serves the purpose of initializing the seeds for the eight pseudo-random number generators, thereby minimizing the

possibility of spurious effects due to periodicity which otherwise might occur as a result of the eight generators producing the same sequence of numbers.

The EQU cards enable the programmer to use symbolic names, in place of numbers, in the operand fields of instructions when referring to GPSS entities. This feature not only makes the program itself more readable but also enhances the appearance of output data.

2. Function Description

a. NRPAG

This function is used in conjunction with the variable SIZE to assign to each job a program size of 1-57 pages. The mean and median of this distribution are approximately eight pages and two pages respectively.

b. REACT

This function is used in conjunction with the variable THINK to assign to each job a user reaction time. This distribution has a mean of 35 seconds and a range of 0-5 minutes.

c. CPUTI

This function points to one of two variables, CPUHI or CPULO, which designate the amount of CPU time a job will require.

d. PAGTI and IOTIM

These functions are used in a manner similar to CPUTI and assign to jobs a length of time which is interpreted as the interval between successive requests for pages and I/O respectively.

e. PGDSK

This function assigns to each job a disk for paging and is unused with less than 16 terminals active.

f. DISK

The function DISK selects a disk for each I/O request.

g. SEEK

This function describes the distribution of seek times for a 2311 disk.

h. EXPON

EXPON is the standard exponential distribution.

i. TIMER

This function is used with the variables TIMES and BTS to provide each job with a priority dependent time slice.

3. Variable Description

Variables are described by comments in the program listing.

4. Timing, Accumulation of Statistics, Output, and Simulation of DISPATCH Routines

A transaction is created every 15 seconds. After counters have been incremented, the 15-second check of runnable users is performed. Upon completion of this check, the contents of appropriate savevalue locations are updated. These savevalues are used to maintain information relating to (1) execution, overhead, and idle percentages; (2) paging rates; and (3) problem mode time, control program time, idle time, and page-blocked idle time. Once the warm-up period has elapsed, appropriate statistics, such as facility utilizations and queue lengths, are printed every 15 seconds. At the block labeled MINUT a test is made to see if the 60-second check should be performed. Clearly, every fourth transaction will cause this check to be made. Finally, the transaction is terminated and the termination count is decremented by one. Two features of this section of code should be noted. First, no blocks exist which, in any way, can delay the transaction and, as a result, the relative and absolute clocks

are not advanced. Second, the TERMINATE block at the end is one of only two places in the entire program where the termination count is decremented; the other being at the block labeled GOODO which is not accessible until sufficient statistics have been gathered to end the run.

5. Pre-processing

A transaction is created for each terminal on the system. These transactions symbolize "jobs" to the computer system and circulate throughout the main sections of the program simulating processing, creating interrupts, and initiating requests for pages and I/O as their parameters dictate. There are two large loops within the main program. The outer loop commences with the block labeled DOIT and ends at one of several blocks in the post-execution phase. Each job traverses the outer loop only once. Each transaction which leaves the ADVANCE block at DOIT represents a new job and from that time until its processing is completed circulates in the inner loop which commences at the block labeled EXEC and ends at a number of locations depending upon the status of the job.

Upon departing the ADVANCE block at DOIT, transactions set a logic switch which is used to indicate that they are not eligible for a priority increase. The status of the logic switch is examined during the aforementioned 15-second check. Jobs then are assigned (1) the amount of CPU time required for processing; (2) the program size in pages; (3) I/O and paging rates; (4) times to first I/O and page requests, and (5) a basic time slice of 50 milliseconds.

After pointers to logic switches have been initialized, the job is assigned a channel for I/O and disk paging. An interrupt then is created as a result of the terminal interaction and the priority of the job is set equal to the datum, or reference level, of 100. It should be

noted that all transactions which simulate interrupts are created with a priority of 125 to ensure their prompt processing. (Access to the CPU is made on the basis of priority.) The BUFFER block restarts the Overall GPSS/360 Scan in order to handle the interrupts which, otherwise, would not be processed until the status change flag was set to ON. The operation of the Overall GPSS/360 Scan is described in the GPSS User's Manual [Ref. 7]. A MARK block then is encountered which records the time the job was created. The combination of this MARK block and a succeeding TABULATE block with M1 in operand field A serves to compute the transit time for the job.

6. Processing

The execution phase, as well as the inner loop of the main program, commences with the block labeled EXEC. The next higher priority level is recorded and will be used in the event the job qualifies for a priority increase during the next 15-second check. One of the conditions for a priority increase then is established by setting a logic switch to indicate that the job is "runnable". This switch is reset immediately when a job obtains control of the CPU.

Jobs compete for and gain control of the CPU on the basis of priority. Once the CPU is obtained, a test is made to determine whether or not the current job was previously interrupted. If so, the job's priority is lowered by one to offset the temporary priority increase received at the time of interruption. That temporary priority increase ensures that the interrupted job will be given preference over all other jobs in its priority class when it returns to the queue for the CPU.

A check then is made to decide whether or not the job's desired page is in real core. If so, processing is allowed to continue; if not, the job relinquishes control of the processor and transfers to the paging

section to initiate a request for the page. Next, an arbitrary percentage of jobs indicate that their current page will be changed during execution which necessitates swapping once a request is made to bring another page into that particular block of real core. This percentage was maintained at 75% throughout all runs.

A priority dependent time slice then is computed and assigned to the job with the provision that this job is not the last interrupted user. If so, the job continues to process with the remaining portion of the time slice it had when interrupted.

The contents of parameters one through four are retained in save-values so that they may be adjusted after a unit of processing has been completed. These parameters contain, in order, (1) time to next page request; (2) time to next I/O request; (3) time remaining in the current time slice; and (4) overall CPU time remaining before the job is completed. The minimum of these four times is selected and the job enters the ADVANCE block with this time. If the job is interrupted prior to taking a normal exit from the ADVANCE block it is sent to the block labeled HOLD, which will be described later. If, however, the job is not interrupted, problem mode time and parameters one through four are adjusted to reflect the amount of time the job had control of the CPU. Control of the processor then is relinquished.

7. Post-processing

This section of code begins with the block labeled NEXTB. In order for a transaction to arrive at this location at least one of its first four parameters must be equal to zero. The number of this zero-valued parameter is contained in parameter ten. The first TRANSFER block sends the transaction to one of the next four blocks depending upon the value of parameter

ten. Three of these four blocks are unconditional transfers to various routines which handle such matters as paging and I/O. If the transaction is sent to the fourth block this signals completion of its required processing and an interrupt is created following the priority increase. If the stabilization period is not over, jobs are sent to DOIT and begin the cycle over again; otherwise transit time is tabulated in the TOTAL table before the transfer to DOIT.

Average transit time is calculated and printed for every five jobs. Finally, if sufficient statistics have not been gathered the transaction is sent back to DOIT. If the specified sample size has been reached, the TOTAL table is printed and all remaining transactions are terminated immediately by means of the blocks labeled COMPL and GOODO, thus completing the run.

8. Miscellaneous Routines

Four miscellaneous routines will be discussed.

a. DEMOT

This routine, commencing with the block labeled DEMOT, is entered from the post-execution phase when parameter three equals zero indicating the job has completed a time slice. The priority of the transaction immediately is lowered one level, a transaction is created which simulates a timer interrupt, and the job is sent back to the queue for the CPU.

b. HOLD

This routine commences with the block labeled HOLD and is entered by transactions which have been interrupted during execution. The source from which the transaction came is determined by a TEST block, the test being made on the basis of priority. If the transaction came

from the IDLER routine, page-blocked idle time is adjusted accordingly and the job is sent back to IDLER. Otherwise, problem mode time is adjusted together with parameters one through four, the transaction's priority is temporarily raised, a flag is set to indicate this job as the last interrupted user and the job is sent back to the queue for the CPU.

c. INT

This routine simulates the processing of interrupts created by a SPLIT block. Interrupts are handled on a FIFO basis. After control of the CPU has been obtained the transaction advances the length of time required to process the interrupt, adjusts control program time, relinquishes the CPU, and terminates without decrementing the termination count.

d. IDLER

A single transaction with a priority of one is generated at the start of the run. This transaction initializes the value of PGMUT to ten. PGMUT, the maximum number of pages allowed in transit, is adjusted dynamically during program execution by means of the 60-second check. This transaction then cycles back and forth between the HOLD routine and the IDLER routine. The only function of the IDLER section is to maintain a record of page-blocked idle time; that is, the time that the CPU is idle during which at least one page request is outstanding and remains to be honored. This is accomplished by means of the TEST GE, GATE NU, and TRANSFER blocks which require that two conditions be satisfied simultaneously before this transaction with its low priority is allowed to gain control of the CPU. Once these conditions are met, the transaction obtains the CPU and attempts to advance for essentially an

unlimited length of time. However, the amount of activity in the model prohibits normal exit from the ADVANCE block. When this transaction is interrupted it will be sent to HOLD where page-blocked idle time will be adjusted. This cycle is repeated until the run is completed.

9. Input/Output

This section of code, commencing with the block labeled IOWAT, is entered from the post-execution phase when parameter two equals zero indicating a request for I/O. A flag is set to denote an I/O request. This is necessary because some of the code in this section also is used by jobs which are paging from a disk. After creating an interrupt, a new time is obtained to the next I/O request. The appropriate page is locked in core, to be unlocked only upon completion of I/O. Then, a disk is assigned and the time required for the I/O to be accomplished is placed in parameter 13. Jobs then queue for the disk, or arm, and wait for the channel to become free. Once the channel is obtained a seek is initiated. Upon completion of the seek the job re-enters the queue for the channel. When control of the channel is obtained once more the I/O request is honored. The value of parameter 26 is tested to determine the source of the transaction. If the transaction was paging from a disk it is returned to the block labeled TEST in the paging section. If the transaction was doing I/O the page is unlocked and, after an interrupt is created, the job is sent back to queue for the processor.

10. Paging

This section of code commences with the block labeled PAGEW and is entered from either the execution or the post-execution phase. Due to the many options which exist in this section of code only one case will be described in detail. An attempt to explain all the possible situations that may arise would result, it is felt, in a mass of confusion. The case

which will be discussed is typical; a job requests a page which is not in real core, a block of real core is available for paging but the old page has been changed and must be swapped, and the requested page resides on the CMS disk. Alternatives to the case described can be understood by examining the source code in combination with the flowcharts.

Initially, the job receives a new time to the next page request and sets a flag indicating that it is paging, as opposed to performing I/O. After an interrupt is created, the number of the desired page is calculated and recorded to facilitate the re-checking which is required in the execution phase. Next, the core status table is scanned by means of the SELECTE block to determine if the requested page already is in core. Under the assumption that it is not in core the exit to FNDPG is taken.

At FNDPG a check is made to ensure that the maximum number of pages are not already in transit. If not, this number is incremented and the job examines the core status table by means of the SELECTMIN block to locate the most desirable core location for paging. Then, the status of that core block is settled by means of the two TEST blocks. Assuming that no problems exist the exit to OKGO is taken, at which point the test is made which indicates that the old page had been changed. Parameter 16 is set equal to two implying that two steps are required. First, the old page must be written onto the drum or disk, the determination of which is made in the next TEST block. Second, the requested page must be read into core. Assuming that the old page was obtained from the drum the exit to NORM is taken. The job then queues for the drum selector channel, writes the old page onto the drum once the channel is

seized, releases the channel, and transfers unconditionally to the block labeled TEST. The time spent writing the page onto the drum is uniform over the interval 3-21 milliseconds, a time interval which is characteristic of the 2301 drum. (See Table I)

At TEST an interrupt is created and the value of parameter 16 is shown to be two by the TEST E block. This number is decremented indicating that half the cycle of swapping has been completed.

The TRANSFER block arbitrarily designates 10% of the requests as CMS page requests. This percentage also was maintained constant throughout all runs. The number of the CMS disk is assigned to parameter 24 and an unconditional transfer is made to DISK+1. Disk paging time is calculated and the transaction then is sent to DSKIO in the I/O section for processing. Once the desired page has been read into core, the transaction returns to the block labeled TEST, at which time another interrupt is created. The value of parameter 16 is now one and, hence, the transaction is sent to the block labeled DONE.

At DONE the number of pages in transit is decremented by one, the scan is restarted, the transit indicator is reset and the job returns to EXEC to queue for the CPU.

Several special cases should be noted.

a. Core Block Unavailable for Paging

If the scan of the core status table indicates that all pages in core have been locked or designated as "in transit" the job is sent to a FIFO user chain, PROB, to await removal of one of these conditions.

b. Page Already in Core

If the requested page already is core resident then only that section of code between PAGEW and FNDPG will be executed.

c. All FIFO Indicators Set

If all FIFO indicators have been set the transaction, before continuing to be processed, resets all the indicators. This is accomplished by means of the three blocks of code starting with REFBT.

d. Maximum Number of Pages in Transit

If the desired page is not core resident and the number of pages in transit is already at the maximum permitted the job will be sent to another FIFO user chain, PGMTQ, to wait until some job has finished paging and decrements the number of pages in transit.

F. USING THE MODEL

An example of the output obtained from the GPSS program each 15 seconds of simulated time is included with the computer output. The first statistics presented are those concerned with CPU utilization and paging activity, and are represented as contents of fullword save-values.

The first three savevalues are titled appropriately and give the percentage overhead, percentage execution, and percentage idle for the CPU. These figures are truncated to tenths of one percent. PAGIN and PGOUT give the pages per second transferred into and out of core, respectively. Next, the standard GPSS facility and queue statistics are presented for the I/O channel (CHANA), the CMS disk (DISK1), the paging disk (DISK2), the I/O disks (DISK3 and DISK4), the processor (CPU), and the drum and its channel (CHAN2). Finally, the average response times, expressed in milliseconds, for two groups of five jobs are printed as the two values of savevalue location 322.

The next two pages of computer output contain examples of standard GPSS table output. During preliminary runs much use was made of data tabulated in this form. The first table presented, PGS, contains an observed distribution of program size in pages. The second table presents an observed distribution of 2311 disk seek times.

For a detailed explanation of the standard GPSS output see Ref. 7.

G. LEVEL OF DETAIL

In general, a model is not intended to reflect the most minute detail of the real system it purports to represent; time considerations alone seldom permit this. Consequently, delays in response due to dynamic address translation, for example, were not included in the model since, as Nielsen points out, "These functions are an order of magnitude finer in detail than the activities at the paging level." [Ref. 11].

It is essential, therefore, to extract from the real system those factors and relationships which have a significant effect upon the performance measure being examined and to disregard all unimportant details; keeping in mind, however, that what is deemed unimportant in one study may have vast implications in another.

Since the system being modeled is page oriented and since the operations of paging and page-swapping are vital factors in determining system response time, some provision was needed to keep track of every page in real core at all times. An explanation of the technique used is contained in a previous discussion on page accounting.

Because of the relatively large number of paging and I/O operations that take place over a small interval of time it was decided to use access times based on the probability distributions of seek time and rotational delay for the device in question rather than attempt to maintain a record

of the actual location of each page on that device. As they exist now, the paging and I/O sections of the model are quite lengthy. To complicate these sections further would only result in a poorer execution ratio without, it is felt, providing any additional information on system performance. The distribution of disk seek times was obtained from Ref. 1 and assumes that all cylinders have an equal probability of being referenced.

An experiment was conducted to measure the amount of execution time that could be saved by replacing the paging section with a reduced amount of coding which, hopefully, would create the same effect on overall system performance. To accomplish this, a pilot run was made to gather statistics on (1) the percentage of jobs whose desired page was already in core at the time a page request was initiated; and (2) the amount of time jobs spent paging when the desired page was not in core. By means of a TRANSFER block using the fractional mode, jobs, upon entering the paging section, were either sent back to the queue for the CPU (page in core) or allowed to page (page not in core). If paging was required the job entered an ADVANCE block whose field A argument was a function, the distribution of which was obtained in the pilot run. Upon departure from the ADVANCE block, jobs were sent back to the queue for the CPU. Thus, all that code relating to the maintenance of the core status tables was eliminated.

The effect of this seemingly major change was unexpected. Although the performance of the system was unaltered measurably, only one minute of execution was saved from a total of 25. Now, the saving of one minute of 360/67 time is not to be taken lightly and if there had been sufficient evidence to indicate that the statistics obtained from the pilot run were

applicable under all conditions this change would have been incorporated in the model. In order to acquire such evidence, however, additional pilot runs would have been necessary, resulting in an increase in the total computer time required to complete this study. Therefore, it was decided to return the code which had been removed and proceed with the study as originally intended. No further attempt was made to condense any portion of the program.

By examining block counts it was concluded that a large share of the execution time was spent either processing simulated interrupts or accumulating statistics. Unfortunately, interrupts occur individually and at rather unpredictable, although deterministic, times. Thus, each interrupt is a separate entity and, as such, must be processed as an individual task.

The standard set of GPSS statistics which is maintained automatically by the system can be obtained easily; so easily, in fact, that it is often more difficult to suppress this information than to acquire it. However, rather devious means must be employed to acquire statistics beyond those normally produced, especially if the statistic does not lend itself to tabulation in a table. The use of these unwieldy procedures consumes much execution time.

H. VALIDATION

Validation is the process of illustrating, by means of various statistical tests, that the behavior of the model conforms to the behavior of the real system under an equivalent load. That is, with a given system configuration and an identical input, or job mix, one tries to show that the variates which reflect some measure of performance in the model are drawn from the same distribution as those which reflect identical measures

of performance in the real system. Statistical tests such as goodness of fit may be employed.

The claim that a model is valid remains only conjecture until such time as statistical evidence is available to support that claim. The fact that the model, under a variety of tests, acts reasonably well can only be interpreted as encouraging. That is, at least the model did not yield any results which are known to be impossible to achieve in the real system. Whether or not the results are truly representative is another matter.

The dilemma that one faces, of course, is determining how much statistical evidence is required. Some may argue, incorrectly it is felt, that no amount of information will suffice to show validity. However, enough statistical evidence must be obtained to show that the model is adequate for the purposes of the study; adequate in the sense that those aspects of system behavior which are relevant to the study are reflected by the behavior of the model.

Unfortunately, it was not possible to even begin to validate the model used in this study, although an attempt was made to secure system performance figures from four different sources. Only one of these sources responded and that one was unable to provide any information which could be used for validation. Thus, it cannot be said with absolute certainty that the model really simulates CP/CMS, in spite of the fact that much care was taken to represent the system faithfully.

The importance of validation cannot be overemphasized. It is, perhaps, the single most important phase of model design and implementation. However, the lack of validation does not preclude using the model to make inferences about the system. To make such inferences one is forced to

resort to the use of sensitivity analysis as a tool; and, although a cogent argument can be made in its behalf, any results obtained by this method must be viewed with a measure of skepticism. Nevertheless, it was precisely this type of analysis that was employed in this study.

I. VERIFICATION

Before any simulation experiments are made, it is necessary to verify the model; in other words, to ensure that the model of the system logically functions as expected.

During preliminary program runs much use was made of the TRACE and SNAP options of GPSS. A close examination of the information provided by these features gave every indication that the model was behaving logically as intended. That is, interrupts were occurring at the proper point in time and were being processed as they occurred, the status of pages in core was being maintained properly, the algorithm for selecting CPU advance time was functioning correctly and, in general, all block counts appeared very reasonable. As expected, for example, no transactions were placed on the problem chain, PROB. Further, jobs in execution were transferred to the correct location either upon causing an interrupt or upon being interrupted.

Job execution times were tabulated and graphed and were hyper-exponentially distributed as intended, with mean execution times in the immediate neighborhood of the theoretical value.

The possibility of performing an analysis of the queueing behavior of the model based upon theoretical concepts was considered and rejected since a Poisson arrival rate cannot be assumed when arrivals are dependent upon previous results in the system [Ref. 10].

Chi-square tests for goodness of fit were applied to distributions obtained from the functions used by the program. In all cases, the hypothesis was accepted at the .05 level of significance that the empirical distribution fit the theoretical distribution. The results of the tests are tabulated in Table IV. The actual data are contained in Appendix A.

TABLE IV

RESULTS OF CHI-SQUARE TESTS
FOR
GOODNESS OF FIT

FUNCTION	DEGREES OF FREEDOM	NUMBER OF OBSERVATIONS	CRITICAL VALUE	COMPUTED VALUE
EXPON	20	412	31.4	17.8
NRPAG	12	380	21.0	6.74
REACT	19	374	30.1	26.5
SEEK	3	8511	7.81	1.25

IV. EXPERIMENTAL AND STATISTICAL CONSIDERATIONS

Once the model had been constructed it remained (1) to design the experiment so that a sufficient quantity of information was acquired in a form suitable for analysis; and (2) to statistically examine that information in order to make, or to substantiate, any conclusions about the system. A review of the literature indicated that several methods are available for analyzing data obtained in a simulation experiment. The purpose of this section is to explain and justify the method used herein.

A. EXPERIMENTAL CONSIDERATIONS

Response time was selected as the primary measure of system performance. It was not, therefore, the intent of this study to recommend procedures for better utilizing system resources, such as core or direct access storage devices, except as those procedures related directly to improving system response time, the user's principal concern. Within the framework of GPSS it was a simple matter to represent response time as an attribute of transactions.

Conway [Ref. 2] suggests three alternatives for measuring the attributes of temporary entities. The first method consists of fixing the duration of the simulation run and including in the sample the pertinent attribute for all transactions which terminate in that fixed interval. The second method is characterized by fixing both the starting time and the sample size and running the simulation until the desired number of observations is obtained, including in the sample the attributes of those transactions which were being processed by the system when the starting

time was achieved. The third method also fixes the starting time and the sample size but only considers the attributes for a continuous set of transactions which commenced processing after the starting time was achieved.

Method two was adopted; that is, the sample was composed of the attributes of the first thousand transactions to terminate after a specified time. Obviously, these transactions were not necessarily the first thousand to be created. Thus, the set of observations which comprised the sample was not determined precisely until the run actually ended. It is seen that this procedure may tend to bias the average transit time toward a more favorable result particularly if, in the model, it is characteristic that jobs requiring a large amount of CPU time never terminate during the course of the run. In the extreme, consider the case with 12 user terminals in which the first 11 jobs created require the maximum amount of processing time and, in addition, have high paging and I/O requirements. Thus, these 11 jobs will remain active in the system for a considerable period of time. It may be that the 12th job requires the minimum amount of CPU time and has virtually no I/O or paging requirements. It is conceivable that this job could terminate almost immediately and be replaced in the system by another job for that terminal with the same minimum requirements. Thus, the second job also may terminate quickly and it is possible that this cycle may continue until the specified number of observations is obtained with the result that the sample is restricted to only those jobs which place a light demand on the system. The researcher, having no reason to believe otherwise, may conclude, quite inaccurately, that the mean transit time is exceptionally good for all types of jobs. However, the use, in pilot runs,

of the TRACE and SNAP options enabled the authors to follow the active history of transactions which imposed a variety of demands upon the system, both heavy and light and to conclude that the situation described above does not occur in the model used to study CP/CMS.

Alternative one was not selected because a minimum sample size, although desired, could not be guaranteed using this strategy. Similarly, alternative three was rejected since it was not felt necessary to eliminate from the sample the relatively small number of transactions which were probably created in the waning seconds of the stabilization period. In addition, the primary concern was to obtain an estimate for the mean transit time and, therefore, the manner in which individual observations were grouped was of no consequence. Although Conway indicates that the third strategy should assist in reducing the variability of the results, it is debatable whether or not these hypothetical reductions would have been so beneficial in tightening confidence intervals as to warrant the expense of inserting additional code to keep track of sets of transactions throughout their active history.

The matter of model equilibrium was raised; that is, the measurement of attributes should not begin until the simulation model has reached some sort of steady-state, or stable condition. Clearly, the length of a suitable stabilization period in simulated time is a function of the amount of activity that occurs within the model during that simulated time. If the basic unit of time is milliseconds and if events are scheduled to occur within the model every few units of time, both of which conditions are satisfied in this model (jobs averaged about 6-8 milliseconds in the CPU during a unit of processing), then one certainly would expect the stabilization period to be expressed in terms of seconds or minutes.

Furthermore, for the model used in this study, it was not at all apparent that a stabilization period was even required since the system itself undergoes a transient period of relative inactivity every time operation is commenced. Unlike job shop models, or the like, in which jobs remain in the system after closing hours and are present when business re-convenes the following day, when it becomes necessary to terminate time-sharing services jobs in execution are effectively lost and no queues for system resources remain overnight. Thus, each day the system starts afresh in a somewhat empty and idle condition. However, since it would not be reasonable to judge the merits of a time-sharing system on the basis of the first few minutes of operation, it was decided to allow for a period of "warm-up".

Pilot runs were made in which one, two, three, four, and five minute stabilization periods were used. An examination of facility utilizations, queue lengths, and other statistics from these runs revealed that, in all cases, the model had stabilized by the end of the "warm-up" period. Therefore, although one minute would have been sufficient, a three minute period was chosen in order to be conservative.

Similarly, the choice of a suitable time interval for measuring the attributes of permanent entities became a matter of some concern. It is interesting to note that resolving these issues led to the method of statistical analysis described in the remainder of Section IV.

B. STATISTICAL CONSIDERATIONS

In discussing recommended procedures for measuring the attributes of permanent entities, Conway states:

> Most importantly, some check on the amount of correlation should always be made. This should be done during pilot runs to determine how measurements are going to be made and tested again during the actual

experimentation. Usually the same dynamic character of the system that forces one to resort to simulation in the first place ensures that correlation will be an important factor and problem in the investigation [Ref. 2].

With regard to the measurement of temporary entities, Conway says:

Again the difficulty lies in the auto-correlation of the observations and arises when one would use the sample variance to estimate the variance of the sample mean. To equate the variance of the mean with the sample variance divided by the number of observations contributing to the mean, requires that the observations be mutually independent and that is rarely the case. Temporary entites existing at the same time are subject to the same system conditions so that the values of the attributes tend to be positively and strongly correlated. The neglect of this correlation results in a substantial understatement of the variance of the mean [Ref. 2].

To reduce the amount of correlation between observations, and subsequently enhance precision, Conway recommends increasing the length of time intervals between successive measurements since, in general, a longer interval length will result in less correlation.

1. Preliminary Analysis

With all this in mind, the authors made the pilot runs referred to in the previous discussion to determine both a suitable stabilization period and an interval length which, on the one hand, would ensure equilibrium and yield essentially uncorrelated observations and, at the same time, could be accommodated in view of the high execution ratio.

The amount of correlation between adjacent observations, X_i and X_{i+1}, was estimated using the standard formula

$$\rho = \frac{(N-1)\Sigma X_i X_{i+1} - \Sigma X_i \, \Sigma X_{i+1}}{\sqrt{[(N-1)\Sigma X_i^2 - (\Sigma X_i)^2]\,[(N-1)\Sigma X_{i+1}^2 - (\Sigma X_{i+1})^2]}},$$

where the index, i, is summed from 1 to N-1 in all cases, N being the number of observations. Initial results were disappointing indicating

in some cases high correlation with relatively long intervals and low correlation with shorter intervals. There did not appear to be, however, any discernible consistency in the results, which were so disparate at times that it was conjectured the perfect random number generator, at long last, had been achieved.

At this stage, possible approaches to the problem were (1) to ignore auto-correlation completely; (2) to arbitrarily increase the lengths of intervals to the point that it was felt the effects of auto-correlation would be negligible; or (3) to perform a full analysis of the generated data in order to use, rather than ignore, the inherent auto-correlation.

The first approach is unsatisfactory since, especially in a scientific discipline, one at least should make an attempt to resolve pertinent issues rather than suspend them from consideration. Similarly, the second strategy tends to be nebulous and inconclusive, and may lead both to the discarding of valid data which should be included in the sample and to inordinately long computer runs. Thus, the third approach was taken, principally because it was the least unsatisfactory of the three from an experimental point of view. As it turned out, the same experimental results would have been obtained even if auto-correlation had been ignored completely, but this was not evident beforehand.

2. Spectral Analysis

If one were interested only in obtaining an estimate for the mean of a stochastic process the sample average would suffice, and it would be a simple matter to run the experiment until a specified number of observations had been acquired, calculate the sample average and quit. However, a problem arises when attempting to make probability statements

about this estimate of the mean, such as the calculation of a confidence interval. Traditional statistical methods generally apply to independent observations only and, hence, cannot be used to analyze the auto-correlated time series of data generated by a computer simulation model. The method of analysis used herein was obtained entirely from the works of Fishman and Kiviat, [Refs. 3 and 4], who point out that:

> The variance of the sample mean computed from a set of independent observations is inversely proportional to the number of observations. This is not true for auto-correlated data. For sufficiently long sample record lengths, however, one may show that the variance of the sample mean for auto-correlated data is inversely proportional to a fraction of the number of observations. This fractional factor depends on the auto-correlation properties of the process. By analogy with the independent case, it seems natural to regard this fraction of the number of observations as the number of equivalent independent observations.

To develop this analogy, we introduce the concept of the correlation time of a process. If a process is observed for a time interval equal to n correlation times, then one may show that, from the point of view of the variance of the sample mean, this time series is equivalent to collecting n/2 independent observations [Ref. 4].

It is the rule rather than the exception that a stochastic process will yield fluctuations about the mean, these fluctuations providing a measure of the variability of that mean. Furthermore, the deviations generally have frequency components which can be described by examining the entire spectrum of the generating process. The method discussed previously of measuring the amount of correlation between adjacent observations took into account only a portion of that spectrum and consequently yielded little information concerning the magnitude and period of any frequency components which may have contributed to the variance.

In general, highly auto-correlated time series tend to be relatively stable, or sluggish, in the sense that major fluctuations about the mean seem to have rather long periods and the process is said to have a long correlation time. Conversely, it is characteristic of less auto-correlated time series that observations will fluctuate rapidly, and with

varying proportions, in what appears to be an unpredictable fashion. Frequency components may seem to be less discernible and the correlation time is said to be short.

Spectral analysis, then, involves an examination of the observed data to determine the magnitude and period of any frequency components which contribute to the variance of the attribute being measured. On the basis of an estimate for the correlation time the number of equivalent independent observations is calculated. An approximate t-test can be applied to obtain confidence intervals.

The generation of data in this simulation experiment can be thought of as a time-dependent covariance stationary stochastic process $\{X_t, t = 0, 1, 2, \ldots\}$. Definitions obtained from Fishman and Kiviat include, the mean

$$\mu = E(X_t);$$

the autocovariance function

$$R_\tau = E[(X_t - \mu)\ (X_{t+\tau} - \mu)] \qquad \tau = 0, 1, 2 \cdots \infty;$$

the spectral density function

$$f(\lambda) = \frac{1}{\pi}\ [1 + 2 \sum_{\tau=1}^{\infty} (R_\tau/R_0) \cos \lambda\tau] \qquad 0 \leq \lambda \leq \pi;$$

and the spectrum

$$g(\lambda) = R_0 f(\lambda).$$

Clearly, R_0 is derived from the second moments of the stochastic process $\{X_t\}$ and it is seen that if $\{X_t\}$ has fixed, but possibly unknown, variance σ^2, then

$$R_0 = E[(X_t - \mu)^2] = \sigma^2.$$

In general

$$R_\tau = \sigma^2 \rho_\tau$$

where ρ_τ is the auto-correlation function which measures the influence of past events in the system on the present observations. The function ρ_τ lies in the closed interval [-1, 1] and equals unity when τ equals zero.

The parameter λ used in both the spectral density function and the spectrum is a measure of angular frequency. As mentioned previously, the process $\{X_t\}$ is thought of as being composed of a number of frequency components and the contribution of angular frequencies around λ to the variance σ^2 is expressed in terms of the spectral density function, $f(\lambda)$.

Considering one simulation run as generating a time series $\{X_t\}$ of length T, the sample average is given by

$$\bar{X}_t = \frac{1}{T} \sum_{t=1}^{T} X_t$$

and its variance is

$$\text{Var } (\bar{X}_t) = \frac{1}{T} [R_O + 2 \sum_{\tau=1}^{T-1} (1 - \tau/T) R_\tau].$$

From the expression

$$\lim_{T \to \infty} T \text{ Var}(\bar{X}_t) = \pi g(0) = \pi R_O f(0),$$

Fishman and Kiviat point out that the large sample variance of $\bar{X}_t$ can be approximated by

$$\text{Var } (\bar{X}_t) \sim 2R_O \tau^*/T,$$

where τ^*, the correlation time, is directly proportional to the amount of auto-correlation and is defined by

$$\tau^* = \pi f(0)/2.$$

The significance of τ^* can be seen by examining a special case. Assuming that the sequence $\{X_t\}$ is uncorrelated with variance σ^2, it is obvious that the variance of the sample mean, $\bar{X}_t$, is σ^2/N, or R_0/N, where N is the number of independent observations. By equating this variance with the variance for the auto-correlated case, it is seen that

$$R_0/N = 2R_0\,\tau^*/T,$$

so that

$$N = T/2\tau^*$$

which yields the number of equivalent independent observations in the auto-correlated sequence $\{X_t\}$ of length T with correlation time τ^*. Thus, the process of observing the auto-correlated sequence over the time interval $2\tau^*$ is equivalent to obtaining one independent observation.

Since there was no reason to expect that any two given simulation runs possessed the same amount of auto-correlation it was necessary to estimate separately for each run (1) the variance of the sample mean; (2) the correlation time; and (3) the number of equivalent independent observations. The following formulas were used to make these estimations:

$$\hat{R}_\tau = \frac{1}{T-\tau} \sum_{t=1}^{T-\tau} (X_t - \bar{X}_t)\,(X_{t+\tau} - \bar{X}_t),$$

$$\hat{V} = \frac{1}{T}\left[\hat{R}_0 + 2\sum_{\tau=1}^{M} (1 - \tau/M)\,\hat{R}_\tau\right] \qquad M < T-1,$$

$$\hat{\tau}^* = T\hat{V}/(2\hat{R}_0),$$

and

$$\hat{N} = T/(2\hat{\tau}^*) = \hat{R}_0/\hat{V}.$$

This estimate of the variance uses M instead of T-1 lags, or offsets, since the variance does not change measurably if more than a certain number, M, of the lags are used when the number of observations is large. The appropriate value for M was chosen empirically for each run in the manner discussed below.

Observations obtained in the simulation runs were supplied as input data to the analysis program, a listing of which appears following the appendices. This program calculated, for all possible values of M, estimates for the variance, the correlation time, and the number of equivalent independent observations. In addition, estimates of the variance for each value of M were plotted. An example of output from the analysis program, including the graph, appears following the appendices.

It can be seen by examining the sample output that the estimates of the variance increase in magnitude as M increases and reach a plateau of relative stability before finally diminishing. This increase in $\hat{V}$ is a reflection of poor resolution in the spectrum and is caused by using an inadequate number of lags. The effects of more frequency components are considered as the number of lags is increased and, as a result, estimates of the spectrum become more representative of the true spectrum. The subsequent decline of $\hat{V}$ represents a bias which is introduced by replacing the mean μ with the sample average $\bar{X}_t$ in the estimate of the autocovariance function $\hat{R}_\tau$. Fishman and Kiviat note that:

As M increases, the bias term clearly increases and causes a reduction in the estimate of V. To avoid this, it is necessary to keep the number of lags M significantly smaller than the sample record length T. Bear in mind, however, that good resolution requires that M be sufficiently large so that the function g changes slowly at frequency π/M [Ref. 4].

Using the output data as an example, it is seen that $\hat{V}$ stabilizes when M is in the vicinity of 29. At this point the estimate of the variance of $\bar{X}_t$ is 3.473 and about 98 equivalent independent observations resulted from this simulation run.

Approximate 90% confidence intervals for $\bar{X}_t$ were calculated for all simulation runs using the statistic

$$t = \frac{\bar{X}_t - \mu}{(2\hat{\sigma}^2\hat{\tau}^*/T)^{1/2}} = \frac{\bar{X}_t - \mu}{(\hat{V})^{1/2}},$$

which was shown to have an approximate Student t distribution with 1.5T/M equivalent degrees of freedom.

Tests for significant differences in means between two runs were made by using the statistic

$$t'_\alpha = \frac{(\bar{X}_{t,1} - \bar{X}_{t,2}) - (\mu_1 - \mu_2)}{(\hat{V}_1 + \hat{V}_2)^{1/2}},$$

which was shown to have a Student t distribution. The possibility that the equivalent degrees of freedom for the two runs, $1.5T_1/M_1$ and $1.5T_2/M_2$, were unequal required the calculation of t'_α by

$$t'_\alpha = \frac{(\hat{V}_1)^{1/2} t_{1,\alpha} + (\hat{V}_2)^{1/2} t_{2,\alpha}}{(\hat{V}_1)^{1/2} + (\hat{V}_2)^{1/2}},$$

where $t_{1,\alpha}$ and $t_{2,\alpha}$ are the critical values obtained from a table of the Student t distribution. The confidence interval is defined by the probability statement

$$P[(\bar{X}_{t,1} - \bar{X}_{t,2}) - t'_\alpha (\hat{V}_1 + \hat{V}_2)^{1/2} \leq \mu_1 - \mu_2 \leq (\bar{X}_{t,1} - \bar{X}_{t,2}) + t'_\alpha (\hat{V}_1 + \hat{V}_2)^{1/2}] \simeq 1 - \alpha$$

Throughout this analysis it was assumed that the sequence $\{X_t\}$ was Gaussian distributed. The validity of this assumption was based on the fact that all observations were obtained by averaging the transit time of five consecutively terminating transactions. The results of test runs indicated that when ten transactions were grouped to form one observation the presence of auto-correlation in the sequence $\{X_t\}$ was obscured. To use this method of analysis, however, it was necessary to be able to observe the auto-correlation which was present and, thus, the number of transactions constituting one observation was reduced to five. As a result of this reduction, a sequence of observations was obtained in which the presence of auto-correlation was readily apparent.

V. EXPERIMENTS AND RESULTS

In order to fulfill the objectives of this study three experiments were performed. Experiment A investigated the behavior of the system under the additional load resulting from an increase in the number of terminals supported. Experiment B was designed to determine if system performance could be improved significantly either by more efficient utilization of the existing hardware or by the installation of additional hardware. In Experiment C the effects of changes to the main scheduling algorithm in DISPATCH were explored.

Although response time was the primary measure of system performance, as mentioned in Section IV, other measures of performance were determined and are included in the tables of results in Appendix D. Among these measures is the Elapsed Time Multiplication Factor (ETMF) introduced by Stimler[Ref. 14]. For the purposes of this study the ETMF is defined to be the average response time under time-sharing divided by the average response time if all jobs were run on a "stand-alone" basis. Thus, as the performance of the system improves, the ETMF decreases, approaching a lower limit of one. The ETMF provides a measure of time-sharing system performance which differs from response time in one important respect. If the average amount of CPU time and/or I/O time required by a group of jobs increases, their average response time naturally increases by at least an equal amount, even if the system is completely efficient. On the other hand, these same increased requirements would not cause any increase in the ETMF. Thus, the ETMF provides a useful means of comparing system performance in experimental runs where the CPU and I/O rates differ. A discussion of how the ETMF was calculated here is given in Appendix C.

A. EXPERIMENT A

In order to measure system degradation under an increased load, four series of five runs each were made. Within each series the conditions were identical with the exception of the number of terminals which, after an initial run with 12 terminals, was raised from 15 to 30 in steps of five. Between series of runs the I/O rate and paging interrupt rate distributions were varied. The mean rates used in all experiments are tabulated in Table V.

TABLE V

MEAN I/O AND PAGING INTERRUPT RATES

DESCRIPTION	MEAN RATE	MEAN INTERVAL
LOW I/O	24.10/sec.	41.5 msec.
HIGH I/O	36.36/sec.	27.5 msec.
LOW PAGING	22.73/sec.	44.0 msec.
HIGH PAGING	38.46/sec.	26.0 msec.

The results of Series A1-A4, Experiment A, are summarized in Tables VI-IX respectively. Figures 3 and 4 present graphical displays of the observed response times and calculated ETMFs respectively. The cumulative distributions for response times of Series A3 are pictured in Figure 5. It was observed that in each series average response times increased as the number of terminals increased with the exception of the first two runs of Series A1, where the average response time decreased as the number of terminals was increased from 12 to 15. The differences between the average response times in each series were tested for significance using the method described in Section IV and, except

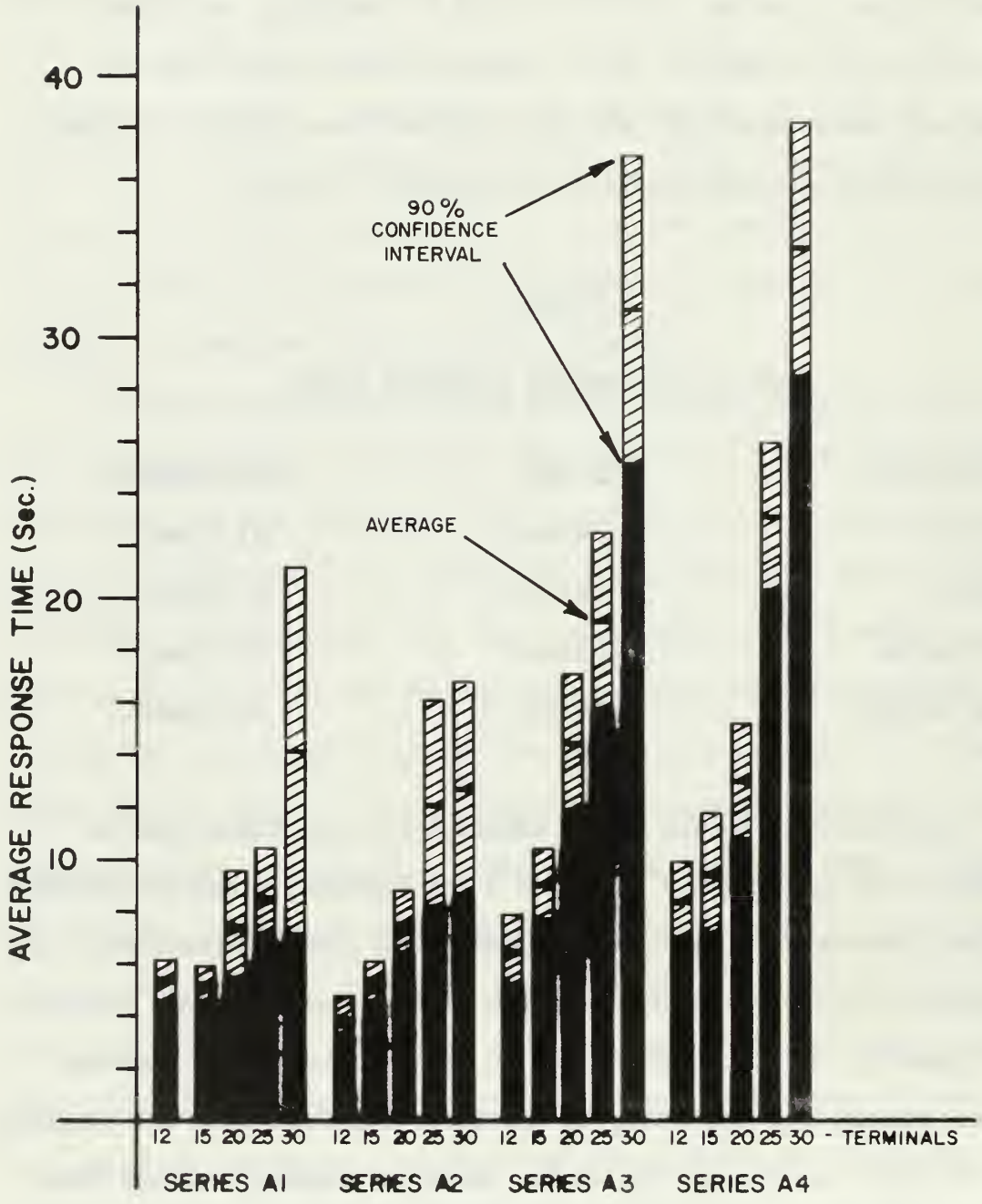


FIGURE 3

COMPARISON OF AVERAGE RESPONSE TIMES
FOR
SERIES A1-A4

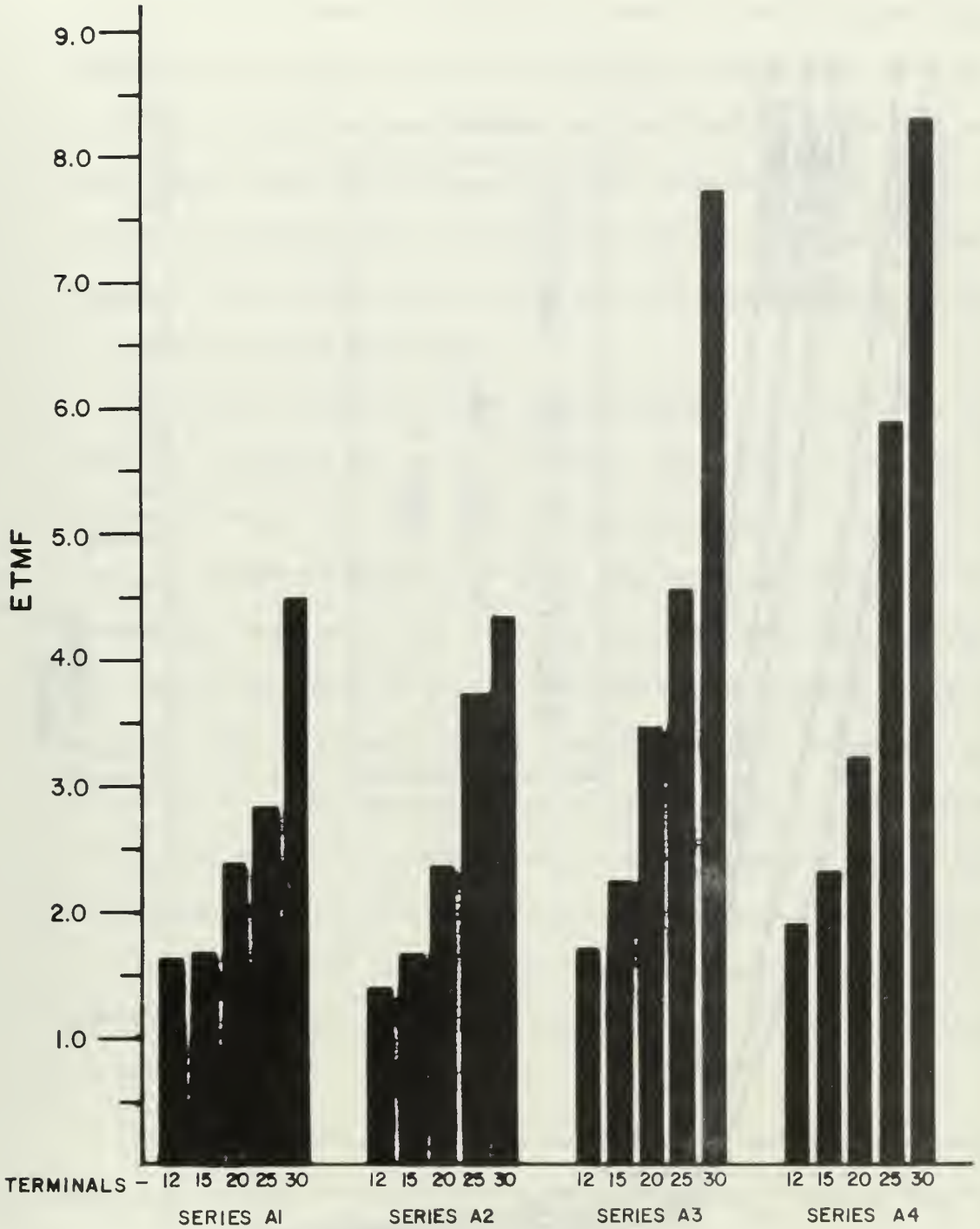


FIGURE 4

COMPARISON OF ETMFs
FOR
SERIES A1 - A4

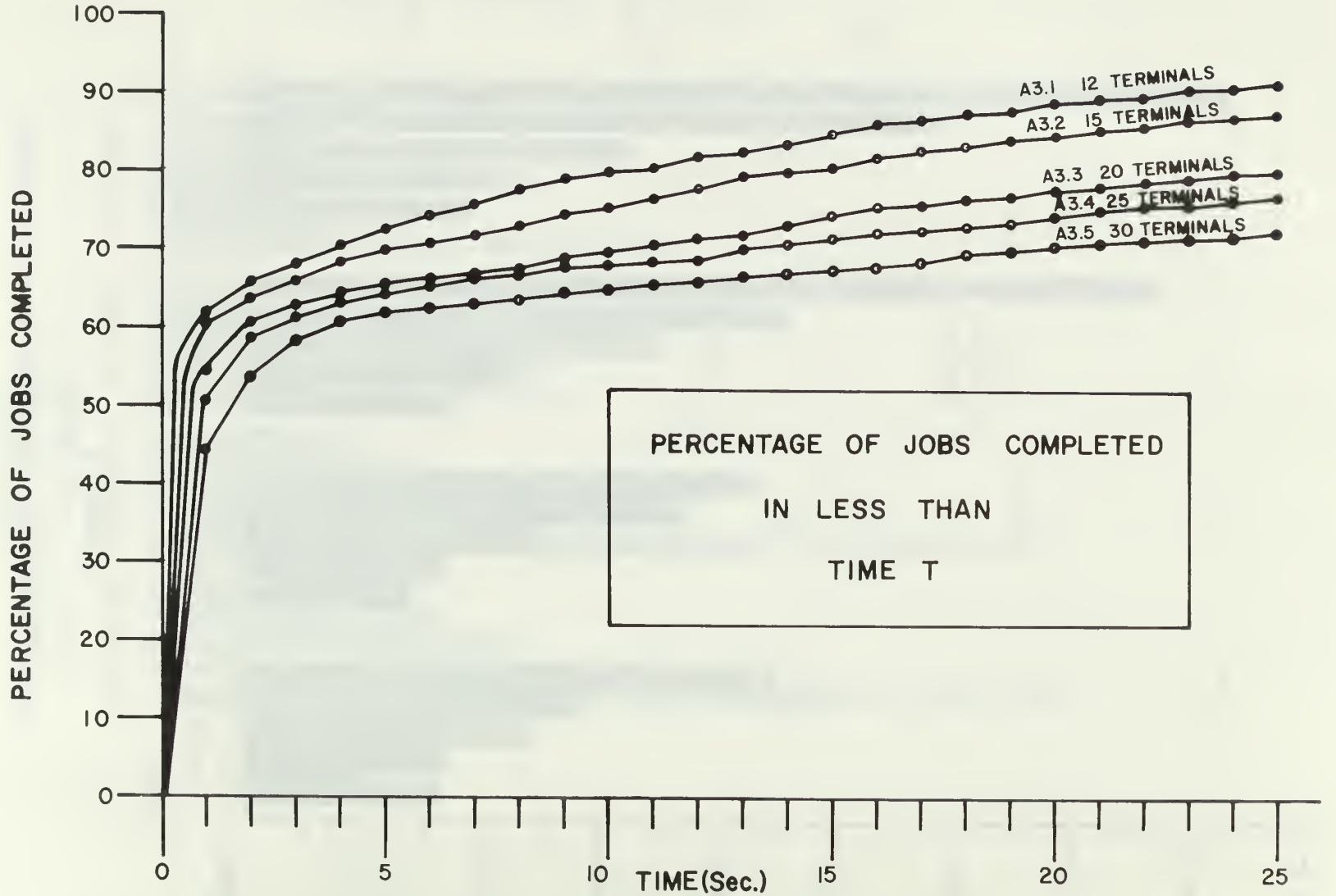

PERCENTAGE OF JOBS COMPLETED
100
90
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25
TIME(Sec.)
A3.1 12 TERMINALS
A3.2 15 TERMINALS
A3.3 20 TERMINALS
A3.4 25 TERMINALS
A3.5 30 TERMINALS
PERCENTAGE OF JOBS COMPLETED
IN LESS THAN
TIME T

FIGURE 5

for three, were found to be significant at the .10 level. The three differences which were not significant were (1) the decrease observed between runs A1.1 and A1.2; (2) the increase between runs A2.4 and A2.5; and (3) the increase between runs A4.1 and A4.2. It also was noted that, within each series, the ETMF increased as the number of terminals increased, with a much greater relative increase in Series A3 and A4, thus indicating a more pronounced degradation of performance when I/O rates were high.

The number of pages read per second increased as the number of terminals increased for all four series. This effect might have been attributed entirely to the fact that the percentage of execution and thus, the number of paging interrupts occurring per second also was increasing. However, it was observed after detailed analysis that even though the number of paging interrupts occurring per second of execution (paging interrupt rate) remained constant within each series, the number of pages read per second of execution increased considerably within each series. In other words, the percentage of paging interrupts which resulted in a page being read was increasing. Therefore, it was concluded that the expected increase in paging activity definitely was present and undoubtedly contributed to the increase in response time and ETMF.

Although all preliminary runs had indicated that the model was stable, Series A5 was run to verify the stability further. Each of the three runs was a duplicate of a previous run but with different seeds for the eight random number generators. Run A5.1 was a repeat of run A1.1; run A5.2 was a repeat of run A2.1; and run A5.3 was a repeat of run A2.5. The results of these runs are tabulated in Table X, Appendix D. As was expected, there were no marked differences between these runs and their

corresponding previous runs, and their average response times fell well within the 90% confidence intervals established by the previous runs. (See Figure 6.) Comparison of the corresponding average response times showed no significant differences at the .10 level. It also can be seen from Figure 7 that there were no appreciable differences in the corresponding ETMFs.

B. EXPERIMENT B

The purpose of Experiment B was to test whether or not an improve - ment in response time would result from (1) more efficient utilization of existing DASDs; or (2) addition of more DASDs. The first hypothesis was tested in Series B1 and the second in Series B2.

As noted earlier, the paging drum can store up to 15 virtual machines. Since a maximum of only 12 virtual machines is required with 12 terminals there exists room for approximately 200 additional pages on the drum. It was felt that response time might be improved if this space could be utilized to store the most frequently referenced pages from the CMS disk. It was determined that 40% of the disk resident CMS pages could be moved to the drum. Furthermore, it was assumed that these 200 pages could be chosen in such a manner as to account for at least 80% of the requests for CMS pages. The minor alterations to the model which were necessary to reflect this change to the system were incorporated in runs B1.1 through B1.4.

Another, perhaps more obvious, area for improvement was in the assignment of 2311 disk drives. With 15 or fewer terminals the paging disk was not used and, in addition, because the CMS disk was used to store only the 500 system pages, approximately 75% of its capacity was unused. The DISK function in the model was changed to make this unused space available for I/O operations in the runs of Series B1.

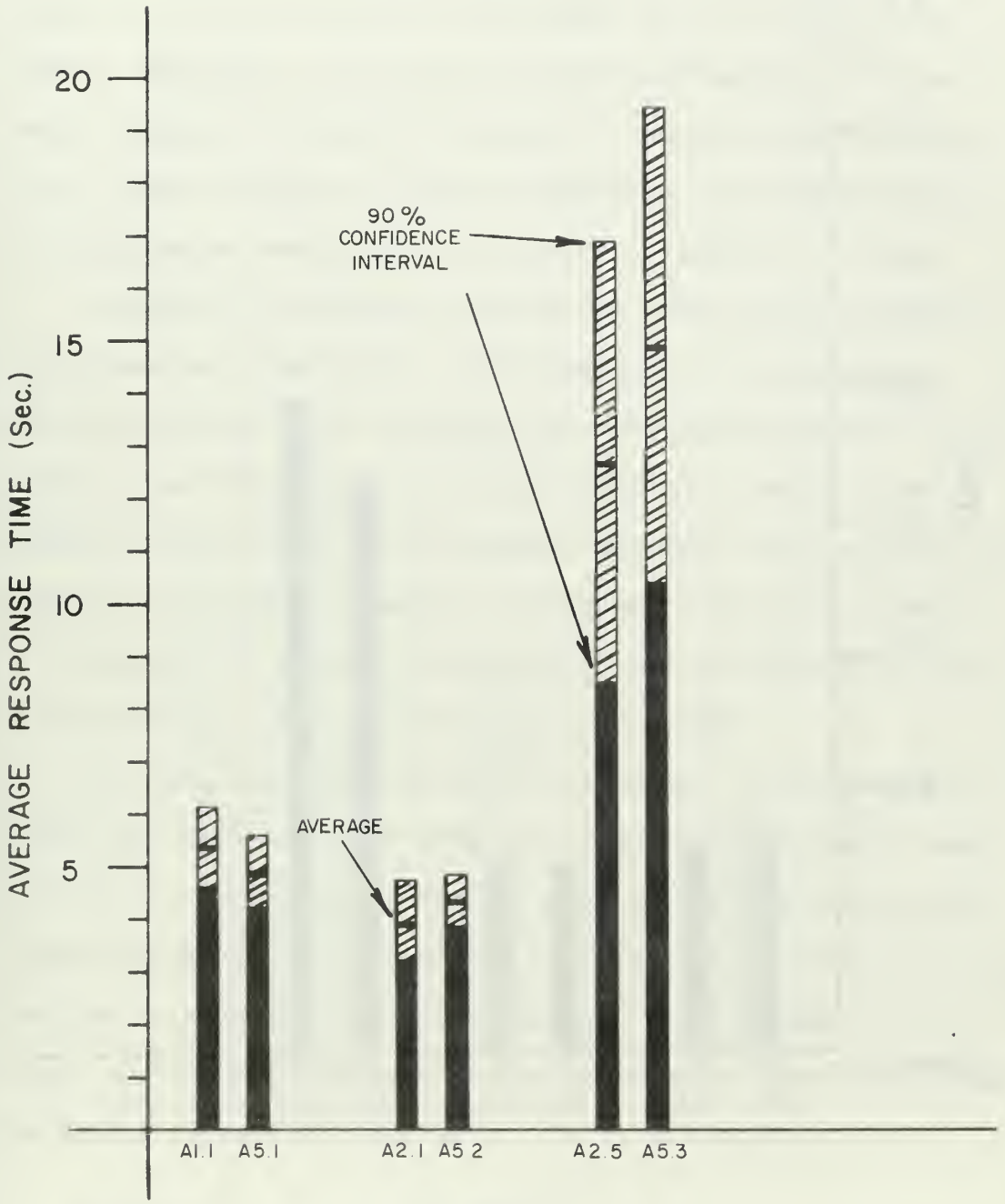


FIGURE 6

COMPARISON OF AVERAGE RESPONSE TIMES
FOR SERIES A5 AND
CORRESPONDING RUNS

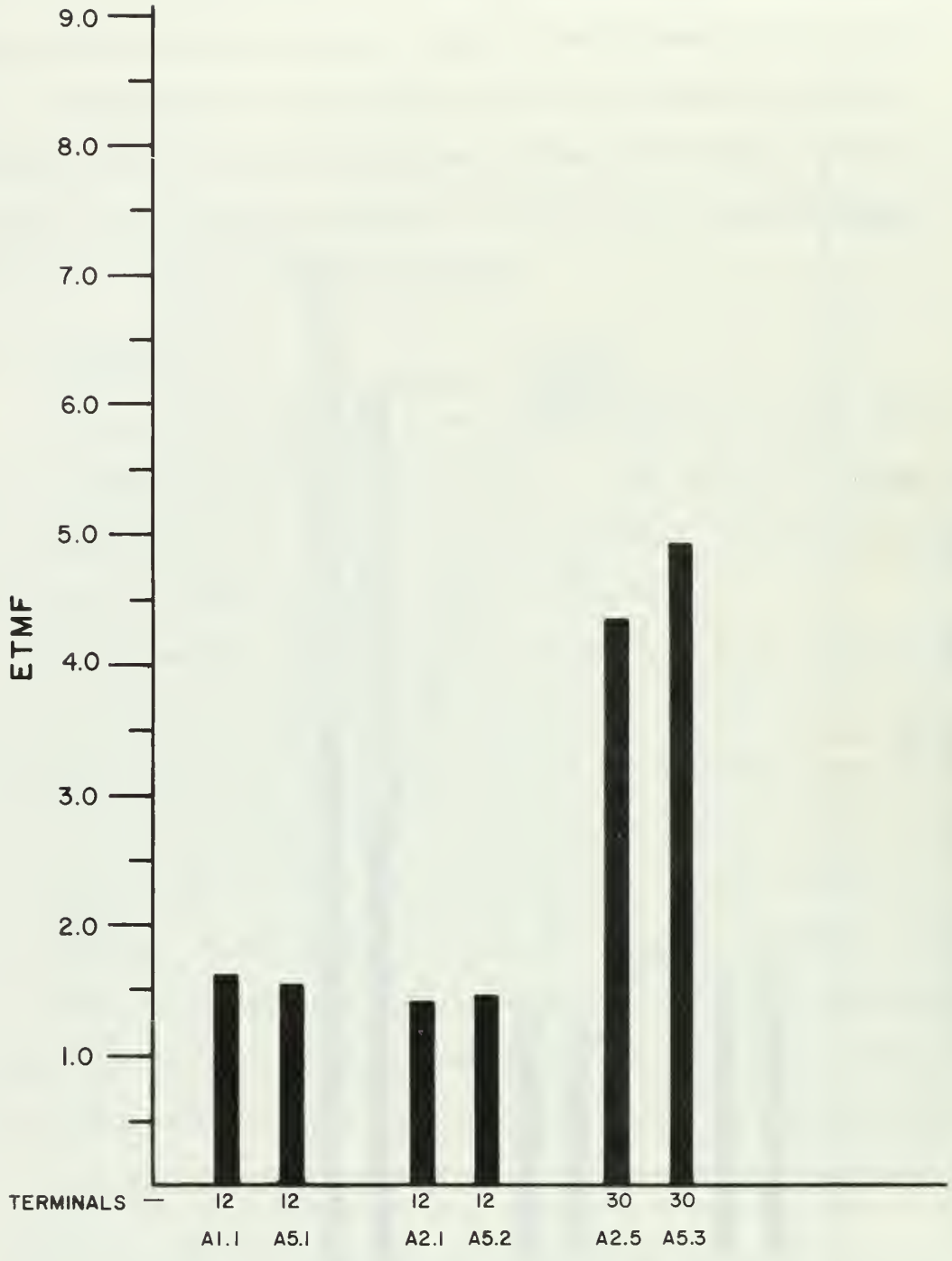


FIGURE 7

COMPARISON OF ETMFs FOR SERIES A5
AND
CORRESPONDING RUNS

The results of Series B1, tabulated in Table XI, Appendix D, were compared with the corresponding runs in Experiment A. The comparison of average response times is presented graphically in Figure 8. It was found that only between runs A4.1 and B1.4 was there a significant reduction in average response time. Furthermore, although the ETMFs for the runs of Series B1 were, in every case, lower than those of their corresponding runs in Experiment A the differences were slight, as depicted in Figure 9.

In Series B2 the effect of adding four additional I/O disks to the system was investigated. This alteration was particularly interesting since, at the time the runs were made, this modification actually was being planned. The DISK function in the model was changed to distribute the I/O requests uniformly over six disks, and four runs were made. These runs, tabulated in Table XII, Appendix D, were identical to their corresponding runs in Experiment A with the exception of the increased number of I/O disks.

The results of these runs were encouraging. As can be seen in Figure 10, average response times decreased significantly for runs B2.2, B2.3, and B2.4. Although the response time for run B2.1 increased slightly, the increase was not significant. Furthermore, as the bar graph in Figure 11 shows, the ETMF was decreased in every case. It also was observed that the percentage of execution increased noticeably in runs B2.3 and B2.4

C. EXPERIMENT C

Experiment C was designed to test the effect on response time of a new scheduling algorithm. The algorithm used was a modified version of one which is incorporated in newer versions of CP/CMS than the one

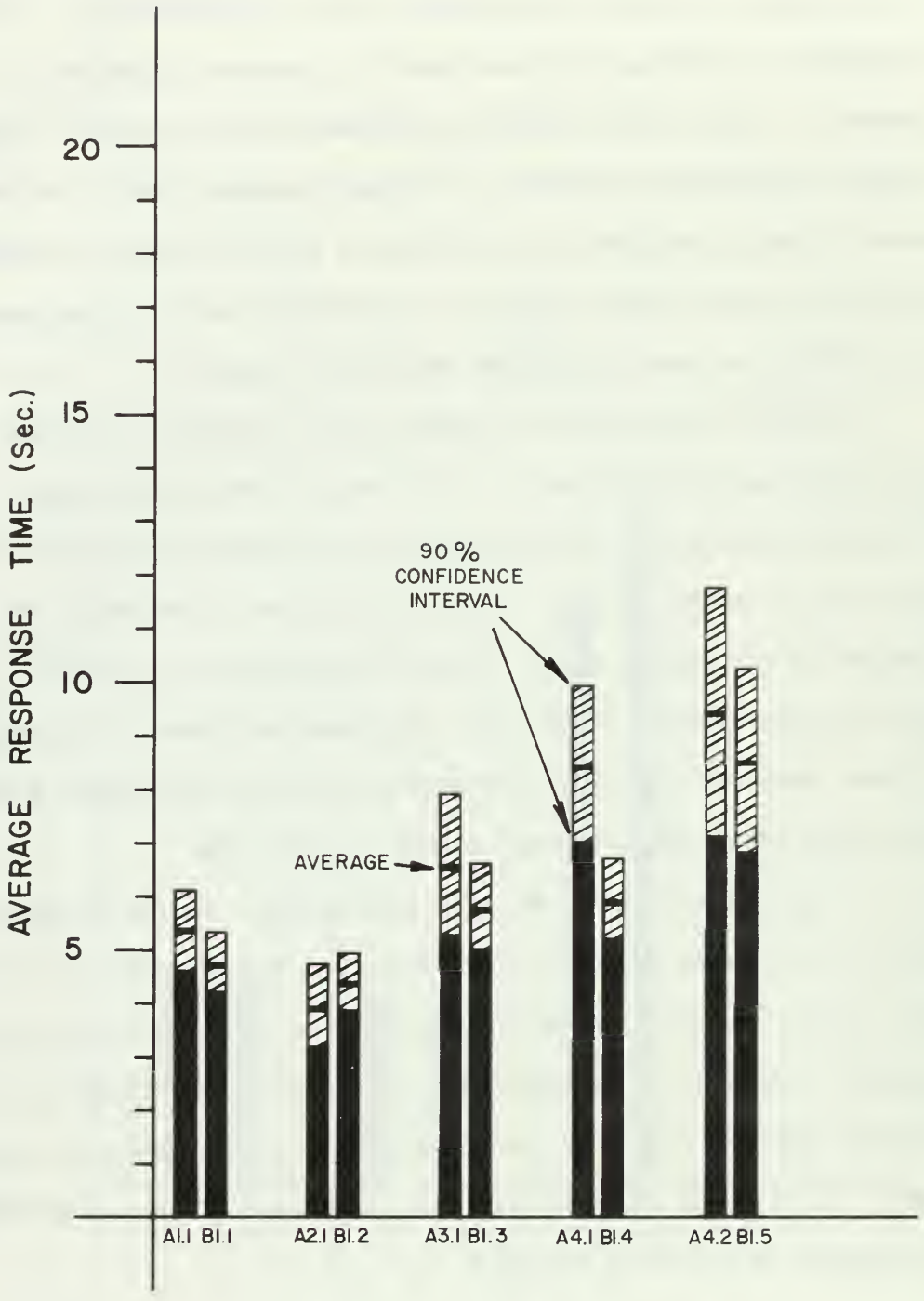


FIGURE 8
COMPARISON OF AVERAGE RESPONSE TIMES
FOR SERIES B1 AND
CORRESPONDING RUNS

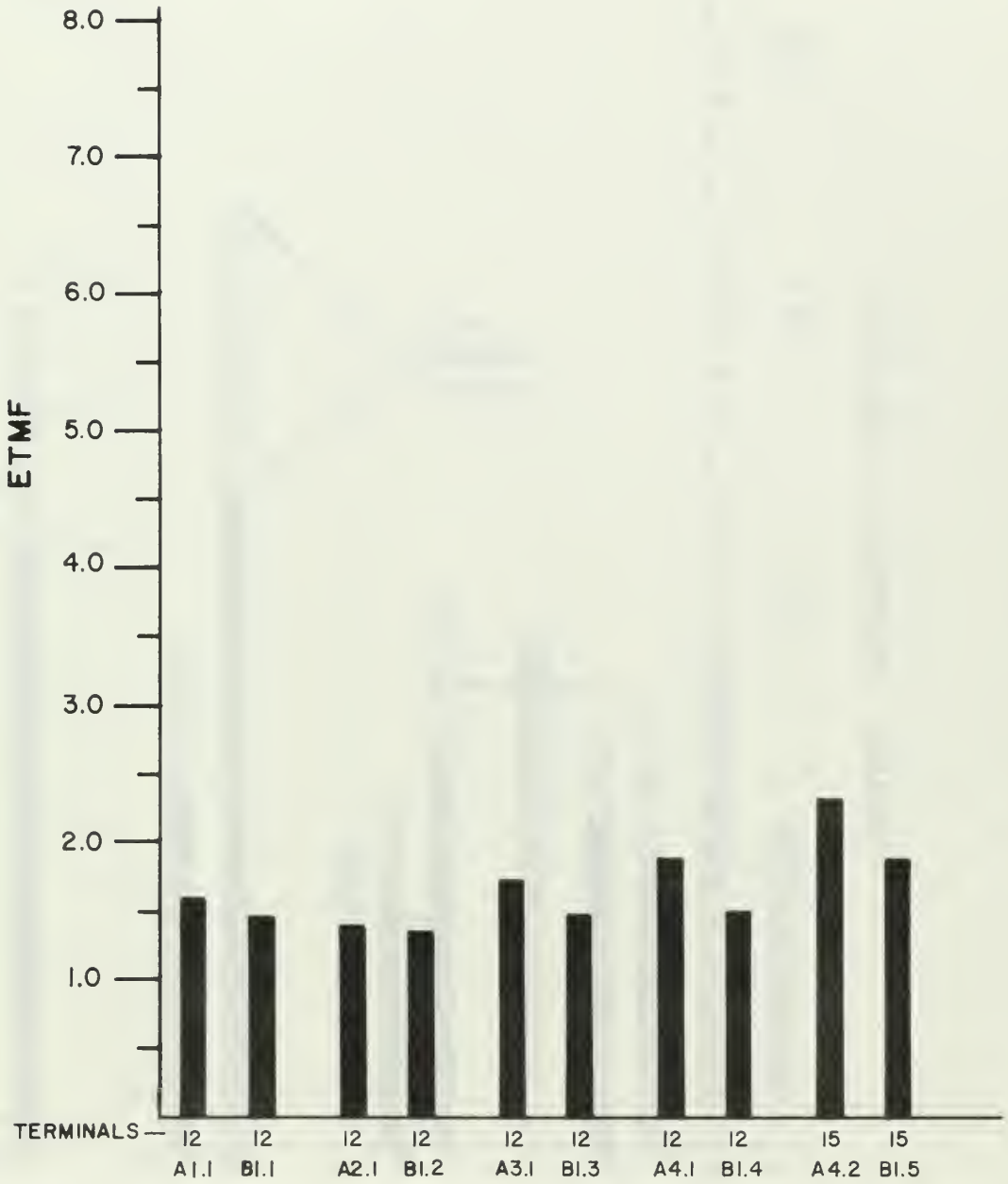


FIGURE 9

COMPARISON OF ETMFs FOR SERIES B1
AND
CORRESPONDING RUNS

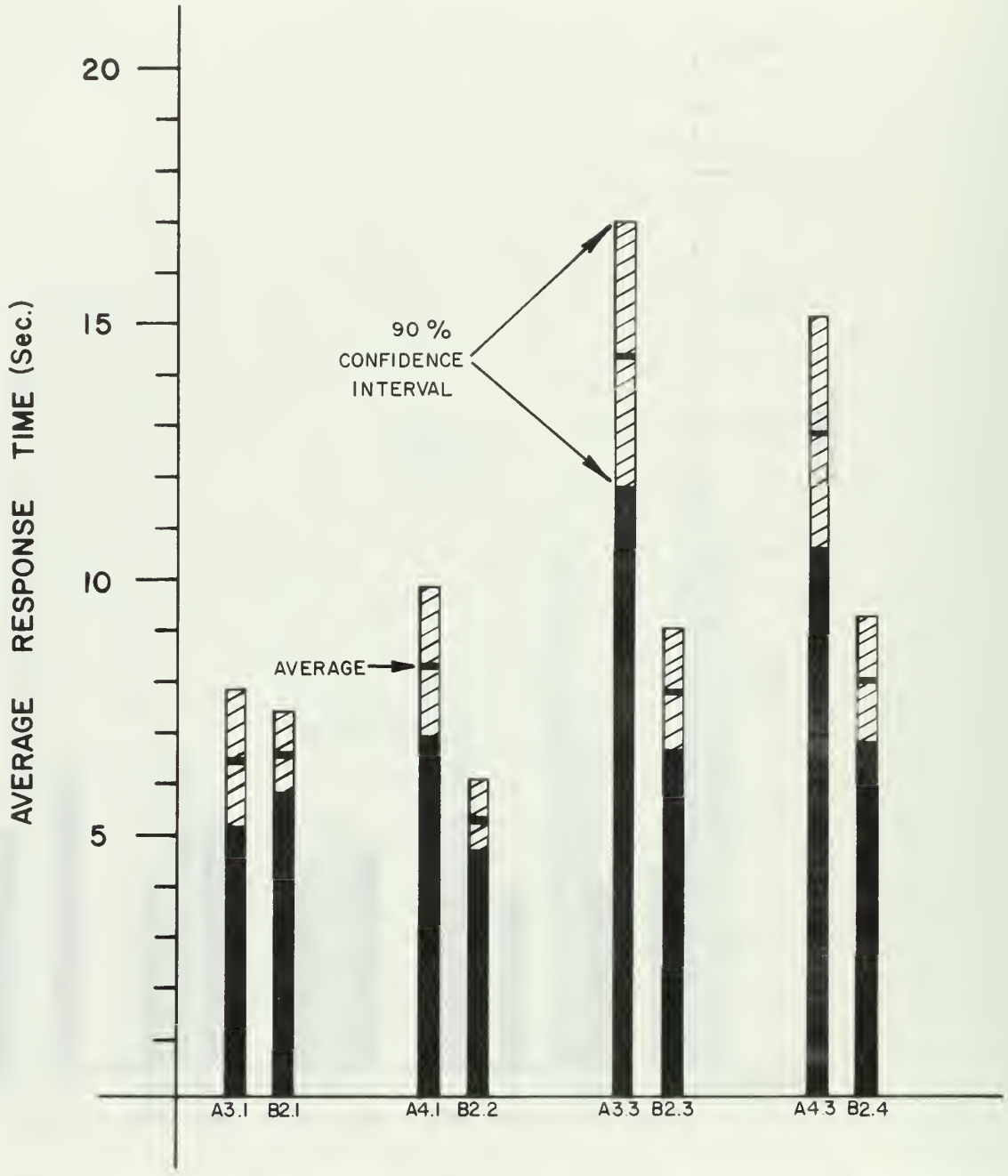


FIGURE 10
COMPARISON OF AVERAGE RESPONSE TIMES
FOR SERIES B2 AND
CORRESPONDING RUNS

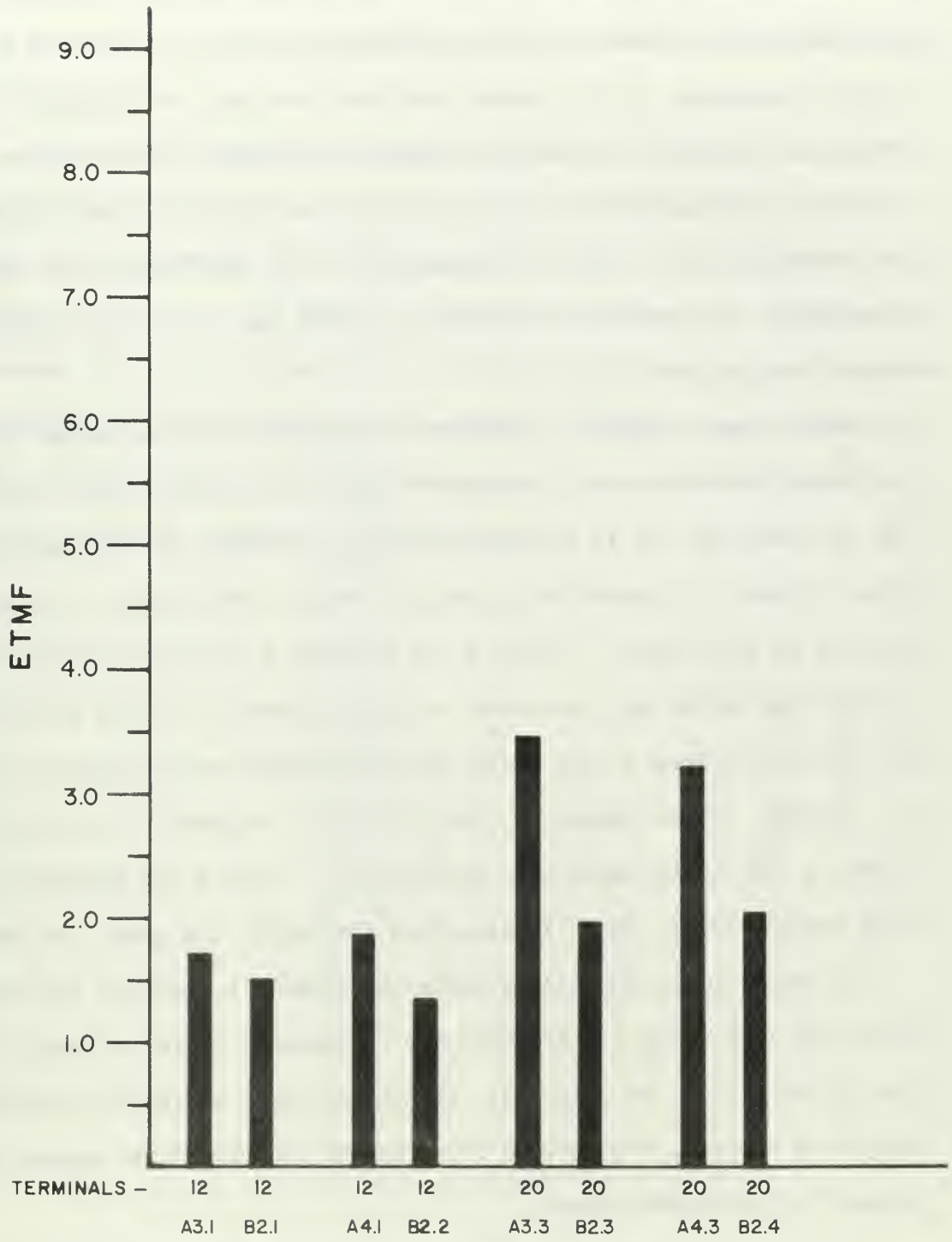


FIGURE 11

COMPARISON OF ETMFs FOR SERIES B2
AND
CORRESPONDING RUNS

modeled. The algorithm is designed to reduce overhead and paging load by limiting the number of jobs which simultaneously can compete for system resources. At any given time there are only two groups of jobs which can compete for resources, Group 1 and Group 2. The maximum number of jobs allowed in Group 1 at any time is N_1, and the maximum for Group 2 is N_2. Thus, a maximum of $N_1 + N_2$ jobs can vie for system resources. Any remaining jobs are in queues waiting to join either Group 1 or Group 2.

When a user completes a terminal interaction and the number of jobs in Group 1 is less than N_1 the user's job is assigned a time slice of 400 milliseconds and is placed in Group 1. However, if the number of jobs in Group 1 is equal to N_1 the job enters a FIFO queue of jobs waiting to join Group 1. When a job in Group 1 finishes its 400 millisecond time slice and the number of jobs in Group 2 is less than N_2 the job is assigned a time slice of five seconds and is placed in Group 2. However, if the number of jobs in Group 2 is equal to N_2 the job enters a FIFO queue waiting to join Group 2. When a job completes its five second time slice it is placed at the end of the queue for Group 2.

If possible the scheduling algorithm selects a runnable job for execution from Group 1. If there are no runnable users in Group 1 a job in Group 2 may be selected. If neither group contains runnable jobs, no job is started, even though there may be runnable jobs waiting in the queues to join either group.

After the appropriate changes were made in the model to reflect the new scheduling algorithm, two series of runs were made. The values assigned to N_1 and N_2 were varied among the runs to measure their effect, if any, on system performance. In both series the I/O rate and paging

interrupt rate were both set to high. The run conditions and results for Series C1 and C2 are summarized in Tables XIII and XIV respectively, in Appendix D.

The average response times of Series C1 and C2 runs are depicted graphically in Figure 12, along with the average response times of previous experiments with the same conditions. The analysis of results revealed that at a .10 level of significance there were no differences between the runs of Experiment C and their corresponding previous runs. Furthermore, it was found that the average response times from those runs in Experiment C which differed only in the values assigned to N_1 and N_2 were not significantly different.

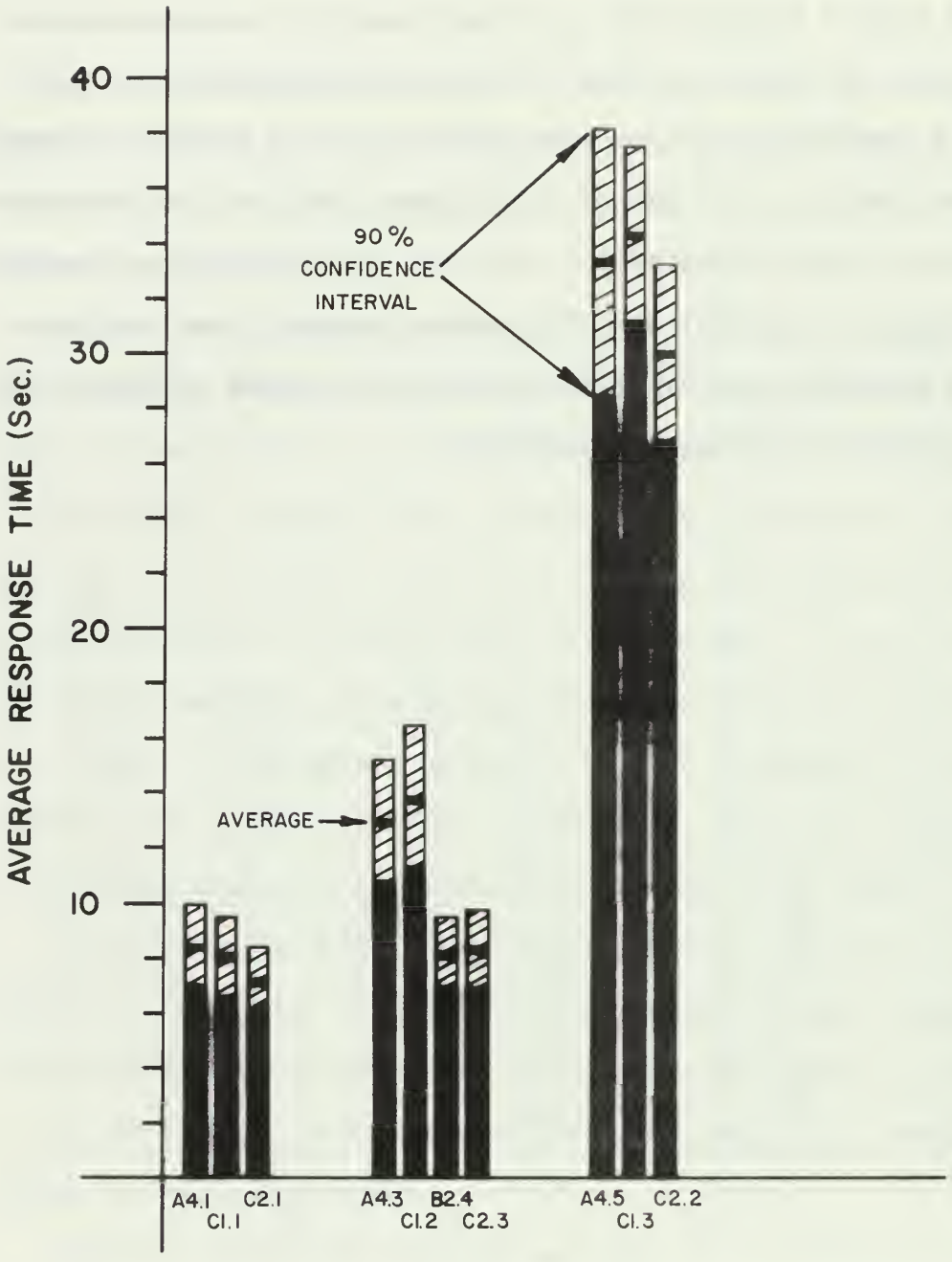


FIGURE 12
COMPARISON OF AVERAGE RESPONSE TIMES
FOR SERIES C1-C2
AND CORRESPONDING RUNS

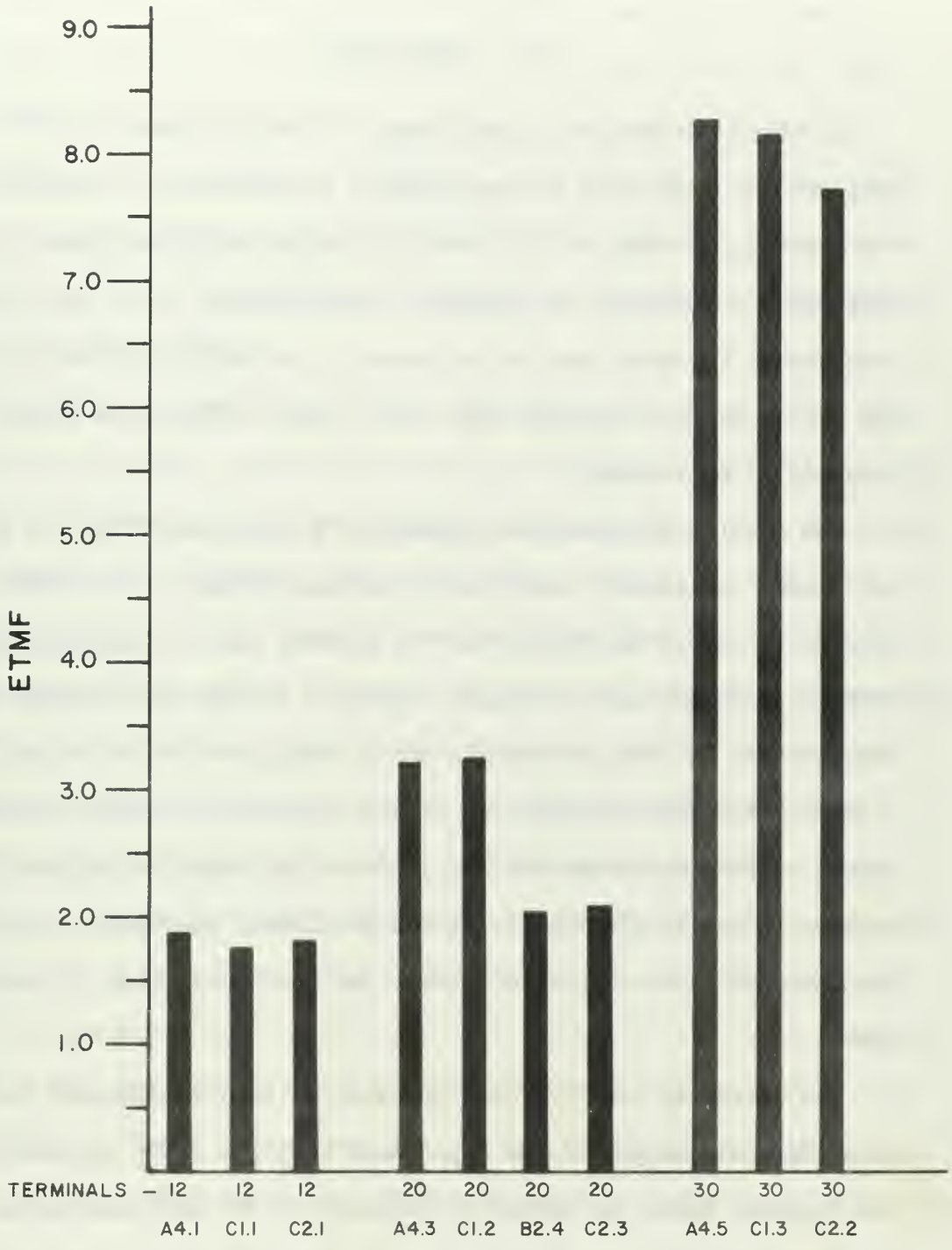


FIGURE 13

COMPARISON OF ETMFs FOR SERIES C1 - C2
AND
CORRESPONDING RUNS

VI. CONCLUSIONS

As mentioned earlier in this paper, it was not possible to validate the model used in the analysis of CP/CMS and, therefore, any conclusions based on the results of experiments performed with that model are subject to question. Nevertheless, it is felt that even though the model was not validated it is definitely similar to the system in most respects and, thus, should reflect the system's behavior to some extent.

The results obtained from Experiment A indicated that, for all job mixes considered, there was a substantial jump in the ETMF between 15 and 20 terminals, and for greater than 20 terminals both ETMF and response time increased rapidly. It also was observed that the increase in ETMFs between 12 and 15 terminals was relatively small in most cases and, furthermore, it was noted that in two of the four series of runs there was not a significant increase in response time between 12 and 15 terminals. It was concluded, therefore, that 15 terminals could be supported without serious degradation of response time.

The extremely high I/O disk utilization figures observed for 20 and more terminals supported the hypothesis that the sharp increase in ETMFs and response times was caused principally by the increased amount of jobs which were required to wait in the queue for I/O disks. This hypothesis was supported when it was calculated that in runs A3.3 and A4.3 each job spent an average of 8.6 and 7.3 seconds, respectively, in the queues for I/O disks.

Although the more efficient use of DASDs in Series B1 caused a significant reduction in response time in only one of the five runs, the ETMFs for the three runs with high I/O rates were reduced by a minimum of 13%. It appears, therefore, that although this experiment did not show a general significant reduction in response times, there is still a strong possibility that system performance could be enhanced by the proposed changes.

More definite results were obtained when Series B2 runs were made with six I/O disks. Three of the four runs showed statistically significant improvement in response times and all four of the runs exhibited a marked decline in ETMFs. The reduction in average I/O disk utilization observed in the 20 terminal runs suggested that the additional I/O disks appreciably reduced high I/O disk queueing times which were observed in runs A3.3 and A4.3. These results indicate that the system could adequately support 20 terminals with the additional four I/O disks.

The scheduling algorithm tested in Experiment C appeared to have little or no effect on either response time or ETMF when N_1 and N_2, the Group 1 and Group 2 sizes, were both equal to six. However, when the Group 2 size was changed to three for run C2.1 and to nine for run C2.2 a noticeable decrease occurred in both response times and ETMFs. Even though the decrease in response time was not statistically significant, the fact that there was an appreciable decrease in these runs, where none was observed in runs C1.1 and C1.3, suggested the possibility that the efficiency of the scheduling algorithm is extremely sensitive to the values selected for N_1 and N_2.

In summary, the results of this study indicated that the performance of the Naval Postgraduate School time-sharing computer system is limited by its small disk I/O capability, and an increased number of disk drives are required for the system to support a large increase in the number of attached terminals.

APPENDIX A

DATA FOR
CHI-SQUARE TEST
FOR
GOODNESS OF FIT

FUNCTION: NRPAG N = 380

INTERVAL	PROBABILITY (P_j)	EXPECTED (E_j)	OBSERVED (O_j)
0.5- 1.5	.20	76	77
1.5- 2.5	.30	114	116
2.5- 3.5	.06	22.8	28
3.5- 4.5	.05	19	19
4.5- 5.5	.04	15.2	12
5.5- 6.5	.04	15.2	16
6.5- 7.5	.04	15.2	19
7.5- 8.5	.03	11.4	12
8.5- 9.5	.02	7.6	10
9.5-10.5	.02	7.6	8
10.5-16.5	.06	22.8	18
16.5-25.5	.04	15.2	10
25.5-57.5	.10	38	35

FUNCTION: REACT N = 374

INTERVAL	PROBABILITY (P_j)	EXPECTED (E_j)	OBSERVED (O_j)
0- 2000	.046	17.20	23
2000- 4000	.084	31.42	45
4000- 6000	.105	39.27	38
6000- 8000	.126	47.12	48
8000- 10000	.096	35.90	25
10000- 12000	.082	30.67	25
12000- 14000	.049	18.33	15
14000- 16000	.041	15.33	18
16000- 18000	.036	13.46	16
18000- 20000	.031	11.59	18
20000- 22000	.025	9.35	13
22000- 24000	.023	8.60	11
24000- 26000	.020	7.48	4
26000- 28000	.018	6.73	4
28000- 30000	.017	6.36	5
30000- 35000	.038	14.21	15
35000- 60000	.030	11.22	9
60000-100000	.033	12.34	14
100000-200000	.050	18.70	17
200000-300000	.100	18.70	11

FUNCTION: EXPON N = 412

INTERVAL	PROBABILITY (P_j)	EXPECTED (E_j)	OBSERVED (O_j)
0	.005	2.06	0
0- 100	.043	17.72	21
100- 200	.047	19.36	15
200- 300	.044	18.13	25
300- 400	.042	17.30	15
400- 500	.040	16.48	18
500- 600	.038	15.66	20
600- 700	.037	15.24	15
700- 800	.033	13.60	16
800- 900	.033	13.60	19
900- 1000	.032	13.18	14
1000- 2000	.236	97.23	101
2000- 3000	.148	60.98	60
3000- 4000	.086	35.43	29
4000- 5000	.054	22.25	19
5000- 6000	.032	13.18	11
6000- 7000	.020	8.24	8
7000- 8000	.011	4.53	3
8000- 9000	.008	3.30	1
9000-10000	.004	1.65	2
10000-16000	.007	2.88	0

FUNCTION: SEEK		N = 8511	
INTERVAL	PROBABILITY (P_j)	EXPECTED (E_j)	OBSERVED (O_j)
25- 65	.19	988.19	1016
65- 75	.45	2340.45	2296
75-108	.28	1456.28	1495
108-135	.08	416.08	394

The formula applied to arrive at the computed value in all cases was

$$\chi^2 = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_2 - E_2)^2}{E_2} + \cdots + \frac{(O_N - E_N)^2}{E_N},$$

or

$$\chi^2 = \sum_{j=1}^{N} \frac{(O_j - E_j)^2}{E_j}$$

where

$$E_j = NP_j.$$

Results of these computations are tabulated in Table IV, Section III.

APPENDIX B

CALCULATION OF CONFIDENCE INTERVALS

The following calculation uses data obtained from the sample output, which corresponds to run A3.4.

The sample average is given by

$$\bar{X}_t = 19.195 \text{ seconds},$$

and the estimate for the variance using 29 lags, is seen to be

$$\hat{V} = 3.473.$$

The equivalent degrees of freedom are obtained from

$$EDF = 1.5(200/29)$$

$$= 10.3.$$

From a table of the Student t distribution, we have

$$t_{.95} = 1.81,$$

so that the confidence interval given by the probability statement

$$P(\bar{X}_t - t_{.95} \hat{V}^{1/2} \leq \mu \leq \bar{X}_t + t_{.95} \hat{V}^{1/2}) = .90,$$

is seen to be

$$19.195 \pm 3.374 \text{ or } (15.821, 22.569).$$

Using runs A4.1 and B2.2 as an example of the test for difference in means, the following data are observed:

$$\bar{X}_1 = 8.493 \text{ seconds}$$

$$\bar{X}_2 = 5.487 \text{ seconds}$$

$$\hat{V}_1 = 0.52362 \qquad \hat{V}^{1/2} = 0.72360$$

$$\hat{V}_2 = 0.16996 \qquad \hat{V}_2^{1/2} = 0.41223$$

From a table of the Student t distribution we have

$$t_1 = 2.02$$

$$t_2 = 1.68$$

so that

$$t_1 \hat{V}_1^{1/2} = 1.462$$

$$t_2 \hat{V}_2^{1/2} = 0.688$$

and the statistic t' is calculated as

$$t' = (1.462 + 0.688)/(0.72360 + 0.41223)$$

$$= 2.150/1.136$$

$$= 1.893.$$

Further,

$$(V_1 + V_2)^{1/2} = (0.69358)^{1/2}$$

$$= 0.8328,$$

and

$$\bar{X}_1 - \bar{X}_2 = 8.493 - 5.487$$

$$= 3.006 \text{ seconds},$$

which is well outside the interval of acceptance given by

$$P(-1.893(0.8328) \leq (\bar{X}_1 - \bar{X}_2) \leq +1.893(0.8328)) = .90.$$

Thus, the hypothesis is rejected at the .10 level of significance that the means are equal.

APPENDIX C

CALCULATION OF THE ELAPSED TIME MULTIPLICATION FACTOR

The Elapsed Time Multiplication Factor, or ETMF, is defined by

$$ETMF = R/T,$$

where R is the average response time and T is the average time to complete a task when it is run alone. When computer programs are run as "stand alone" tasks T is given by

$$T = E + I,$$

where E is the average CPU time required and I is the average I/O time required. If IR is the average rate of I/O requests per second of execution and S is the average I/O service time, then

$$I = E(IR)S.$$

Substituting in the above equations yields

$$T = E(1 + (IR)S),$$

and

$$ETMF = R/(E(1 + (IR)S)).$$

The values of R and E were obtained from each experimental run. IR, the average I/O request rate for a given run, was approximated by the mean of the appropriate I/O request rate distribution. Similarly, S, the average I/O service time, was approximated by the sum of the means of the seek time distribution (given by the function SEEK) and the rotational and transmission delay distribution (given by the variable DIOT). Preliminary runs indicated that these approximations were very good.

As an example, for run B1.2 the following values were obtained from Tables V and IX:

$$R = 4.389 \text{ seconds}$$

$$E = 872 \text{ msec.} = 0.872 \text{ seconds}$$

$$IR = 24.10/\text{second.}$$

Furthermore,

$$S = \text{mean of SEEK} + \text{mean of DIOT}$$

$$= 0.075 + 0.038$$

$$= 0.113 \text{ seconds.}$$

Therefore,

$$ETMF = \frac{4.389}{0.872(1 + (24.10)(0.113))}$$

$$= 1.35.$$

APPENDIX D

TABULATED EXPERIMENTAL RESULTS

TABLE VI

RESULTS OF SERIES A1 RUNS (LOW I/O, LOW PAGING)

	RUN NUMBER				
	A1.1	A1.2	A1.3	A1.4	A1.5
NUMBER OF TERMINALS	12	15	20	25	30
TIME TO COMPLETE 1000 TASKS (MIN.)	50.8	41.3	33.3	29.0	28.5
% EXECUTION	29.5	33.7	42.4	47.0	49.6
% OVERHEAD	6.8	7.6	9.9	10.9	12.5
% IDLE	63.7	58.7	47.7	42.1	37.9
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	898	834	846	818	848
AVERAGE RESPONSE TIME (SEC.)	5.380	5.170	7.489	8.623	14.208

TABLE VI (continued)

	RUN NUMBER				
	A1.1	A1.2	A1.3	A1.4	A1.5
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(4.612, 6.146)	(4.374, 5.960)	(5.373, 9.603)	(7.001, 10.441)	(7.199, 21.215)
EIMF	1.61	1.67	2.37	2.83	4.49
PAGES READ/SECOND	0.9	0.9	1.9	2.3	3.8
PAGES READ/SECOND OF EXECUTION	3.2	2.6	4.5	4.8	7.6
PAGING DRUM UTILIZATION	.023	.020	.032	.033	.040
PAGING DISK UTILIZATION	-	-	.136	.221	.539
I/O DISK UTILIZATION	.457	.508	.680	.767	.847

TABLE VII

RESULTS OF SERIES A2 RUNS
(LOW I/O, HIGH PAGING)

	RUN NUMBER				
	A2.1	A2.2	A2.3	A2.4	A2.5
NUMBER OF TERMINALS	12	15	20	25	30
TIME TO COMPLETE 1000 TASKS (MIN.)	57.3	44.3	35.0	32.8	27.3
% EXECUTION	22.2	32.8	41.5	44.7	48.9
% OVERHEAD	6.1	9.0	12.1	13.8	15.1
% IDLE	71.7	58.2	46.4	41.5	36.0
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	763	871	846	878	785
AVERAGE RESPONSE TIME (SEC.)	3.969	5.324	7.609	12.131	12.673
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(3.192, 4.746)	(4.548, 6.100)	(6.429, 8.789)	(8.189, 16.073)	(8.507, 16,839)

TABLE VII (continued)

	RUN NUMBER				
	A2.1	A2.2	A2.3	A2.4	A2.5
EIMF	1.40	1.65	2.35	3.71	4.33
PAGES READ/SECOND	0.8	1.4	3.0	4.6	5.5
PAGES READ/SECOND OF EXECUTION	3.6	4.3	4.5	10.3	11.5
PAGING DRUM UTILIZATION	.019	.032	.047	.066	.067
PAGING DISK UTILIZATION	-	-	.239	.440	.660
I/O DISK UTILIZATION	.332	.485	.684	.772	.814

TABLE VIII

RESULTS OF SERIES A3 RUNS
(HIGH I/O, LOW PAGING)

	RUN NUMBER				
	A3.1	A3.2	A3.3	A3.4	A3.5
NUMBER OF TERMINALS	12	15	20	25	30
TIME TO COMPLETE 1000 TASKS (MIN.)	60.3	45.3	41.0	38.5	35.5
% EXECUTION	21.0	28.6	33.5	35.8	37.0
% OVERHEAD	6.2	8.5	10.4	11.0	12.5
% IDLE	72.8	62.9	56.1	53.2	50.5
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	759	777	824	827	788
AVERAGE RESPONSE TIME (SEC.)	6.604	9.053	14.516	19.195	31.062
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(5.293, 7.915)	(7.665, 10.441)	(11.919, 17.113)	(15.821, 22.569)	(25.177, 36.947)

TABLE VIII (continued)

	RUN NUMBER				
	A3.1	A3.2	A3.3	A3.4	A3.5
ETMF	1.71	2.28	3.44	4.54	7.70
PAGES READ/SECOND	0.7	0.9	1.6	2.1	3.9
PAGES READ/SECOND OF EXECUTION	3.3	3.0	4.8	5.8	10.6
PAGING DRUM UTILIZATION	.015	.020	.026	.028	.050
PAGING DISK UTILIZATION	-	-	.124	.248	.475
I/O DISK UTILIZATION	.487	.682	.843	.896	.950

TABLE IX

RESULTS OF SERIES A4 RUNS
(HIGH I/O, HIGH PAGING)

	RUN NUMBER				
	A4.1	A4.2	A4.3	A4.4	A4.5
NUMBER OF TERMINALS	12	15	20	25	30
TIME TO COMPLETE 1000 TASKS (MIN.)	59.5	47.3	39.5	36.8	37.3
% EXECUTION	24.8	28.4	33.4	35.1	35.3
% OVERHEAD	8.6	10.1	12.0	13.7	14.0
% IDLE	66.6	61.5	54.6	51.2	50.7
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	885	805	792	774	779
AVERAGE RESPONSE TIME (SEC.)	8.493	9.485	13.024	23.160	33.391
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(7.031, 9.955)	(7.165, 11.805)	(10.776, 15.272)	(20.336, 25.984)	(28.583, 38.199)

TABLE IX (continued)

	RUN NUMBER				
	A4.1	A4.2	A4.3	A4.4	A4.5
ETMF	1.88	2.31	3.22	5.86	8.28
PAGES READ/SECOND	1.0	1.4	2.2	4.0	4.8
PAGES READ/SECOND OF EXECUTION	4.1	4.8	6.6	11.4	13.6
PAGING DRUM UTILIZATION	.023	.031	.037	.049	.054
PAGING DISK UTILIZATION	-	-	.161	.517	.649
I/O DISK UTILIZATION	.590	.680	.847	.931	.951

TABLE X

RESULTS OF SERIES A5 RUNS
(CHECK OF STABILITY)

	RUN NUMBER		
	A5.1	A5.2	A5.3
I/O RATE	LOW	LOW	LOW
PAGING INTERRUPT RATE	LOW	HIGH	HIGH
NUMBER OF TERMINALS	12	12	30
TIME TO COMPLETE 1000 TASKS (MIN.)	56.0	52.3	27.8
% EXECUTION	25.5	25.1	48.8
% OVERHEAD	5.7	7.0	15.0
% IDLE	68.8	67.9	36.2
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	857	794	813
AVERAGE RESPONSE TIME (SEC.)	4.903	4.363	14.888

TABLE X (continued)

	RUN NUMBER		
	A5.1	A5.2	A5.3
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(4.180, 5.626)	(3.885, 4.839)	(10.360, 19.414)
ETMF	1.53	1.47	4.92
PAGES READ/SECOND	0.7	0.8	5.1
PAGES READ/SECOND OF EXECUTION	2.9	3.3	10.4
PAGING DRUM UTILIZATION	.017	.019	.062
PAGING DISK UTILIZATION	-	-	.695
I/O DISK UTILIZATION	.392	.392	.841
CORRESPONDING RUN	A1.1	A2.1	A2.5

TABLE XI

RESULTS OF SERIES B1 RUNS
(CMS PAGES ON DRUM & 3.75 I/O DISKS)

	RUN NUMBER				
	B1.1	B1.2	B1.3	B1.4	B1.5
NUMBER OF TERMINALS	12	12	12	12	15
I/O RATE	LOW	LOW	HIGH	HIGH	HIGH
PAGING INTERRUPT RATE	LOW	HIGH	LOW	HIGH	HIGH
TIME TO COMPLETE 1000 TASKS (MIN.)	55.8	52.3	53.5	58.3	45.8
% EXECUTION	26.2	27.8	24.0	22.4	32.8
% OVERHEAD	6.0	7.5	6.9	8.0	11.4
% IDLE	67.8	64.7	69.1	69.6	56.8
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	870	872	770	782	900
AVERAGE RESPONSE TIME (SEC.)	4.751	4.389	5.824	5.931	8.577

TABLE XI (continued)

	RUN NUMBER				
	B1.1	B1.2	B1.3	B1.4	B1.5
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(4.172, 5.330)	(3.864, 4.914)	(5.018, 6.630)	(5.145, 6.717)	(6.836, 10.318)
ETMF	1.47	1.35	1.48	1.49	1.87
PAGES READ/SECOND	0.6	1.2	0.7	1.1	1.3
PAGES READ/SECOND OF EXECUTION	2.4	4.1	2.7	4.8	4.0
PAGING DRUM UTILIZATION	.015	.027	.015	.025	.030
I/O DISK UTILIZATION	.232	.218	.300	.294	.463
CORRESPONDING RUN	A1.1	A2.1	A3.1	A4.1	A4.2

TABLE XII

RESULTS OF SERIES B2 RUNS
(SIX I/O DISKS)

	RUN NUMBER			
	B2.1	B2.2	B2.3	B2.4
NUMBER OF TERMINALS	12	12	20	20
I/O RATE	HIGH	HIGH	HIGH	HIGH
PAGING INTERRUPT RATE	LOW	HIGH	LOW	HIGH
TIME TO COMPLETE 1000 TASKS	53.8	59.3	33.5	34.3
% EXECUTION	27.2	22.2	40.1	38.3
% OVERHEAD	8.1	7.7	12.2	18.3
% IDLE	64.7	70.1	47.7	43.4
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	877	789	806	787
AVERAGE RESPONSE TIME (SEC.)	6.748	5.487	7.986	8.203

TABLE XII (continued)

	RUN NUMBER			
	B2.1	B2.2	B2.3	B2.4
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(5.977, 7.519)	(4.799, 6.175)	(6.804, 9.168)	(6.991, 9.415)
EIMF	1.51	1.36	1.94	2.04
PAGES READ/SECOND	0.9	1.0	1.7	2.2
PAGES READ/SECOND OF EXECUTION	3.2	4.4	4.3	5.7
PAGING DRUM UTILIZATION	.020	.022	.029	.036
PAGING DISK UTILIZATION	-	-	.140	.190
I/O DISK UTILIZATION	.237	.180	.396	.376
CORRESPONDING RUN	A3.1	A4.1	A3.3	A4.3

TABLE XIII

RESULTS OF SERIES C1 RUNS
(NEW SCHEDULING ALGORITHM)

	RUN NUMBER		
	C1.1	C1.2	C1.3
NUMBER OF TERMINALS	12	20	30
GROUP 1 SIZE	6	6	6
GROUP 2 SIZE	6	6	6
TIME TO COMPLETE 1000 TASKS (MIN.)	66.1	42.8	40.0
% EXECUTION	22.9	32.6	34.4
% OVERHEAD	7.7	12.4	13.5
% IDLE	69.4	56.0	52.1
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	895	836	826
AVERAGE RESPONSE TIME (SEC.)	8.073	13.892	34.354

TABLE XIII (continued)

	RUN NUMBER		
	C1.1	C1.2	C1.3
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(6.649, 9.497)	(11.317, 16.467)	(31.215, 37.493)
EIMF	1.77	3.25	8.15
PAGES READ/SECOND	0.7	3.3	4.0
PAGES READ/SECOND OF EXECUTION	3.0	10.1	11.5
PAGING DRUM UTILIZATION	.016	.051	.040
PAGING DISK UTILIZATION	-	.277	.600
I/O DISK UTILIZATION	.537	.820	.909
CORRESPONDING RUN(S)	A4.1 C2.1	A4.3	A4.5 C2.2

TABLE XIV

RESULTS OF SERIES C2 RUNS
(NEW SCHEDULING ALGORITHM)

	RUN NUMBER		
	C2.1	C2.2	C2.3
NUMBER OF TERMINALS	12	30	20
GROUP 1 SIZE	6	6	6
GROUP 2 SIZE	3	9	6
TIME TO COMPLETE 1000 TASKS (MIN.)	56.0	36.3	35.8
% EXECUTION	23.5	35.0	36.6
% OVERHEAD	8.2	14.3	13.2
% IDLE	69.3	50.7	50.2
AVERAGE CPU TIME REQUIRED/TASK (MSEC.)	790	761	785
AVERAGE RESPONSE TIME (SEC.)	7.296	30.028	8.369

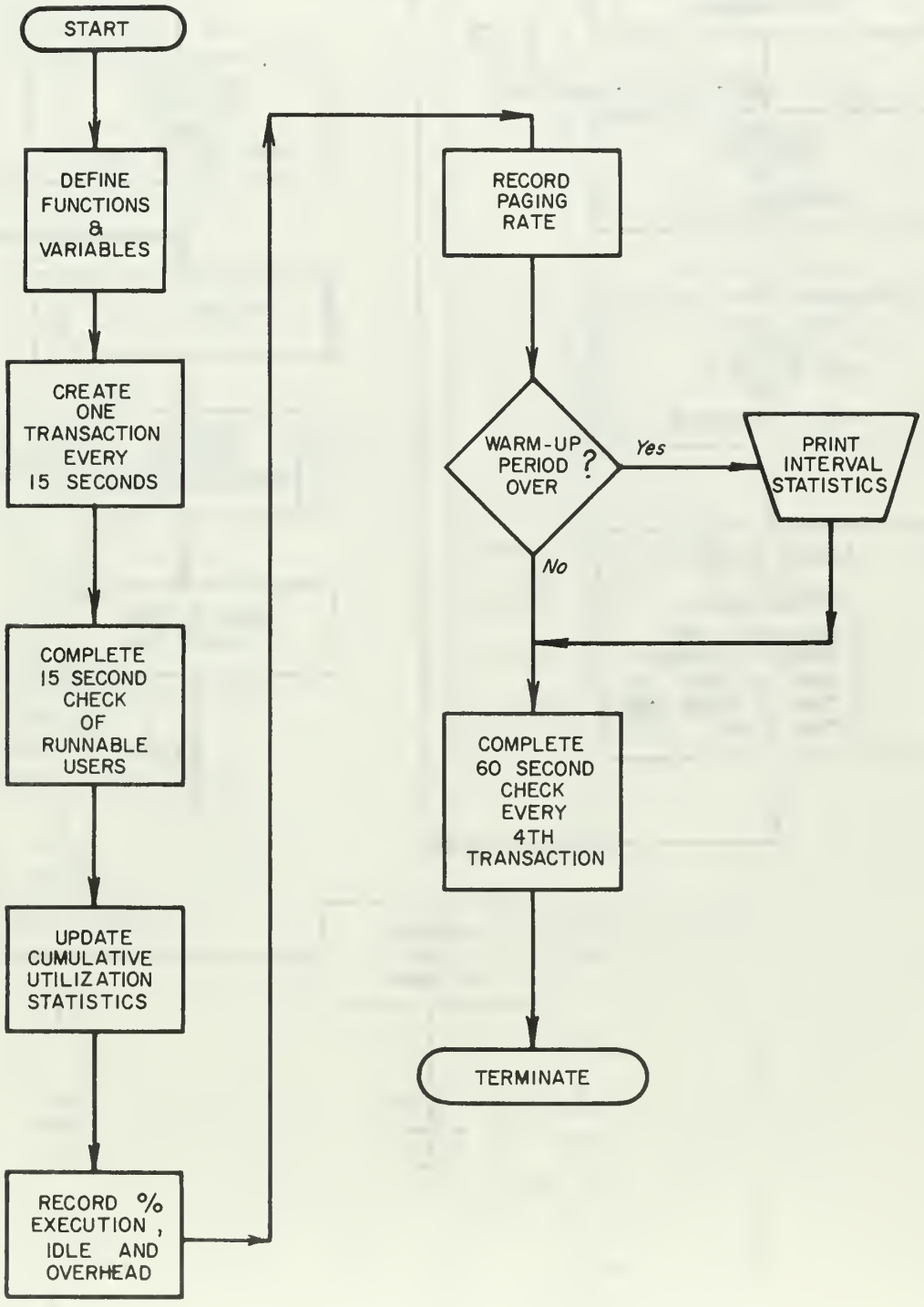
TABLE XIV (continued)

	RUN NUMBER		
	C2.1	C2.2	C2.3
90% CONFIDENCE INTERVAL FOR AVERAGE RESPONSE TIME	(6.200, 8.392)	(26.824, 33.231)	(6.983, 9.755)
ETMF	1.81	7.72	2.09
PAGES READ/SECOND	1.3	5.0	2.4
PAGES READ/SECOND OF EXECUTION	5.6	14.2	6.5
PAGING DRUM UTILIZATION	.030	.054	.039
PAGING DISK UTILIZATION	-	.703	.185
I/O DISK UTILIZATION	.545	.937	.356
CORRESPONDING RUN(S)	A4.1 C1.1	A4.5 C1.3	B2.4

APPENDIX E

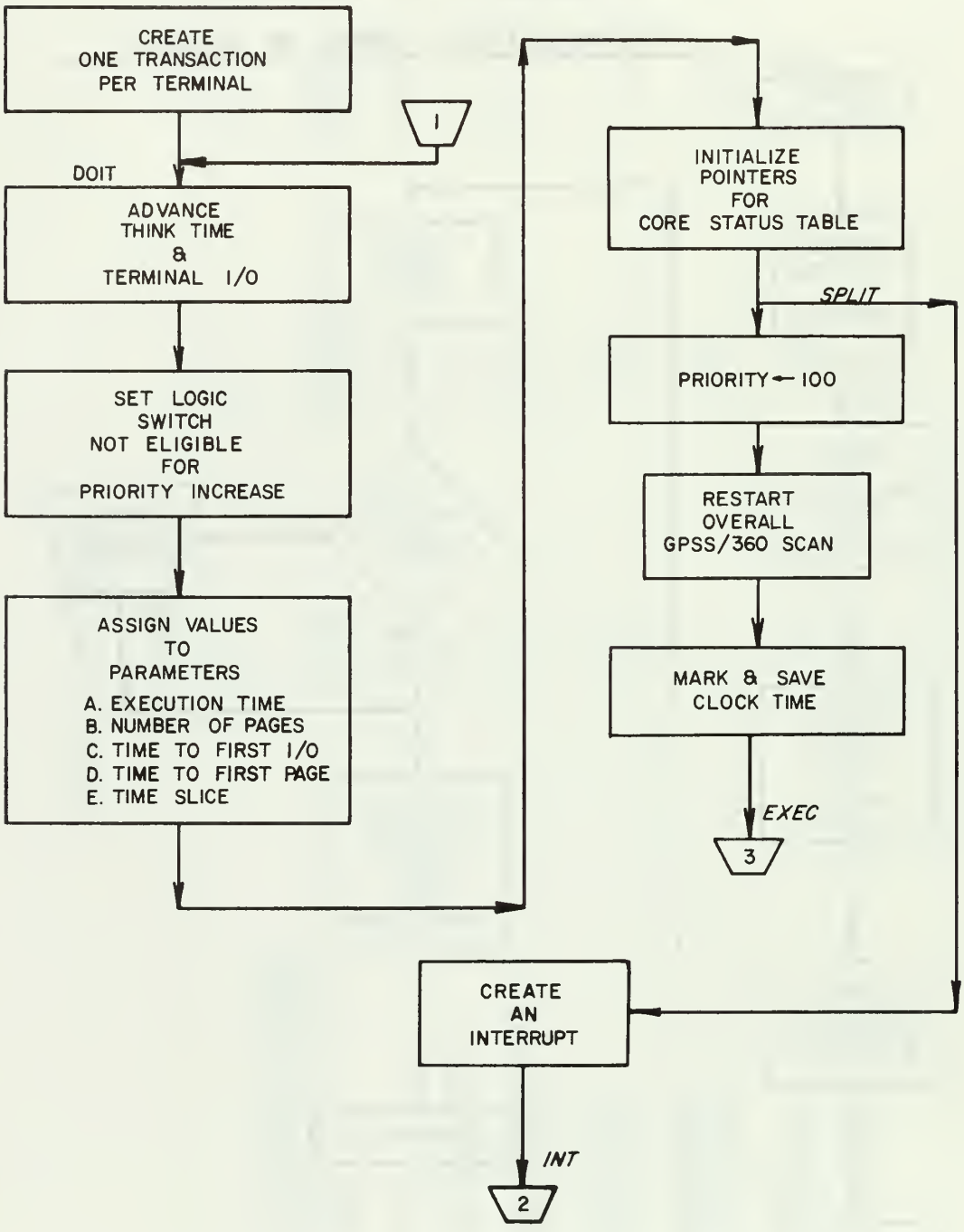
FLOWCHARTS OF THE MODEL

INITIALIZATION, TIMING & OUTPUT

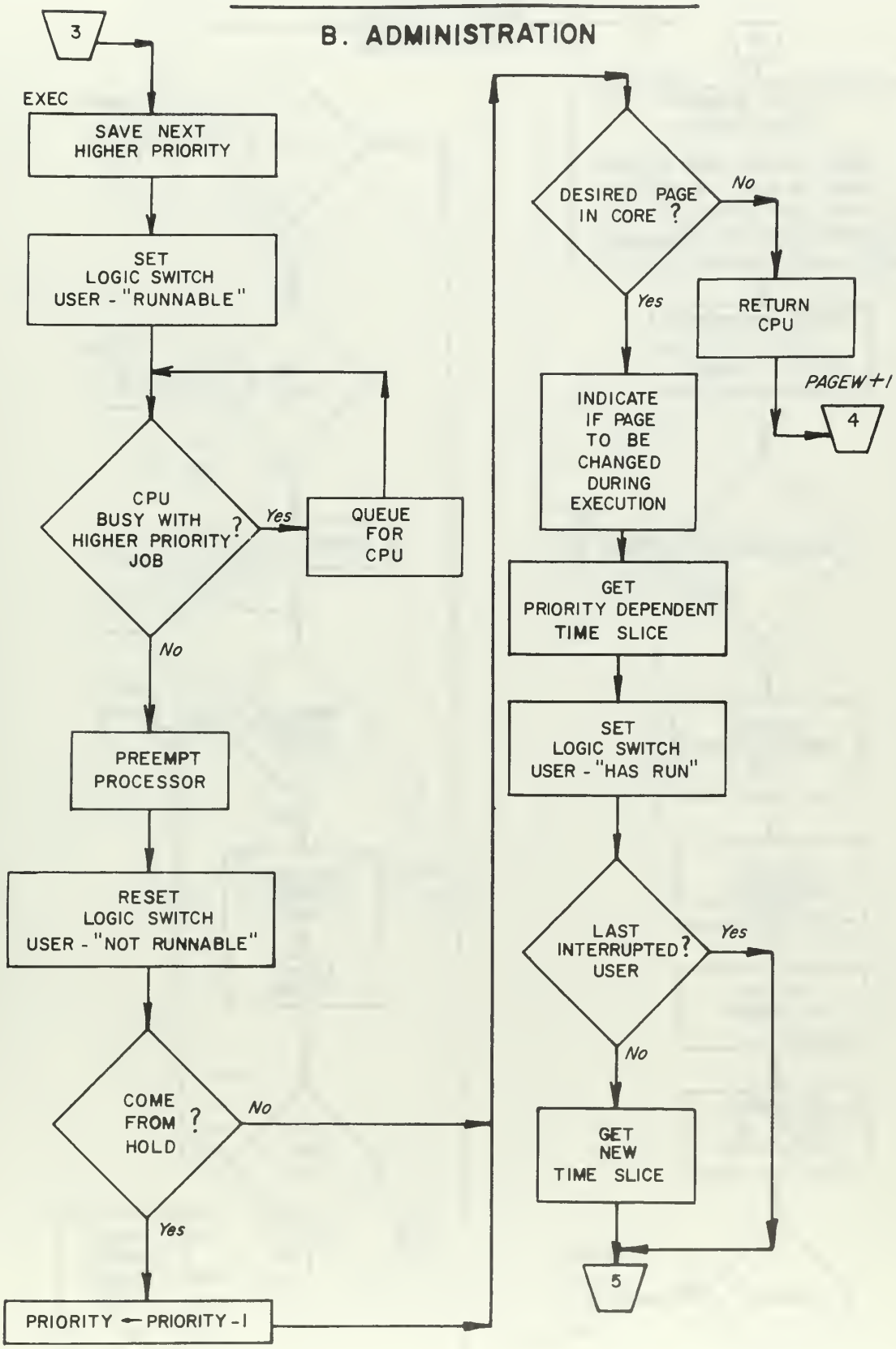


EXECUTION

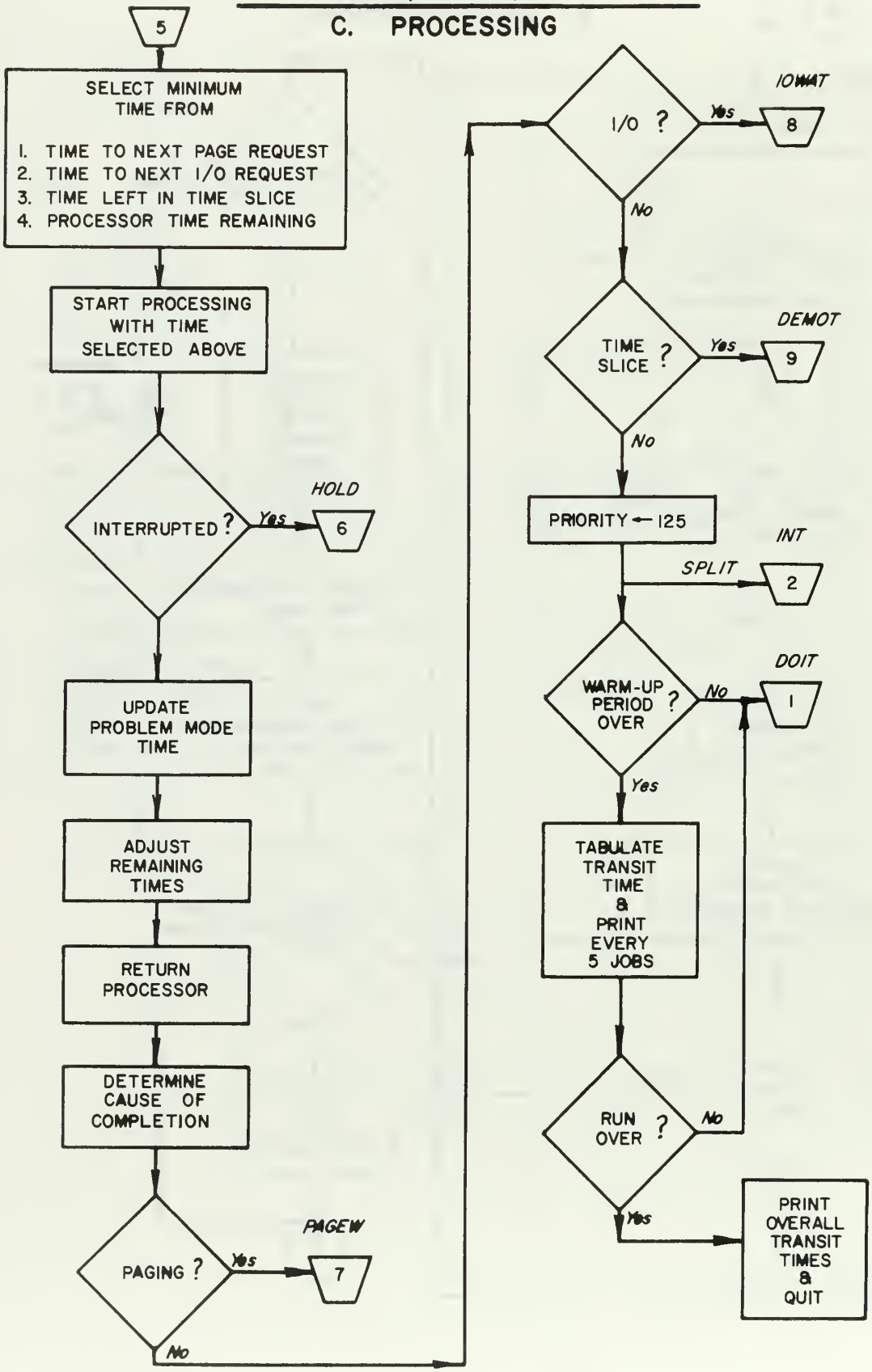
A. INITIALIZATION

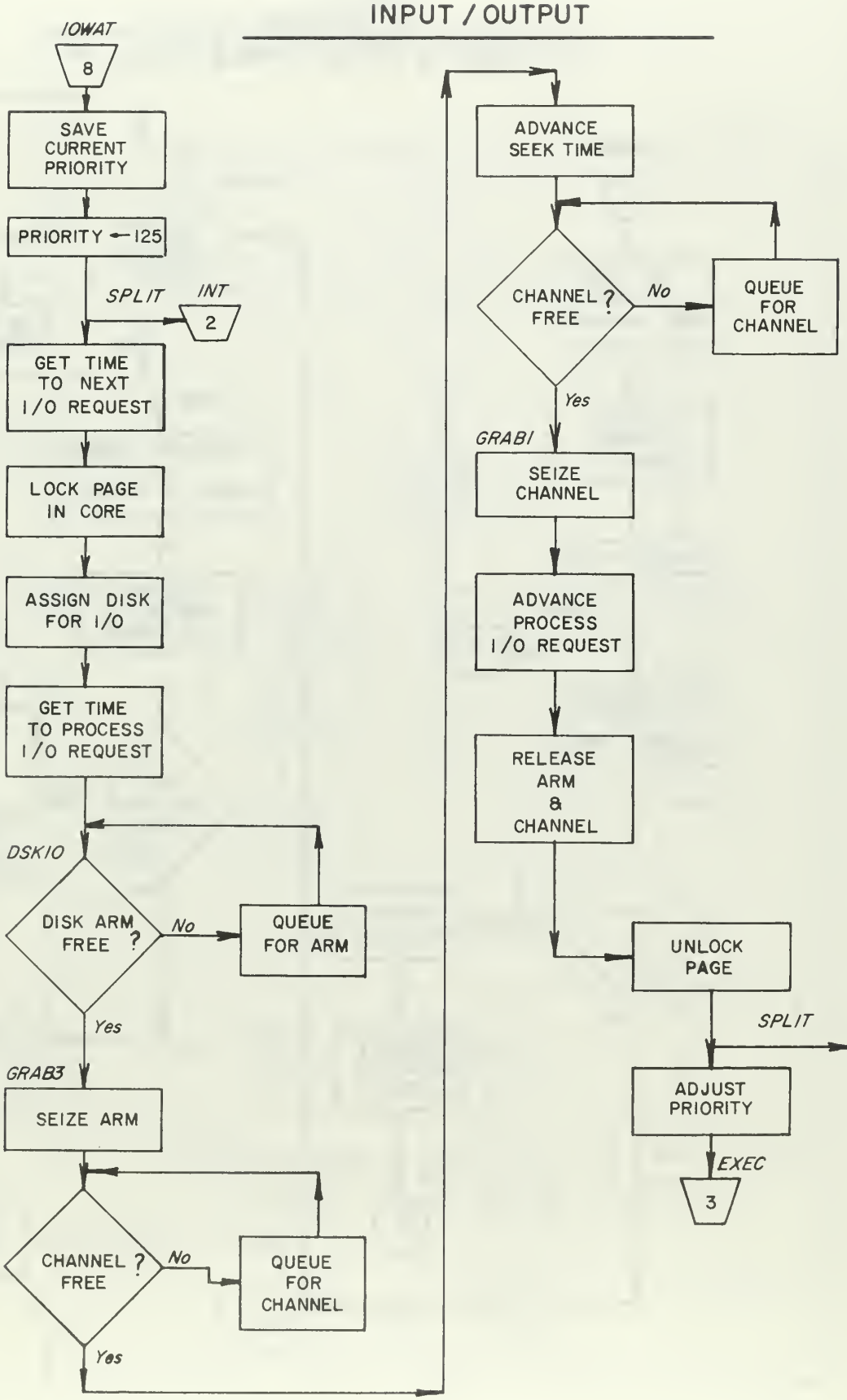


B. ADMINISTRATION



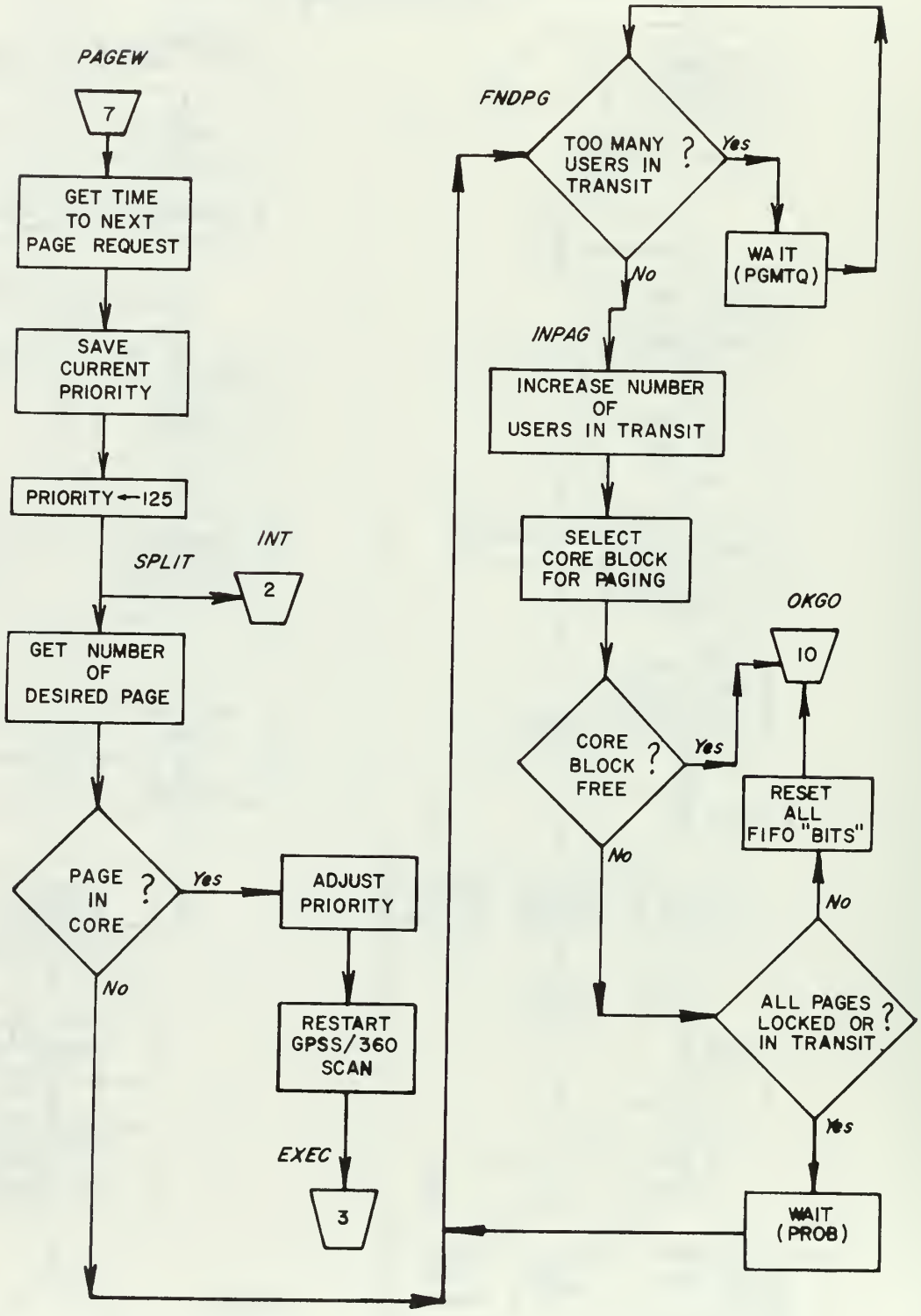
C. PROCESSING



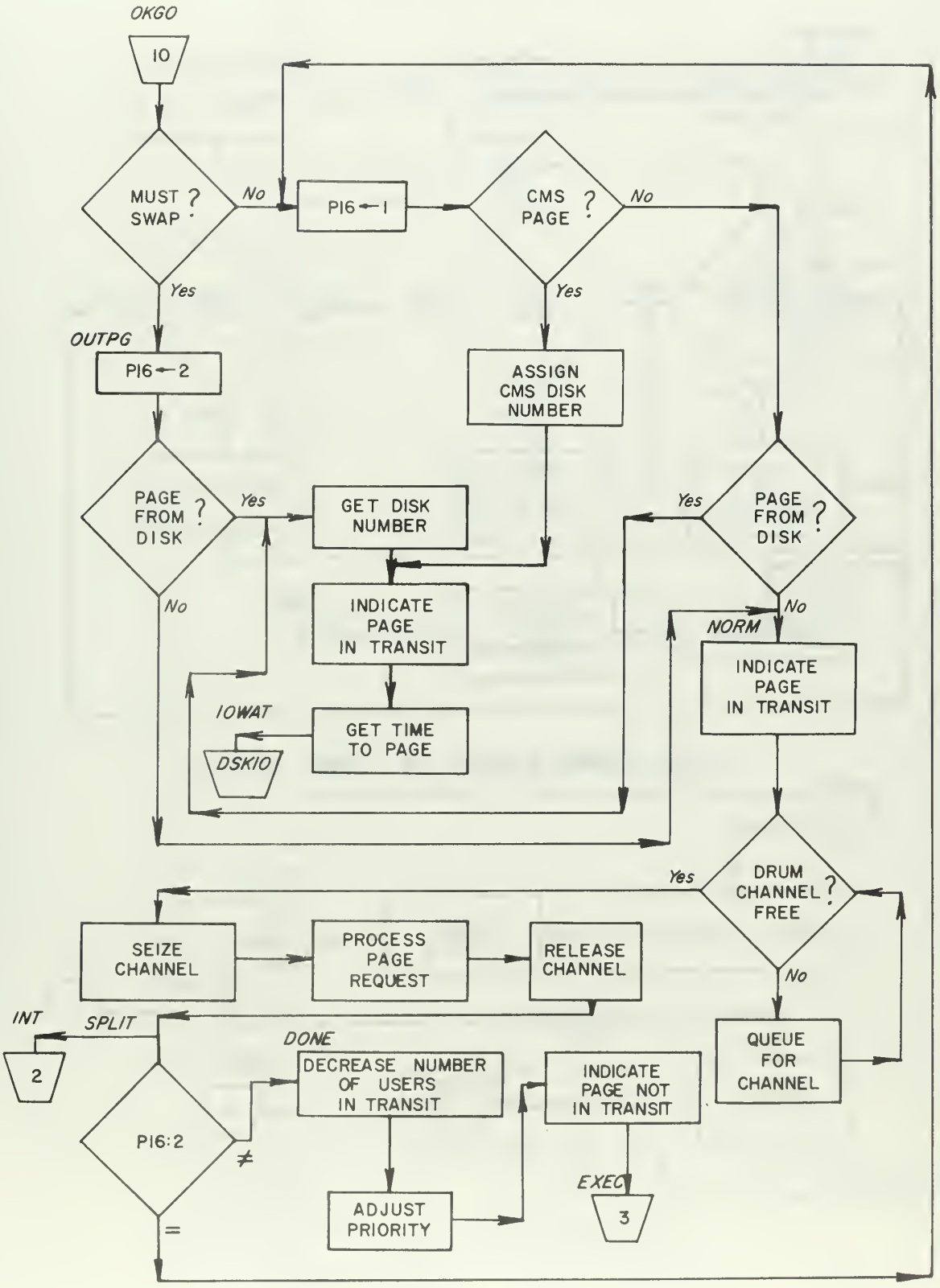

IOWAT
8
SAVE CURRENT PRIORITY
PRIORITY ← 125
SPLIT
INT
2
GET TIME TO NEXT I/O REQUEST
LOCK PAGE IN CORE
ASSIGN DISK FOR I/O
GET TIME TO PROCESS I/O REQUEST
DSKIO
DISK ARM FREE ?
No
QUEUE FOR ARM
Yes
GRAB3
SEIZE ARM
CHANNEL ? FREE
No
QUEUE FOR CHANNEL
Yes
ADVANCE SEEK TIME
CHANNEL ? FREE
No
QUEUE FOR CHANNEL
Yes
GRAB1
SEIZE CHANNEL
ADVANCE PROCESS I/O REQUEST
RELEASE ARM & CHANNEL
UNLOCK PAGE
SPLIT
INT
2
ADJUST PRIORITY
EXEC
3

PAGING

A. CORE STATUS TABLE LOOK-UP

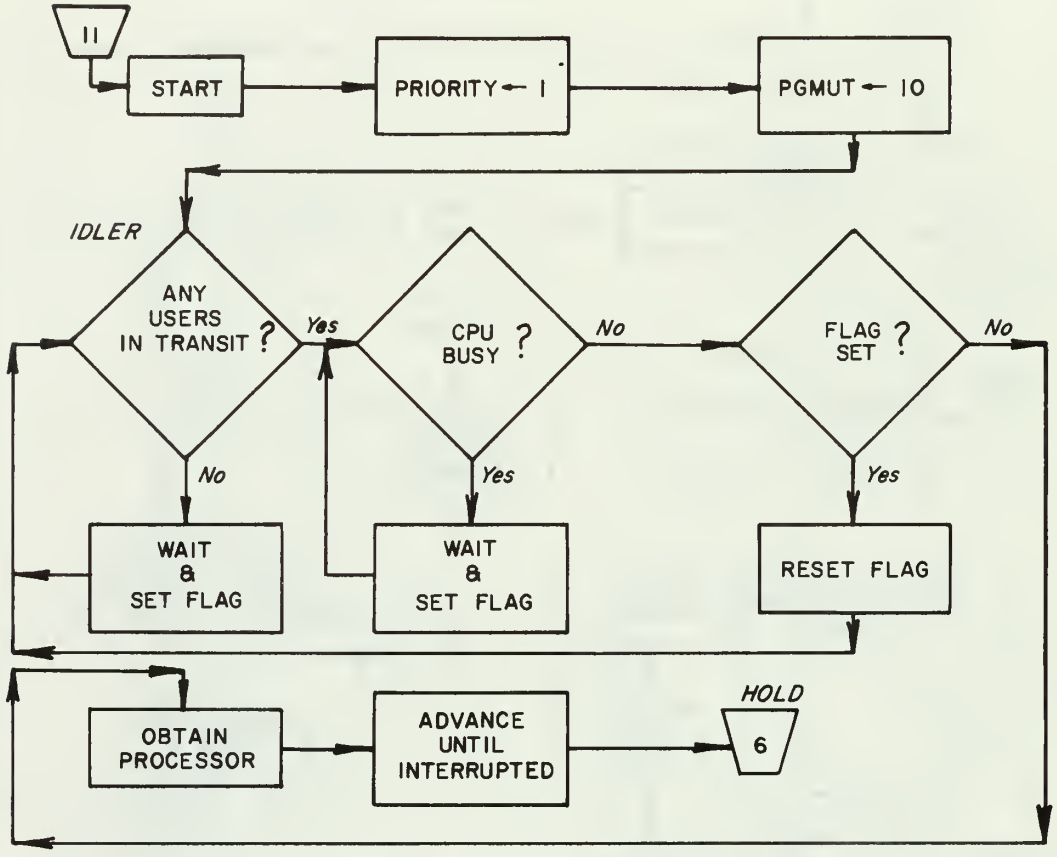


B. PROCESSING REQUEST

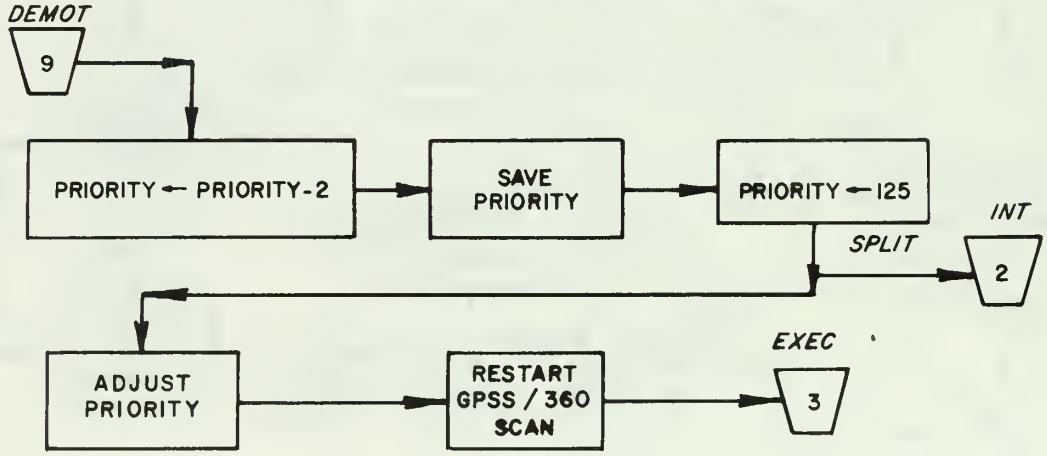


SUPPLEMENTARY ROUTINES

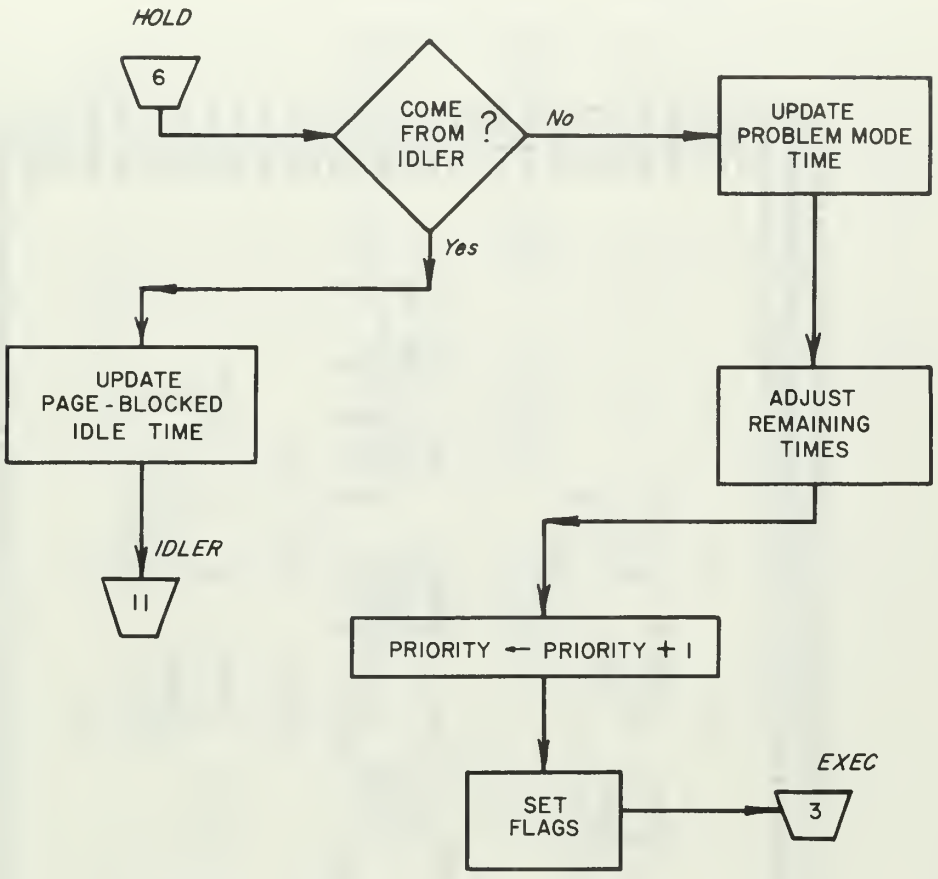
A. TIMER FOR PAGE-BLOCKED IDLE TIME



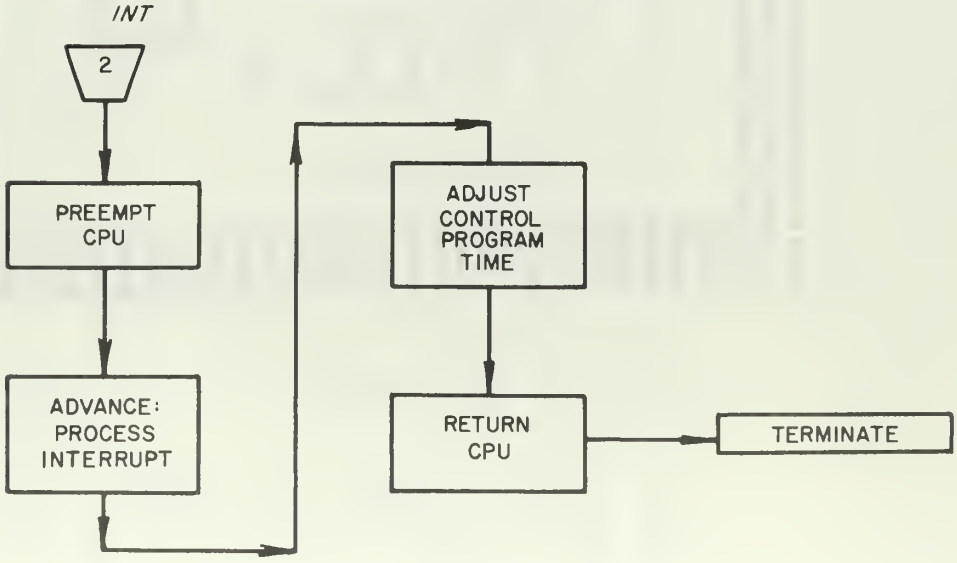
B. COMPLETION OF TIME SLICE

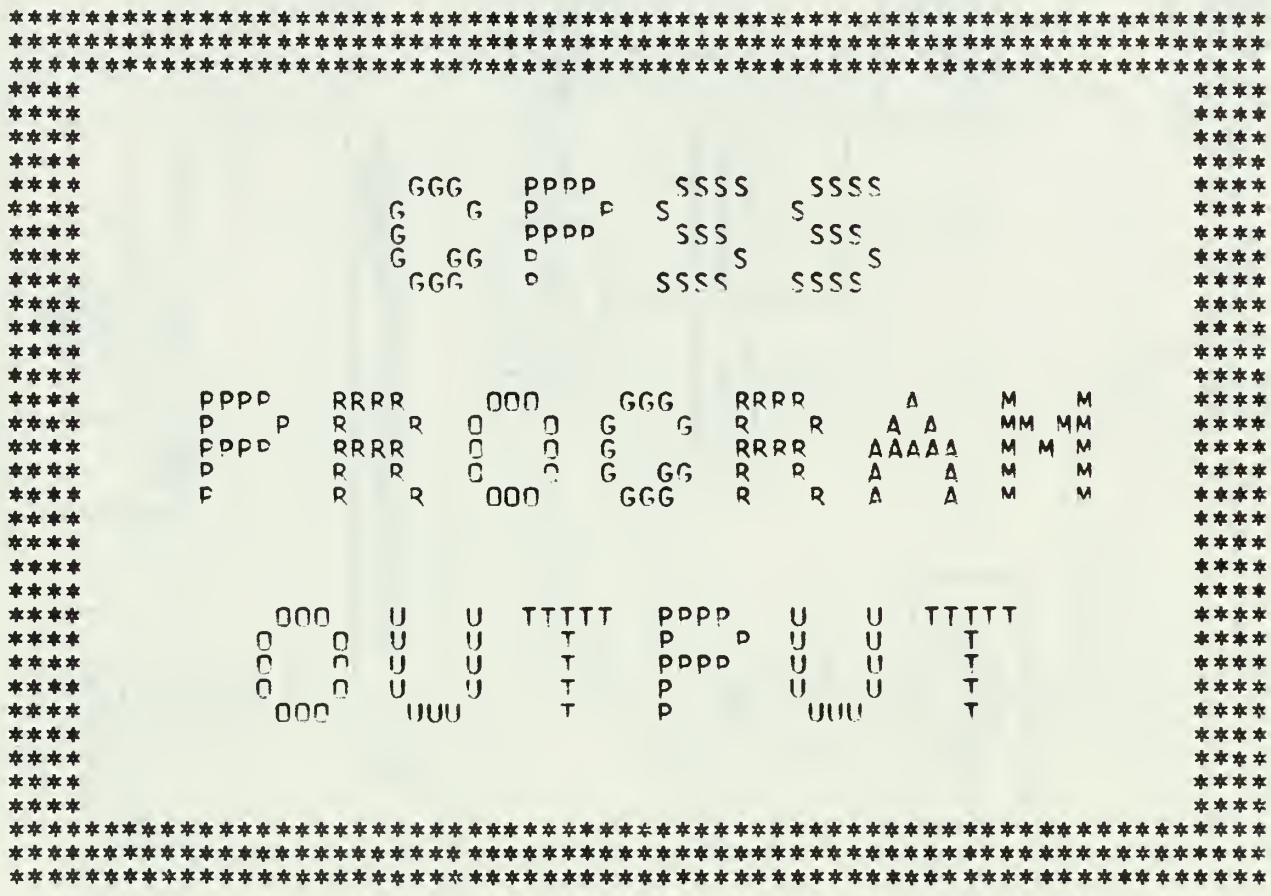


C. INTERRUPTED USERS



D. INTERRUPT HANDLING




GGG PPPP SSSS SSSS
G G P P S S
G PPPP SSS SSS
G GG P S S
GGG P SSSS SSSS
PPPP RRRR OOO GGG RRRR A M M
P P R R O O G G R R A A MM MM
PPPP RRRR O O G RRRR AAAAA M M M
P R R O O G GG R R A A M M
P R R OOO GGG R R A A M M
OOO U U TTTTT PPPP U U TTTTT
O O U U T P P U U T
O O U U T PPPP U U T
O O U U T P U U T
OOO UUU T P UUU T

```
CONTENTS OF FULLWORD SAVEVALUES (NON-ZERO)
SAVEVALUE  NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE
         OVRHD          126   EXCTN          456   IDLE           417   PAGIN           27   PGOUT          26

FACILITY        AVERAGE          NUMBER        AVERAGE          SEIZING    PREEMPTING
                UTILIZATION      ENTRIES       TIME/TRAN        TRANS. NO. TRANS. NO.
   CHAN4            .619             242          38.388          15
   DISK1            .047               5         141.199
   DISK2            .164              17         144.823
   DISK3            .821              95         129.663          15
   DISK4            .967             126         115.158          21

FACILITY        AVERAGE          NUMBER        AVERAGE          SEIZING    PREEMPTING
                UTILIZATION      ENTRIES       TIME/TRAN        TRANS. NO. TRANS. NO.
   CPU              .715            1335          8.041                      4
   CHAN2            .050              58         13.155
QUEUE       MAXIMUM     AVERAGE      TOTAL       ZERO       PERCENT      AVERAGE      $AVERAGE      TABLE      CURRENT
            CONTENTS    CONTENTS     ENTRIES     ENTRIES    ZEROS        TIME/TRANS   TIME/TRANS    NUMBER     CONTENTS
   CHAN4           3        .177         483         380       78.6          5.527       25.922
   DISK1           1        .000           5           5      100.0           .000         .000
   DISK2           1        .000          17          17      100.0           .000         .000
   DISK3           6       1.510          98          18       18.3        231.163      283.174                      4
   DISK4           7       3.283         125           7        5.5        394.071      417.448
$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

QUEUE       MAXIMUM     AVERAGE      TOTAL       ZERO       PERCENT      AVERAGE      $AVERAGE      TABLE      CURRENT
            CONTENTS    CONTENTS     ENTRIES     ENTRIES    ZEROS        TIME/TRANS   TIME/TRANS    NUMBER     CONTENTS
   CPU             3        .229         512                     .0          6.724        6.724
   CHAN2           1        .000          58          58      100.0           .000         .000
$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

CONTENTS OF FULLWORD SAVEVALUES (NON-ZERO)
SAVEVALUE  NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE
           322         16032

CONTENTS OF FULLWORD SAVEVALUES (NON-ZERO)
SAVEVALUE  NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE      NR,       VALUE
           322          9309
```

TABLE PGS

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS	
380	7.857	12.410	2986.000	NON-WEIGHTED

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
0	0	.00	.0	100.0	-.000	-.633
1	77	20.26	20.2	79.7	.127	-.552
2	116	30.52	50.7	49.2	.254	-.472
3	28	7.36	58.1	41.8	.381	-.391
4	19	4.99	63.1	36.8	.509	-.310
5	12	3.15	66.3	33.6	.636	-.230
6	16	4.21	70.5	29.4	.763	-.149
7	19	4.99	75.5	24.4	.890	-.069
8	12	3.15	78.6	21.3	1.018	.011
9	10	2.63	81.3	18.6	1.145	.092
10	8	2.10	83.4	16.5	1.272	.172
11	2	.52	83.9	16.0	1.399	.253
12	2	.52	84.4	15.5	1.527	.333
13	4	1.05	85.5	14.4	1.654	.414
14	1	.26	85.7	14.2	1.781	.494
15	3	.78	86.5	13.4	1.908	.575
16	6	1.57	88.1	11.8	2.036	.656
17	1	.26	88.4	11.5	2.163	.736
18	2	.52	88.9	11.0	2.290	.817
19	2	.52	89.4	10.5	2.417	.897
20	1	.26	89.7	10.2	2.545	.978
21	2	.52	90.2	9.7	2.672	1.058
22	0	.00	90.2	9.7	2.799	1.139
23	1	.26	90.5	9.4	2.926	1.220
24	0	.00	90.5	9.4	3.054	1.300
25	1	.26	90.7	9.2	3.181	1.381
26	0	.00	90.7	9.2	3.308	1.461
27	3	.78	91.5	8.4	3.436	1.542
28	2	.52	92.1	7.8	3.563	1.623
29	0	.00	92.1	7.8	3.690	1.703
30	1	.26	92.3	7.6	3.817	1.784
31	0	.00	92.3	7.6	3.945	1.864
32	2	.52	92.8	7.1	4.072	1.945
33	1	.26	93.1	6.8	4.199	2.025
34	1	.26	93.4	6.5	4.326	2.106
35	1	.26	93.6	6.3	4.454	2.187
36	1	.26	93.9	6.0	4.581	2.267
37	0	.00	93.9	6.0	4.708	2.348
38	1	.26	94.2	5.7	4.835	2.428
39	1	.26	94.4	5.5	4.963	2.509
40	1	.26	94.7	5.2	5.090	2.589
41	2	.52	95.2	4.7	5.217	2.670
42	1	.26	95.5	4.4	5.344	2.751
43	1	.26	95.7	4.2	5.472	2.831
44	0	.00	95.7	4.2	5.599	2.912
45	0	.00	95.7	4.2	5.726	2.992
46	1	.26	96.0	3.9	5.853	3.073
47	0	.00	96.0	3.9	5.981	3.154
48	0	.00	96.0	3.9	6.108	3.234
49	2	.52	96.5	3.4	6.235	3.315
50	0	.00	96.5	3.4	6.363	3.395
51	0	.00	96.5	3.4	6.490	3.476
52	3	.78	97.3	2.6	6.617	3.556
53	1	.26	97.6	2.3	6.744	3.637
54	2	.52	98.1	1.8	6.872	3.718
55	1	.26	98.4	1.5	6.999	3.798
56	3	.78	99.2	.7	7.126	3.879
57	3	.78	100.0	.0	7.253	3.959

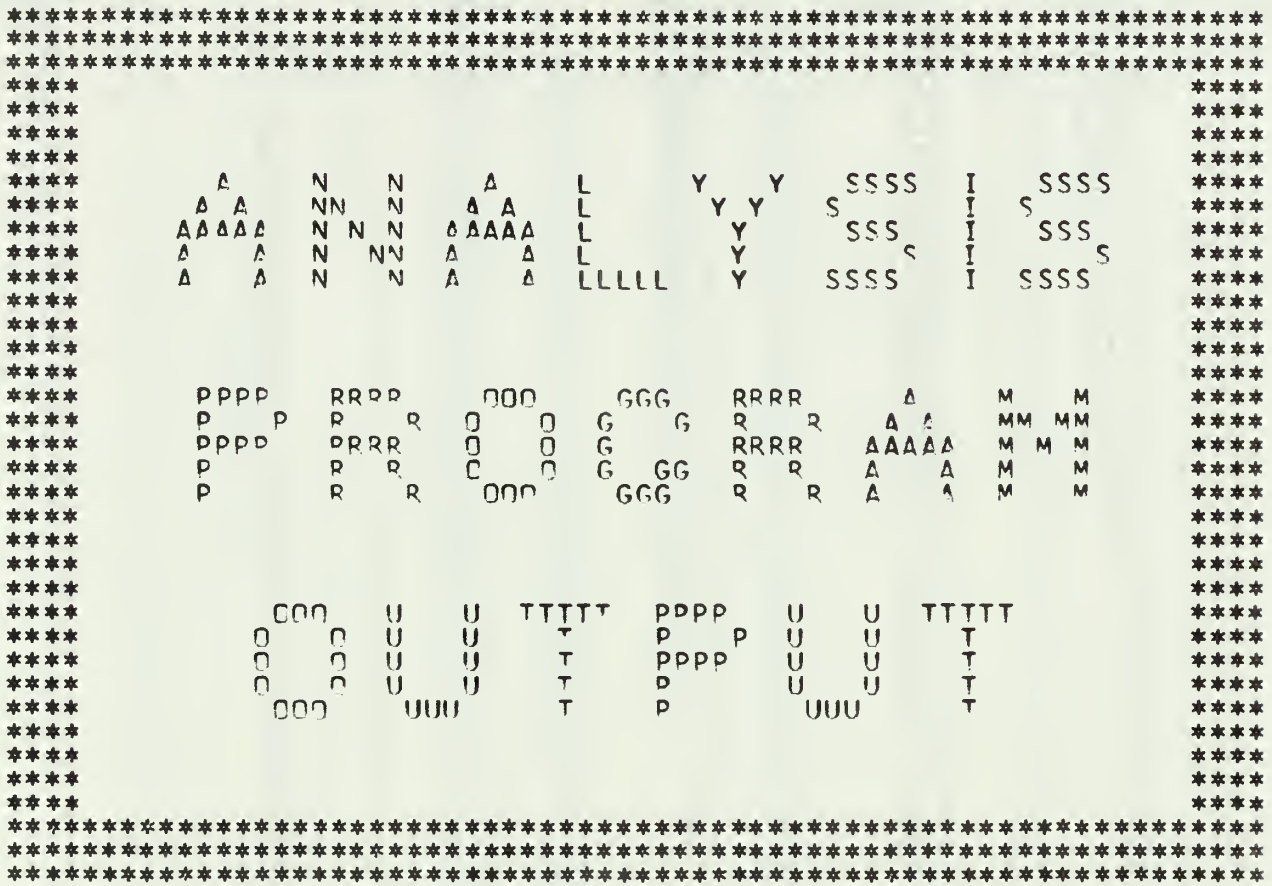
REMAINING FREQUENCIES ARE ALL ZERO

TABLE ZEKE

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS	
8511	75.127	21.812	639414.000	NON-WEIGHTED

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
20	0	.00	.0	100.0	.266	-2.527
22	0	.00	.0	100.0	.292	-2.435
24	0	.00	.0	100.0	.319	-2.343
26	68	.79	.7	99.2	.346	-2.252
28	67	.78	1.5	98.4	.372	-2.160
30	92	1.08	2.6	97.3	.399	-2.068
32	61	.71	3.3	96.6	.425	-1.977
34	65	.76	4.1	95.8	.452	-1.885
36	87	1.02	5.1	94.8	.479	-1.793
38	79	.92	6.0	93.9	.505	-1.702
40	85	.99	7.0	92.9	.532	-1.610
42	71	.83	7.9	92.0	.559	-1.518
44	82	.96	8.8	91.1	.585	-1.427
46	72	.84	9.7	90.2	.612	-1.335
48	100	1.17	10.9	89.0	.638	-1.243
50	76	.89	11.8	88.1	.665	-1.151
52	74	.86	12.6	87.3	.692	-1.060
54	71	.83	13.5	86.4	.718	-.968
56	93	1.09	14.6	85.3	.745	-.876
58	92	1.08	15.6	84.3	.772	-.785
60	78	.91	16.6	83.3	.798	-.693
62	97	1.13	17.7	82.2	.825	-.601
64	86	1.01	18.7	81.2	.851	-.510
66	753	8.84	27.5	72.4	.878	-.418
68	739	8.68	36.2	63.7	.905	-.326
70	791	9.29	45.5	54.4	.931	-.235
72	778	9.14	54.7	45.2	.958	-.143
74	768	9.02	63.7	36.2	.984	-.051
76	150	1.76	65.5	34.4	1.011	.039
78	141	1.65	67.1	32.8	1.038	.131
80	159	1.86	69.0	30.9	1.064	.223
82	134	1.57	70.6	29.3	1.091	.315
84	142	1.66	72.2	27.7	1.118	.406
86	153	1.79	74.0	25.9	1.144	.498
88	142	1.66	75.7	24.2	1.171	.590
90	151	1.77	77.5	22.4	1.197	.681
92	174	2.04	79.5	20.4	1.224	.773
94	133	1.56	81.1	18.8	1.251	.865
96	150	1.76	82.8	17.1	1.277	.956
98	124	1.45	84.3	15.6	1.304	1.048
100	138	1.62	85.9	14.0	1.331	1.140
102	150	1.76	87.7	12.2	1.357	1.231
104	138	1.62	89.3	10.6	1.394	1.323
106	160	1.87	91.2	8.7	1.410	1.415
108	82	.96	92.1	7.8	1.437	1.507
110	69	.81	92.9	7.0	1.464	1.598
112	41	.48	93.4	6.5	1.490	1.690
114	49	.57	94.0	5.9	1.517	1.782
116	58	.68	94.7	5.2	1.544	1.873
118	57	.66	95.4	4.5	1.570	1.965
120	42	.49	95.8	4.1	1.597	2.057
122	56	.65	96.5	3.4	1.623	2.148
124	49	.57	97.1	2.8	1.650	2.240
126	48	.56	97.6	2.3	1.677	2.332
128	48	.56	98.2	1.7	1.703	2.423
130	51	.59	98.8	1.1	1.730	2.515
132	46	.54	99.4	.5	1.757	2.607
134	51	.59	100.0	.0	1.783	2.699

REMAINING FREQUENCIES ARE ALL ZERO


ANNA LYSIS
PROGRAM
COUNTING

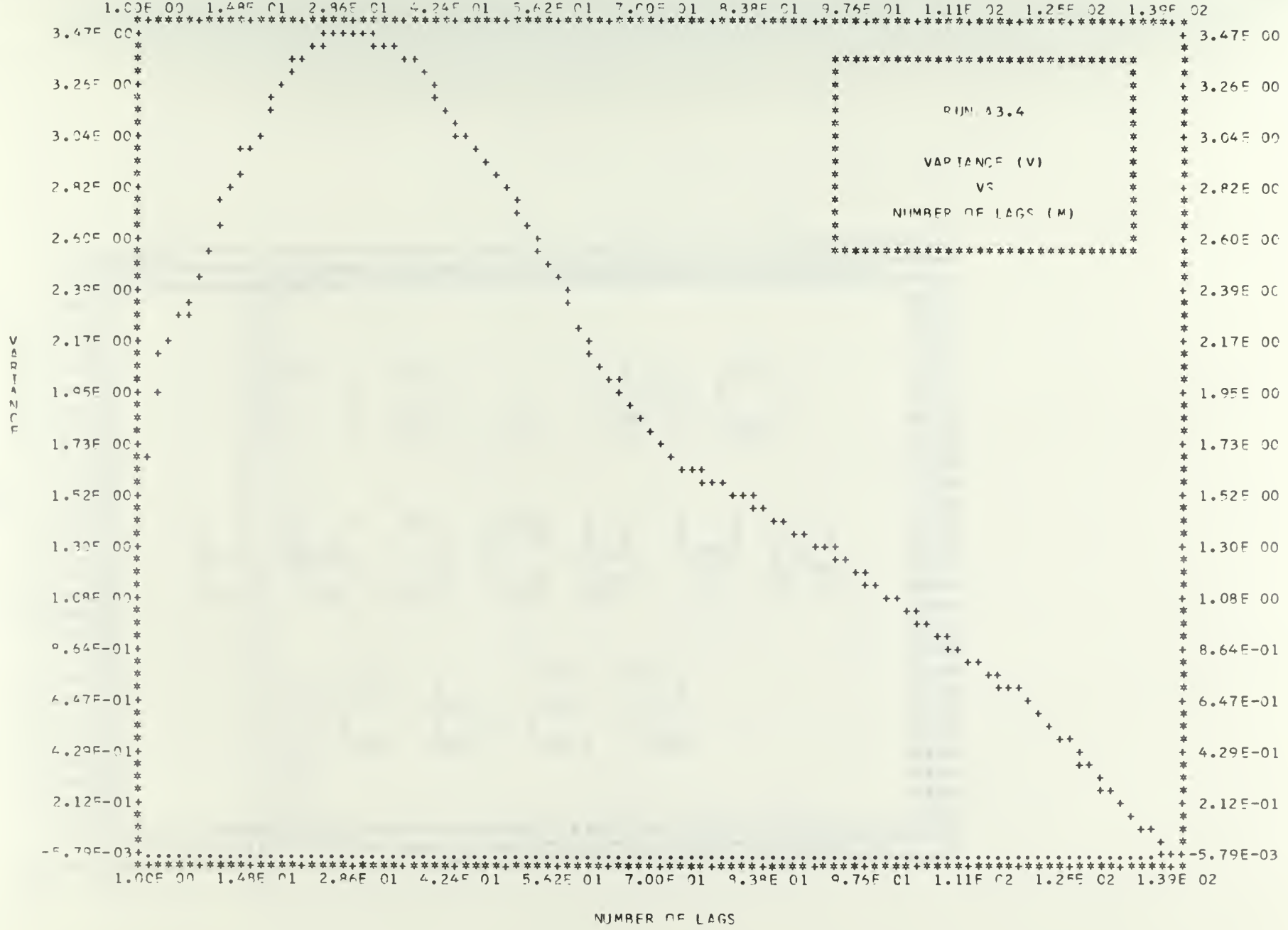
AVERAGE = 19.194916 R(0) = 340.501465

M	V	TAU	N	RATIO
1	1.702507	0.500000	200.000031	0.067976
2	1.959867	0.575583	173.736969	0.072933
3	2.107810	0.619031	161.542755	0.075636
4	2.168674	0.636906	157.009094	0.076720
5	2.255798	0.662493	150.944992	0.078246
6	2.299652	0.675372	148.066498	0.079003
7	2.346672	0.689181	145.099716	0.079807
8	2.413383	0.708773	141.088821	0.080933
9	2.529820	0.742969	134.595108	0.082863
10	2.638547	0.774900	129.048843	0.084624
11	2.750105	0.807663	123.813980	0.086395
12	2.825203	0.829718	120.522827	0.087567
13	2.897977	0.851091	117.496262	0.088687
14	2.959827	0.869255	115.040985	0.089629
15	3.009785	0.883927	113.131500	0.090382
16	3.053070	0.896639	111.527557	0.091029
17	3.128271	0.918724	108.846527	0.092144
18	3.202317	0.940471	106.329697	0.093228
19	3.264687	0.958788	104.298355	0.094131
20	3.308875	0.971765	102.905502	0.094766
21	3.354324	0.985113	101.511185	0.095415
22	3.385403	0.994240	100.579300	0.095856
23	3.415359	1.003037	99.697128	0.096279
24	3.428636	1.006936	99.311066	0.096466
25	3.453034	1.014102	98.609344	0.096809
26	3.456161	1.015020	98.520142	0.096852
27	3.462828	1.016978	98.330460	0.096946
28	3.470150	1.019129	98.122971	0.097048
29	3.473181	1.020019	98.037354	0.097091
30	3.467278	1.018285	98.204239	0.097008
31	3.459465	1.015990	98.426041	0.096899
32	3.444810	1.011686	98.844772	0.096693
33	3.426119	1.006197	99.384018	0.096431
34	3.405040	1.000007	99.999252	0.096133
35	3.384580	0.993998	100.603760	0.095844
36	3.359676	0.986685	101.349472	0.095491
37	3.339048	0.980627	101.975586	0.095197
38	3.313600	0.973153	102.758774	0.094834
39	3.267151	0.959512	104.219681	0.094167
40	3.220267	0.945742	105.737015	0.093489

N	V	TAU	N	RATIC
41	3.170778	0.931208	107.387344	0.092768
42	3.119086	0.916027	109.167053	0.092008
43	3.062401	0.899379	111.187744	0.091168
44	3.018242	0.886411	112.814499	0.090509
45	2.982254	0.875842	114.175873	0.089968
46	2.943240	0.864384	115.689316	0.089377
47	2.909844	0.854576	117.017059	0.088869
48	2.872313	0.843553	118.546097	0.088294
49	2.829526	0.830988	120.338684	0.087634
50	2.774668	0.814877	122.717911	0.086780
51	2.711845	0.796427	125.560791	0.085792
52	2.647028	0.777391	128.635376	0.084760
53	2.588005	0.760057	131.569077	0.083810
54	2.536173	0.744834	134.257980	0.082967
55	2.483339	0.729318	137.114349	0.082098
56	2.431016	0.713952	140.065491	0.081228
57	2.372620	0.696802	143.512863	0.080247
58	2.312676	0.679197	147.232635	0.079227
59	2.249325	0.660591	151.379410	0.078134
60	2.192421	0.643880	155.308426	0.077139
61	2.136976	0.627596	159.337967	0.076158
62	2.083890	0.612006	163.397034	0.075206
63	2.031816	0.596713	167.584824	0.074260
64	1.992534	0.585176	170.888672	0.073539
65	1.950492	0.572829	174.572098	0.072759
66	1.906364	0.559869	178.612991	0.071931
67	1.853041	0.544209	183.752823	0.070918
68	1.805246	0.530173	188.617722	0.069997
69	1.762256	0.517547	193.219101	0.069159
70	1.725343	0.506706	197.352936	0.068431
71	1.697007	0.498385	200.648209	0.067866
72	1.671883	0.491006	203.663498	0.067362
73	1.650151	0.484623	206.345596	0.066923
74	1.627500	0.477971	209.217529	0.066462
75	1.602856	0.470734	212.434250	0.065957
76	1.584039	0.465207	214.957779	0.065569
77	1.564500	0.459469	217.642365	0.065163
78	1.546419	0.454159	220.187042	0.064785
79	1.532859	0.450176	222.134903	0.064501
80	1.522850	0.447237	223.594864	0.064290

M	V	TAU	N	RATIC
81	1.514798	0.444872	224.783386	C.064120
82	1.500520	0.440679	226.922333	C.063817
83	1.477681	0.433972	230.429581	C.063329
84	1.452451	0.426562	234.432358	0.062786
85	1.426054	0.418810	238.771774	C.062213
86	1.406684	0.413121	242.059677	0.061789
87	1.386301	0.407135	245.618698	0.061340
88	1.369825	0.402296	248.572891	0.060974
89	1.354045	0.397662	251.469833	0.060622
90	1.337251	0.392729	254.627991	C.060245
91	1.316263	0.386566	258.687744	0.059770
92	1.297286	0.380993	262.471924	0.059338
93	1.275600	0.374624	266.934082	0.058840
94	1.250832	0.367350	272.219971	0.058266
95	1.227375	0.360461	277.422363	0.057717
96	1.202277	0.353090	283.213623	0.057124
97	1.178132	0.345999	289.018066	0.056547
98	1.151188	0.338086	295.782471	0.055897
99	1.125871	0.330651	302.433838	0.055279
100	1.107963	0.325391	307.322021	0.054837
101	1.089328	0.319919	312.579346	0.054374
102	1.064063	0.312499	320.000977	0.053740
103	1.037298	0.304638	328.257812	C.053060
104	1.009708	0.296536	337.227295	0.052349
105	0.981774	0.288332	346.822510	0.051620
106	0.953284	0.279965	357.187500	0.050866
107	0.927777	0.272474	367.007813	0.050181
108	0.908893	0.266928	374.633301	0.049667
109	0.888956	0.261073	383.034912	0.049120
110	0.864530	0.253899	393.856934	0.048440
111	0.835932	0.245500	407.331543	0.047632
112	0.809866	0.237845	420.441406	0.046884
113	0.786368	0.230944	433.004883	0.046198
114	0.760957	0.223481	447.464600	0.045446
115	0.738962	0.217022	460.783447	0.044784
116	0.723250	0.212407	470.793701	0.044306
117	0.705395	0.207164	482.709961	0.043755
118	0.680740	0.199923	500.193115	0.042984
119	0.652719	0.191693	521.666260	0.042090
120	0.622007	0.182674	547.423340	0.041088

M	V	TAU	N	RATIO
121	0.589912	0.173248	577.207275	0.040014
122	0.559776	0.164398	608.281006	0.038578
123	0.531105	0.155977	641.118652	0.037967
124	0.502789	0.147661	677.225342	0.036941
125	0.474305	0.139296	717.895264	0.035879
126	0.439696	0.129132	774.402588	0.034545
127	0.401459	0.117902	848.160400	0.033009
128	0.360848	0.105976	943.613525	0.031295
129	0.321768	0.094498	1058.221680	0.029552
130	0.281227	0.082592	1210.771729	0.027628
131	0.241860	0.071031	1407.843262	0.025621
132	0.206185	0.060553	1651.434326	0.023656
133	0.173915	0.051076	1957.863525	0.021726
134	0.144034	0.042301	2364.028809	0.019772
135	0.112052	0.032908	3038.781738	0.017439
136	0.079944	0.023478	4259.257812	0.014730
137	0.047843	0.014051	7117.054687	0.011395
138	0.019327	0.005676	17617.558594	0.007243


RUN 43.4
VARIANCE (V)
VS
NUMBER OF LAGS (M)
VARIANCE
NUMBER OF LAGS
1.00E 00
1.48E 01
2.86E 01
4.24E 01
5.62E 01
7.00E 01
8.38E 01
9.76E 01
1.11E 02
1.25E 02
1.39E 02
3.47E 00
3.26E 00
3.04E 00
2.82E 00
2.60E 00
2.39E 00
2.17E 00
1.95E 00
1.73E 00
1.52E 00
1.30E 00
1.08E 00
8.64E-01
6.47E-01
4.29E-01
2.12E-01
-5.79E-03


GGG PPPP SSSS SSSS
G G P P S S
G PPPP SSS SSS
G GG P S S
GGG P SSSS SSSS
PPPP RRRR OOO GGG RRRR A M M
P P R R O O G G R R A A MM MM
PPPP RRRR O O G RRRR AAAAA M M M
P R R O O G GG R R A A M M
P R R OOO GGG R R A A M M
L I SSSS TTTTT I N N GGG
L I S T I NN N G G
L I SSS T I N N G
L I S T I N NN G GG
LLLLL I SSSS T I N N GGG

```
        SIMULATE
        RMULT     72519,85657,98713,11471,45003,25473,13879,96355
PMT     EQU       303,X
PBT     EQU       304,X
CPT     EQU       305,X
UIT     EQU       306,X
PGMUT   EQU       307,X
TPMT    EQU       308,X
TCPT    EQU       309,X
TIDL    EQU       310,X
OVRHD   EQU       311,X
EXCTN   EQU       312,X
IDLE    EQU       313,X
PAGIN   EQU       314,X
PGOUT   EQU       315,X
PGCPT   EQU       316,X
PGPMT   EQU       317,X
CHAN2   EQU       18,Q,F,C
CPU     EQU       17,Q,F
CPUHI   EQU       1,V
CPULO   EQU       2,V
PAGHI   EQU       3,V
PAGLO   EQU       4,V
IOHI    EQU       5,V
IOLO    EQU       6,V
CHANA   EQU       1,F,Q,C
CHANB   EQU       2,F,C,Q
DISK1   EQU       3,F,Q,C
DISK2   EQU       4,F,C,Q
DISK3   EQU       5,F,Q,C
DISK4   EQU       6,F,Q,C
DISK5   EQU       7,F,Q,C
DISK6   EQU       8,F,Q,C
DISK7   EQU       9,F,Q,C
DISK8   EQU       10,F,Q,C
TOTAL   EQU       3,T
PROCE   EQU       4,T
XTIME   EQU       5,T
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***      FUNCTIONS & VARIABLES       ***
                    ***                                  ***
                    ****************************************
                    ****************************************

 NRPAG FUNCTION     RN7,C14
0       0       .2      1       .5      2       .56     3       .61     4       .65     5
.69     6       .73     7       .76     8       .78     9       .80     10      .86     16
.90     25      1.0     57

 REACT FUNCTION     RN1,C36
0       0       .014    1       .046    2       .086    3       .130    4       .179    5
.235    6       .306    7       .361    8       .411    9       .457    10      .500    11
.539    12      .565    13      .588    14      .609    15      .629    16      .648    17
.665    18      .681    19      .696    20      .709    21      .721    22      .733    23
.744    24      .754    25      .764    26      .773    27      .782    28      .791    29
.799    30      .837    35      .867    60      .900    100     .950    200     1.0     300

 CPUTI FUNCTION     RN2,D2
.600    2       1.0     1

 PAGTI FUNCTION     RN8,D2
.200    3       1.0     4

 IOTIM FUNCTION     RN4,D2
.800    5       1.0     6

 PGDSK FUNCTION     RN1,S2,F
.50     4       1.0     4

 DISK  FUNCTION     RN6,S6,F
.167    5       .333    6       .500    7       .667    8       .833    9       1.0     10

 SEEK  FUNCTION     RN4,C5
0       25      .19     65      .64     75      .92     108     1.0     135

 EXPON FUNCTION     RN3,C24
0       0       .1      .104    .2      .222    .3      .355    .4      .509    .5      .69
.6      .915    .7      1.2     .75     1.38    .8      1.6     .84     1.83    .88     2.12
.9      2.3     .92     2.52    .94     2.81    .95     2.99    .96     3.2     .97     3.5
.98     3.9     .99     4.6     .995    5.3     .998    6.2     .999    7.0     .9997   8.0

 TIMER FUNCTION     PR,D4
84      4       88      3       94      2       102     1
```

```
RAISE  VARIABLE    PR+1              NEXT HIGHER PRIORITY
RASE2  VARIABLE    PR+2              NEXT HIGHER PRIORITY LEVEL
LOWER  VARIABLE    PR-1              NEXT LOWER PRIORITY
LOWR2  VARIABLE    PR-2              NEXT LOWER PRIORITY LEVEL
SAVE1  VARIABLE    P7+100            POINTER
SEC15  VARIABLE    X$LST15+15000     TIME TO NEXT 15 SECOND CHECK
SCDUP  BVARIABLE   (LS*8)*(LR*7)     PRIORITY INCREASE CONDITIONS
SAVE2  VARIABLE    P8+100            POINTER
SAVE3  VARIABLE    P7+200              "
CPUHI  FVARIABLE   2000*FN$EXPON     HIGH CPU TIME
CPULO  FVARIABLE   25*FN$EXPON       LOW CPU TIME
PAGHI  FVARIABLE   RN6/50+10         HIGH PAGING RATE
PAGLO  FVARIABLE   RN6/50+40         LOW PAGING RATE
IOHI   FVARIABLE   RN5/40+10         HIGH I/O RATE
IOLO   FVARIABLE   RN5/40+35         LOW I/O RATE
HOUR   VARIABLE    3600000           ONE HOUR IN MILLISECONDS
MINIT  VARIABLE    60000             ONE MINUTE IN MILLISECONDS
BTS    VARIABLE    50                BASIC TIME SLICE IN MILLISECONDS
DPT    VARIABLE    RN8@50+26         DISK PAGING TIME
DIOT   VARIABLE    RN8@75+1          DISK I/O TIME
TIMES  VARIABLE    FN$TIMER*V$BTS    PRIORITY DEPENDENT TIME SLICE
COR1   VARIABLE    V$CORE+1          POINTER
COR2   VARIABLE    V$CORE*2            "
COR3   VARIABLE    P18+V$CORE          "
COR4   VARIABLE    P25-V$CORE          "
INDX   VARIABLE    P20+V$CORE          "
RFB    VARIABLE    (XH*21-XH*21@1000)+XH*21@100   RESET FIFO BIT
CHNG   VARIABLE    (XH*25-XH*25@100)+11       RESET CHANGED BIT
SIZE   FVARIABLE   FN$NRPAG+1        NUMBER OF PAGES
PGNUM  VARIABLE    RN7@P17+1+100*P7      PAGE NUMBER
CORE   VARIABLE    40                PAGES AVAILABLE
ADJST  VARIABLE    X*15-P6           ADJUSTED TIMES
INRPT  FVARIABLE   FN$EXPON+2        INTERRUPT PROCESSING TIME
TERMS  VARIABLE    12                NUMBER OF USER TERMINALS
PBIT   VARIABLE    90000-P6          PAGE BLOCKED IDLE TIME
RATIO  FVARIABLE   ((X$PBT+X$PGCPT+X$PGPMT)/X$PBT)*10
PGMAX  VARIABLE    20                BOUND ON RATIO
PGMIN  VARIABLE    50                BOUND ON RATIO
MAX    VARIABLE    V$TERMS           MAXIMUM # USERS IN TRANSIT
PCPT   FVARIABLE   1000*X$TCPT/C1    PERCENT CONTROL PROGRAM TIME
PPMT   FVARIABLE   1000*X$TPMT/C1    PERCENT PROBLEM MODE TIME
PIT    FVARIABLE   1000*X$TIDL/C1    PERCENT IDLE TIME
IDL    VARIABLE    C1-X$TPMT-X$TCPT      IDLE TIME
THINK  FVARIABLE   1000*FN$REACT+1   USER REACTION TIME
INRTE  FVARIABLE   10000*N$INPAG/C1      PAGES IN
OUTRT  FVARIABLE   10000*N$OUTPG/C1      PAGES OUT
AVGTM  FVARIABLE   X320/5            AVERAGE RESPONSE TIME
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***              TIMING              ***
                    ***                                  ***
                    ***          15 SECOND CHECK         ***
                    ***                                  ***
                    ***            & BOOKEEPING          ***
                    ***                                  ***
                    ****************************************
                    ****************************************

        GENERATE    15000              CREATE TRANSACTION EVERY 15 SECONDS
        SAVEVALUE   318+,K1            COUNTER
        SAVEVALUE   319+,K1            COUNTER
        ASSIGN      8,K1               CHECK EACH RUNNABLE USER TO DETERMINE
        ASSIGN      7,V$SAVE2          IF USER RUN SINCE LAST 15 SECOND CHECK
STEP2   TEST E      BV$SCDUP,1,END     SHOULD BUMP PRIORITY LEVEL?
        ALTER       P8,ALL,PR,X*7      ALTER PRIORITY IF APPROPRIATE
        LOGIC S     P7                 SET INDICATOR SWITCH
        TRANSFER    ,END+1
END     LOGIC R     P7                 RESET INDICATOR SWITCH
        ASSIGN      8+,1               ADD ONE TO COUNTER
        ASSIGN      7+,1                    "
        TEST G      P8,V$TERMS,STEP2       CHECK COMPLETED?
        SAVEVALUE   PGCPT+,X$CPT       BOOKEEPING
        SAVEVALUE   PGPMT+,X$PMT       BOOKEEPING
        SAVEVALUE   TPMT+,X$PMT        BOOKEEPING
        SAVEVALUE   TCPT+,X$CPT        BOOKEEPING
        SAVEVALUE   TIDL,V$IDL         BOOKEEPING
        SAVEVALUE   OVRHD,V$PCPT       BOOKEEPING
        SAVEVALUE   EXCTN,V$PPMT       BOOKEEPING
        SAVEVALUE   IDLE,V$PIT         BOOKEEPING
        SAVEVALUE   PAGIN,V$INRTE      BOOKEEPING
        SAVEVALUE   PGOUT,V$OUTRT      BOOKEEPING
        SAVEVALUE   CPT,K0             RESET CONTROL PROGRAM TIME
        SAVEVALUE   PMT,K0             RESET PROBLEM MODE TIME
```

```
                ****************************************
                ****************************************
                ***                                  ***
                ***              OUTPUT              ***
                ***                                  ***
                ***                &                 ***
                ***                                  ***
                ***         60 SECOND CHECK          ***
                ***                                  ***
                ****************************************
                ****************************************

        TEST GE     X319,K13,MINUT  WARM UP PERIOD OVER?
        PRINT       311,315,X,D
        PRINT       1,6,F
        PRINT       17,18,F
        PRINT       1,6,Q,2
        PRINT       17,18,Q
MINUT   TEST GE     X318,K4,FINEE   TIME FOR 60 SECOND CHECK?
        SAVEVALUE   318,K0          YES.  RESET COUNTER
        TEST LE     X$PBT,K0,*+3    PAGE BLOCKED TIME < = 0?
        SAVEVALUE   PGMUT,V$MAX     MAXIMIZE # USERS IN TRANSIT
        TRANSFER    ,ZERO           UNCONDITIONAL BRANCH
        TEST L      V$RATIO,V$PGMAX,*+4  RATIO < PGMAX?
        TEST G      X$PGMUT,K2,ZERO PGMUT > 2?
        SAVEVALUE   PGMUT-,K1       REDUCE PGMUT BY ONE
        TRANSFER    ,ZERO
        TEST G      V$RATIO,V$PGMIN,ZERO RATIO > PGMIN?
        TEST L      X$PGMUT,V$MAX,ZERO   PGMUT < MAXIMUM?
        SAVEVALUE   PGMUT+,K1       YES.  RAISE PGMUT BY ONE
ZERO    SAVEVALUE   PBT,K0
        SAVEVALUE   PGCPT,K0
        SAVEVALUE   PGPMT,K0
FINEE   TERMINATE   1               GET RID OF THIS TRANSACTION
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***            MAIN PROGRAM          ***
                    ***                                  ***
                    ****************************************
                    ****************************************

          GENERATE  ,,,V$TERMS,125,26,F  CREATE TRANSACTIONS
          SAVEVALUE 300+,1               NUMBER TRANSACTIONS
          ASSIGN    7,X300               ASSIGN TERMINAL NUMBERS
          ASSIGN    18,P7                INITIALIZE ARRAY POINTER
          ASSIGN    25,V$COR3                        "
          JOIN      P7                   JOIN GROUP  (PRIORITY ALTERATION)
*
*
 DOIT     ADVANCE   V$THINK              USER REACTION TIME & TERMINAL I/O
          LOGIC S   V$SAVE1              NOT ELIGIBLE FOR PRIORITY INCREASE
          ASSIGN    13,FN$CPUTI          ASSIGN EXECUTION TIMES
          ASSIGN    4,V*13                           "
          ASSIGN    17,V$SIZE            ASSIGN NUMBER OF PAGES REQUIRED
          ASSIGN    22,FN$PAGTI          ASSIGN PAGING RATE
          ASSIGN    1,V*22               ASSIGN PAGING TIME
          ASSIGN    23,FN$IOTIM          ASSIGN I/O RATE
          ASSIGN    2,V*23               ASSIGN I/O TIME
          ASSIGN    3,V$TIMES            TIME SLICE
          ASSIGN    8,P7                 POINTER
          ASSIGN    9,P7                   "
          ASSIGN    14,1                 CHANNEL NUMBER
          ASSIGN    15,V$SAVE3           POINTER
          SPLIT     1,INT                INTERRUPT CPU
          PRIORITY  100                  INITIAL PRIORITY LEVEL
          BUFFER                         RESTART SCAN
          MARK                           RECORD ENTRY TIME FOR TRANSACTION
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***            COMMENCE              ***
                    ***                                  ***
                    ***            EXECUTION             ***
                    ***                                  ***
                    ****************************************
                    ****************************************

EXEC   SAVEVALUE   V$SAVE1,V$RASE2 SAVE NEXT HIGHER PRIORITY LEVEL
       LOGIC S     P8              INDICATE RUNNABLE
BACK   QUEUE       CPU             QUEUE FOR PROCESSOR
       PREEMPT     CPU,PR,HOLD-2,6,RE   GRAB CPU IF PRIORITY HIGH
       LOGIC R     P8              NOT RUNNABLE
       TEST E      P11,1,*+3       COME FROM HOLD?
       ASSIGN      11,K0           YES.   RESET INDICATOR
       PRIORITY    V$LOWER         LOWER PRIORITY BY ONE
       TEST NE     XH*18,P19,*+4   PAGE IN CORE?
       DEPART      CPU             NO - LEAVE QUEUE FOR CPU
       RETURN      CPU             RETURN CPU
       TRANSFER    ,PAGEW+1        GET PAGE
       TRANSFER    .250,*+1,*+2    INDICATE PAGE CHANGED
       SAVEVALUE   *25,V$CHNG,H
SLICE  ASSIGN      5,V$TIMES       ASSIGN TIME SLICE
       LOGIC S     V$SAVE1         HAS RUN
       DEPART      CPU             DEPART QUEUE FOR CPU
       TEST NE     X55,P7,TIME     LAST INTERRUPTED USER?
       ASSIGN      3,P5            NO.  GET NEW TIME SLICE
TIME   SAVEVALUE   55,K0            RESET LAST INTERRUPTED USER INDICATOR
       SAVEVALUE   81,P1           SAVE APPROPRIATE TIMES
       SAVEVALUE   82,P2                    "
       SAVEVALUE   83,P3                    "
       SAVEVALUE   84,P4                    "
       SELECTMIN   10,81,84,,X     GET TIME TO ADVANCE
       ASSIGN      10-,K80         ADJUST ADVANCE TIME INDICATOR
       SAVEVALUE   P15,P*10        SAVE ADVANCE TIME
       ADVANCE     P*10            PROCESS
       SAVEVALUE   PMT+,P*10       RECORD PROBLEM MODE TIME
       ASSIGN      1-,X*15         ADJUST TIMES
       ASSIGN      2-,X*15              "
       ASSIGN      3-,X*15              "
       ASSIGN      4-,X*15              "
       RETURN      CPU             RETURN PROCESSOR
```

```
                ****************************************
                ****************************************
                ***                                  ***
                ***          PROCESSING              ***
                ***                                  ***
                ***          COMPLETED               ***
                ***                                  ***
                ****************************************
                ****************************************

NEXTB  TRANSFER   P,10,NEXTB      TAKE APPROPRIATE EXIT
       TRANSFER   ,PAGEW                  "
       TRANSFER   ,IOWAT                  "
       TRANSFER   ,DEMOT                  "
       PRIORITY   125             IF GET HERE  -  ALL DONE
       SPLIT      1,INT           INTERRUPT CPU
       BUFFER                     RESTART SCAN
       TEST GE    X319,K13,DOIT   WARM UP PERIOD OVER?
       TABULATE   TOTAL
       SAVEVALUE  320+,M1         ACCUMULATE TRANSIT TIMES
       SAVEVALUE  321+,K1         COUNTER
       TEST GE    X321,K5,DOIT    FIVE JOBS DONE?
       SAVEVALUE  321,K0          YES.  RESET COUNTER
       SAVEVALUE  322,V$AVGTM     CALCULATE AVERAGE TRANSIT TIME
       PRINT      322,322,X,D
       SAVEVALUE  323+,K1         COUNTER
       SAVEVALUE  320,K0          RESET COUNTER
       TEST GE    X323,K200,DOIT  TEST FOR DONE
       TEST GE    X319,K113,DOIT        "
       PRINT      3,3,T
COMPL  SPLIT      1,COMPL         COMPLETE ALL TRANSACTIONS
GOODO  TERMINATE  1
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***          MISCELLANEOUS           ***
                    ***                                  ***
                    ***             ROUTINES             ***
                    ***                                  ***
                    ****************************************
                    ****************************************

*
*   COME HERE WHEN TIME SLICE COMPLETED
*
 DEMOT PRIORITY    V$LOWR2          DOWN ONE PRIORITY LEVEL
       ASSIGN      12,PR            SAVE CURRENT PRIORITY
       PRIORITY    125              PRIORITY BUMP FOR INTERRUPT
       SPLIT       1,INT            CREATE AN INTERRUPT
       PRIORITY    P12              BACK TO VALID PRIORITY AGAIN
       BUFFER                       RESTART THE SCAN
       TRANSFER    ,EXEC            BACK TO QUEUE FOR CPU
*
*   COME HERE WHEN INTERRUPTED
*
 HOLD  TEST NE     PR,K1,HOLD2      COME FROM IDLER?
       SAVEVALUE   PMT+,V$ADJST     NO.  UPDATE PROBLEM MODE TIME
       ASSIGN      1-,V$ADJST       ADJUST REMAINING TIMES
       ASSIGN      2-,V$ADJST                 "
       ASSIGN      3-,V$ADJST                 "
       ASSIGN      4-,V$ADJST                 "
       PRIORITY    V$RAISE          RAISE PRIORITY TEMPORARILY
       SAVEVALUE   55,P7            INDICATE LAST INTERRUPTED USER
       ASSIGN      11,K1            SET INDICATOR
       TRANSFER    ,EXEC            BACK TO QUEUE FOR CPU
 HOLD2 SAVEVALUE   PBT+,V$PBIT      CAME FROM IDLER. ADJUST CONTROL TIME
       TRANSFER    ,IDLER           BACK TO IDLER
*
*   INTERRUPTS COME HERE
*
 INT   PREEMPT     CPU,PR,HOLD,6,RE      OBTAIN PROCESSOR
       ASSIGN      13,V$INRPT       TIME TO PROCESS INTERRUPT
       ADVANCE     P13              PROCESS THE INTERRUPT
       SAVEVALUE   CPT+,P13         UPDATE CONTROL PROGRAM TIME
       RETURN      CPU              RETURN PROCESSOR
 QUIT  TERMINATE                    INTERRUPT FINISHED
```

```
                ****************************************
                ****************************************
                ***                                  ***
                ***          INPUT / OUTPUT          ***
                ***                                  ***
                ****************************************
                ****************************************

IOWAT  ASSIGN      12,PR              RECORD CURRENT PRIORITY
       ASSIGN      26,K1              INDICATE I/O
       PRIORITY    125                ASSIGN INTERRUPT PRIORITY
       SPLIT       1,INT              INTERRUPT CPU
       BUFFER                         RESTART SCAN
       ASSIGN      2,V*23             TIME TO NEXT I/O INTERRUPT
       SAVEVALUE   *25+,10000,H       LOCK PAGE IN CORE
       ASSIGN      24,FN$DISK         GET DISK NUMBER
       ASSIGN      13,V$DIOT          GET DISK TIME FOR I/O
DSKIO  QUEUE       P24                QUEUE FOR DISK ARM
       LINK        P24,FIFO,GRAB3     USER CHAIN
GRAB3  SEIZE       P24                SEIZE DISK ARM
       DEPART      P24                LEAVE THE QUEUE
       QUEUE       P14                QUEUE FOR THE CHANNEL
       GATE NU     P14                WAIT FOR CHANNEL TO BECOME FREE
       DEPART      P14                LEAVE THE QUEUE
       ADVANCE     FN$SEEK            TIME FOR SEEK
       QUEUE       P14                QUEUE FOR CHANNEL
       LINK        P14,FIFO,GRAB1     USER CHAIN
GRAB1  SEIZE       P14                SEIZE THE CHANNEL
       DEPART      P14                LEAVE THE QUEUE
       ADVANCE     P13                PROCESS I/O REQUEST
       RELEASE     P24                RELEASE THE ARM
       UNLINK      P24,GRAB3,1
       RELEASE     P14                RELEASE THE CHANNEL
       UNLINK      P14,GRAB1,1
       TEST E      P26,K1,TEST        I/O OR PAGING?
       SAVEVALUE   *25-,10000,H       UNLOCK PAGE (WE WERE DOING I/O)
       UNLINK      12,FNDPG,1
       SPLIT       1,INT              INTERRUPT CPU
       PRIORITY    P12                RESET PROPER PRIORITY
       BUFFER                         RESTART SCAN
       TRANSFER    ,EXEC              BACK TO QUEUE FOR CPU
```

```
                    ****************************************
                    ****************************************
                    ***                                  ***
                    ***              PAGING              ***
                    ***                                  ***
                    ****************************************
                    ****************************************

PAGEW  ASSIGN      1,V*22              TIME TO NEXT PAGE REQUEST
       ASSIGN      26,K2               INDICATE PAGING
       ASSIGN      12,PR               RECORD CURRENT PRIORITY
       PRIORITY    125                 INTERRUPT PRIORITY
       SPLIT       1,INT               INTERRUPT CPU
       ASSIGN      19,V$PGNUM          NUMBER OF PAGE REQUESTED
       SELECTE     18,1,V$CORE,P19,XH,FNDPG      PAGE IN CORE?
       ASSIGN      25,V$COR3           POINTER
       PRIORITY    P12                 YES.  RESET PROPER PRIORITY
       BUFFER                          RESTART SCAN
       TRANSFER    ,EXEC               BACK TO QUEUE FOR CPU
*
FNDPG  TEST L      X$UIT,X$PGMUT,PGMTQ  TOO MANY USERS IN TRANSIT?
INPAG  SAVEVALUE   UIT+,K1             NO. INCREASE # IN TRANSIT BY ONE
       SELECTMIN   25,V$COR1,V$COR2,,XH FIND DESIRABLE CORE BLOCK
       ASSIGN      18,V$COR4           POINTER
       TEST GE     XH*25,K100,CKGO CAN WE GO?
       TEST L      XH*25,K1000,PROB       IN TROUBLE?
       ASSIGN      20,V$CORE           NO.  RESET FIFO BITS
REFBT  ASSIGN      21,V$INDX                    "
       SAVEVALUE   V$INDX,V$RFB,H               "
       LOOP        20,REFBT                     "
```

```
OKGO   TEST GE     XH*25,K10,TEST+2      PAGE CHANGED?
OUTPG  ASSIGN      16,K2                 YES.  MUST SWAP
       TEST GE     XH*18,K16C0,NORM      PAGE FROM DRUM OR DISK?
DISK   ASSIGN      24,FN$PGDSK           MUST BE DISK.  GET NUMBER
       SAVEVALUE   *25,K1101,H           UPDATE CORE STATUS TABLE
       SAVEVALUE   *18,P19,H             RECORD PAGE NUMBER
       ASSIGN      13,V$DPT              GET DISK PAGING TIME
       TRANSFER    ,DSKIO                UP TO DSKIO FOR PROCESSING
TEST   SPLIT       1,INT                 CREATE AN INTERRUPT
       TEST E      P16,K2,DONE           ALL DONE?
       ASSIGN      16,K1                 NO.  DECREASE COUNTER
       TRANSFER    .900,*+1,*+3          CMS DISK?
       ASSIGN      24,K3                 YES.  ASSIGN CMS DISK NUMBER
       TRANSFER    ,DISK+1               BRANCH FOR PROCESSING
       TEST LE     P7,K15,DISK           PAGE FROM DRUM?
NORM   SAVEVALUE   *25,K1101,H           SET INDICATORS IN CORE STATUS TABLE
       SAVEVALUE   *18,P19,H             RECORD PAGE NUMBER
       QUEUE       CHAN2                 QUEUE FOR CHANNEL
       LINK        18,FIFO,GRAB2
GRAB2  SEIZE       CHAN2                 SEIZE CHANNEL
       DEPART      CHAN2                 DEPART QUEUE FOR CHANNEL
       ADVANCE     12,9                  PROCESS PAGE REQUEST
       RELEASE     CHAN2                 RELEASE CHANNEL
       UNLINK      18,GRAB2,1
       TRANSFER    ,TEST                 SEE IF DONE
DONE   SAVEVALUE   UIT-,K1               DECREASE # USERS IN TRANSIT
       UNLINK      11,FNDPG,1
       PRIORITY    P12                   ADJUST PRIORITY
       BUFFER
       SAVEVALUE   *25-,K1000,H          RESET TRANSIT INDICATOR
       UNLINK      12,FNDPG,1
       TRANSFER    ,EXEC                 BACK TO QUEUE FOR CPU
```

```
                    ******************************************
                    ******************************************
                    ***                                    ***
                    ***            USER CHAINS             ***
                    ***                                    ***
                    ***          IDLER ROUTINE             ***
                    ***                                    ***
                    ***            & TABLES                ***
                    ***                                    ***
                    ******************************************
                    ******************************************

 PROR   LINK        10,FIFO
*
 PGMTQ  LINK        11,FIFO
*
*
        GENERATE    ,,,1,1,,F       GENERATE A TRANSACTION
        SAVEVALUE   PGMUT,10        INITIALIZE PGMUT
 IDLER  TEST GE     X$UIT,K1        ANY USERS IN TRANSIT?
        GATE NU     CPU             YES.  SEE IF CPU BUSY
        TRANSFER    SIM,,IDLER      SIMULTANEOUS CONDITIONS
        PREEMPT     CPU,PR          NOTHING GOING ON.  GET CPU
        ADVANCE     90000           START PROCESSING.  NEVER GET DONE
        RETURN      CPU             SHOULD NEVER GET HERE
 NDRF   TERMINATE                            "
*
*
 TOTAL  TABLE       M1,0,200,250
*
 PROCE  TABLE       P4,0,100,320
*
 XTIME  TABLE       M1,0,500,100
*
        START       12,NP
        RESET
         INITIAL    X308-X315,0
        START       1,NP
        RESET       TB1-TB4
         INITIAL    X308-X315,0
        START       1,NP
          *
          *
          *
        END
```


A N N A L Y Y SSSS I SSSS
A A NN N A A L Y Y S I S
AAAAA N N N AAAAA L Y SSS I SSS
A A N NN A A L Y S I S
A A N N A A LLLLL Y SSSS I SSSS
PPPP RRRR OOO GGG RRRR A M M
P P R R O O G G R R A A MM MM
PPPP RRRR O O G RRRR AAAAA M M M
P R R O O G GG R R A A M M
P R R OOO GGG R R A A M M
L I SSSS TTTTT I N N GGG
L I S T I NN N G G
L I SSS T I N N N G
L I S T I N NN G GG
LLLLL I SSSS T I N N GGG

```
C     ****************************************************
C     *                                                  *
C     *          RUN # 19 -  EXECUTION TIME (25T)        *
C     *                                                  *
C     ****************************************************
C
C     INITIALIZE MISCELLANEOUS VARIABLES
C
      DIMENSION A(1000),R(200),X(200),Z(200,1)
      D = 0.0
      NMBR = 0
      NCUR = 1
      NDIM = 200
      ISCALE = 0
C
C     READ NUMBER OF POINTS
C
      READ(5,100) ITIME
      T = ITIME
      L = 200
      IF(ITIME.LE.200) L = ITIME-1
C
C     INITIALIZE X ARRAY FOR PLOT
C
      DO 2 I = 1,L
    2 X(I) = I
C
C     READ IN SERIES
C
      READ (5,105) (A(I),I=1,ITIME)
C
C     CAR WILL CALCULATE MEAN AND AUTOCOVARIANCES FOR 1 TO L LAGS
C
      CALL CAR(A,ITIME,L,R,AVER)
C
C     PRINT AUTOCOVARIANCES & PLOT AUTOCOVARIANCE VS. LAGS
C
      N = (L+9)/10
      DO 15 K = 1,N
      J = K*10-10
   15 WRITE(6,125) (R(I+J),I = 1,10)
      WRITE (6,130)
      DO 17 I = 1,L
   17 Z(I,1) = R(I)
      CALL W4PLOT(X,Z,L,NDIM,NCUR,ISCALE,D,D,D,D)
      WRITE (6,115) AVER,R(1)
      WRITE (6,120)
```

```
C
CC      COMPUTE V, TAU, N, SQRT(V)/AVERAGE
C
        DO 20 J = 1,L
        N = J
        FM = J
        SUM = 0.0
        K = J-1
        DO 10 I = 1,K
        FI = I
   10   SUM = SUM + (1. - FI/FM)*R(I+1)
        V = (R(1) + 2*SUM)/T
        Z(J,1) = V
        IF(V.LE.0.0) GO TO 30
        TAU = T*V/(2.0*R(1))
        RATIO = SQRT(V)/AVER
        FN = R(1)/V
C
CC      PRINT RESULTS AND COUNT LINES
C
        WRITE (6,110) J,V,TAU,FN,RATIO
        NMBR = NMBR + 1
        IF(NMBR.LE.39) GO TO 20
        NMBR = 0
        WRITE (6,140)
   20   CONTINUE
C
CC      PLOT VARIANCE VS. LAGS
C
   30   WRITE(6,130)
        CALL W4PLOT(X,Z,N,NDIM,NCUR,ISCALE,D,D,D,D)
        STOP
C
CC      FORMATS
C
  100   FORMAT (I3)
  105   FORMAT (12F6.3)
  110   FORMAT (' ',35X,I3,4F15.6)
  115   FORMAT ('1',////////////////////,48X,'AVERAGE =',F10.6,'   R(0) =
       2',F10.6//)
  120   FORMAT (//,38X,'M',10X,'V',13X,'TAU',12X,'N',13X,'RATIO'//)
  125   FORMAT(//10F12.5)
  130   FORMAT('1')
  140   FORMAT ('1',////////////////////,37X,'M',10X,'V',13X,'TAU',12X,'N'
       2,13X,'RATIO'//)
        END
```

```
C
CCC     CAR CALCULATES AVERAGE  & L AUTOCOVARIANCES
C
C
C
C
      SUBROUTINE CAR (A,N,L,P,AVER)
      DIMENSION A(1),R(1)
C
CCC     CALC   AVERAGE
C
      AVER =0.0
      IF(N-L) 50,50,100
   50 R(1)=0.0
      RETURN
  100 DO 110 I=1,N
  110 AVER=AVER+A(I)
      FN=N
      AVER=AVER/FN
C
CCC     CALC   AUTOCOVARIANCES
C
      DO 130 J=1,L
      NJ=N-J+1
      SUM=0.0
      DO 120 I=1,NJ
      IJ=I+J-1
  120 SUM=SUM+(A(I)-AVER)*(A(IJ)-AVER)
      FNJ=NJ
  130 R(J)=SUM/FNJ
      RETURN
      END
```

LIST OF REFERENCES

1. Abate, J., Dubner, H., and Weinberg, S. B., "Queueing Analysis of the IBM 2314 Disk Storage Facility," Journal of the Association for Computing Machinery, v. 15, p. 577-589, October 1968.

2. Conway, R. W., "Some Tactical Problems in Digital Simulation," Management Science, v. 10, p. 47-61, October 1963.

3. Fishman, G. S., and Kiviat, P. J., Spectral Analysis of Time Series Generated by Simulation Models, RAND Corporation Memorandum RM-4393-PR, February 1965.

4. Fishman, G. S., and Kiviat, P. J., Problems in the Statistical Analysis of Simulation Experiments: The Comparison of Means and the Length of Sample Records, RAND Corporation Memorandum RM-4880-PR, February 1966.

5. International Business Machines Corporation Form A27-2719-0, IBM System/360 Model 67 Functional Characteristics, 1967.

6. International Business Machines Corporation Form C20-1649-2, Introduction to IBM System/360 Direct Access Storage Devices and Organization Manual, 1967.

7. International Business Machines Corporation Form H20-0304-1, General Purpose Simulation System/360 User's Manual, 1967.

8. International Business Machines Corporation Form H20-0326-2, General Purpose Simulation System/360 Introductory User's Manual, 1967.

9. International Business Machines Corporation Form 360D-05.2.005 Control Program-67/Cambridge Monitor System (CP-67/CMS).

10. Martin J., Design of Real Time Computer Systems, p. 353-460, Prentice-Hall, 1967.

11. Nielsen, N., "The Simulation of Time Sharing Systems," Communications of the ACM, v. 10, p. 397-412, July 1967.

12. Scherr, A. L., An Analysis of Time-Shared Computer Systems, The M.I.T. Press, 1967.

13. Smith, J. L., "An Analysis of Time-sharing Computer Systems Using Markov Models," Proceedings AFIPS 1966 Spring Joint Computer Conference, v. 28, p. 87-95, 1966.

14. Stimler, S., "Some Criteria for Time-sharing System Performance," Communications of the ACM, v. 12, p. 47-53, January 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chief of Naval Operations (OP-91) Department of the Navy Washington, D. C. 20350	1
4.	Asst. Professor G. Heidorn Code 55 Hd (thesis advisor) Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	15
5.	Asst. Professor W. S. Brainerd, Code 53 Bz Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
6.	Professor D. G. Williams, Code 0211 Computing Center Naval Postgraduate School Monterey, California 93940	1
7.	LT Ronald M. Goodman, USN 1646 Christine Lane West Chester, Pennsylvania 19380	1
8.	LT Leo M. Pivonka, USNR SMC, Box 2208 Naval Postgraduate School Monterey, California 93940	1

DOCUMENT CONTROL DATA - R & D

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

1. ORIGINATING ACTIVITY *(Corporate author)*
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
A Simulation Study of the Time-Sharing Computer System at the Naval Postgraduate School

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Master's Thesis, June 1969

5. AUTHOR(S) *(First name, middle initial, last name)*
Ronald M. Goodman
Leo M. Pivonka

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1969	149	14

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

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

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A GPSS model of the CP/CMS time-sharing computer system at the Naval Postgraduate School was constructed, and was used in three experiments to investigate the performance of the system under a variety of conditions. In each of the experiments the model generated auto-correlated sequences of observations which were analyzed using techniques adapted from spectral analysis.

An experiment to determine the effect of an increased number of terminals on average response time revealed that the system adequately could support a 25% increase in the number of terminals, and that the number of terminals was limited by the magnetic disk I/O capability. In a second experiment it was found that increasing the number of disks from four to eight would enable the system to support up to 20 terminals. A final experiment involving the examination of a new scheduling algorithm showed no significant changes in average response times.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer Simulation						
Time-sharing						
Spectral analysis						