THE SIMULATION OF THE BEHAVIOR

OF

A STUDENT-CREATED OPERATING SYSTEM USING GPSS

THE SIMULATION OF THE BEHAVIOR
OF
A STUDENT-CREATED OPERATING SYSTEM USING GPSS

By

GEORGE DROUCAS, B.Sc.

A Project

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

August 1976

MASTER OF SCIENCE (1976) McMaster University
(Computation) Hamilton, Ontario

TITLE: The Simulation of the Behavior of a Student-Created Operating System using GPSS

AUTHOR: George Droucas

B.Sc. (National and Capodistrial University of Athens)

SUPERVISORS: Professor R.A. Rink and Professor D.J. Kenworthy

NUMBER OF PAGES: vi, 168

ABSTRACT

While operating system concepts are taught to students in undergraduate programs in Computer Science, a student project involving the development of an operating system creates a difficult situation due to time and financial considerations. Using GPSS to simulate the behavior of a student-created operating system can reduce these problems and serve as an effective learning device. Many features and concepts can be simulated that might otherwise be ignored in a student project. An implementation of a student-created operating system is discussed. Statistics collected from the GPSS simulated model are used to indicate some measure of performance for the operating system.

ACKNOWLEDGEMENTS

I wish to record my great appreciation for Dr. R.A. Rink and Dr. D.J. Kenworthy, my supervisors, for their guidance, encouragement and valuable discussions throughout the duration of this work.

I am thankful to the Department of Applied Mathematics, McMaster University for the financial assistance made available to me during this past year.

I would like to thank my fellow graduate students, the staff and faculty of the Department of Applied Mathematics, for their help in making my stay enjoyable.

Finally, I would like to dedicate this work to my wife Maria in appreciation of the sacrifices she has made throughout my academic career.

This study was supported in part by the National Research Council of Canada, and by the Science and Engineering Board of McMaster University.

TABLE OF CONTENTS

1. INTRODUCTION

In a course on operating systems students learn principles and basic techniques. Examples and exercises help them to understand how these techniques can be applied practically. Unfortunately, lectures are not enough. For a student to appreciate some of the problems, he has to face them himself.

The main problem in giving students practical experience in operating systems is cost. They cannot be allowed to tamper with the production operating system of the university. It is not only costly in terms of money, but can degrade the service provided to other users. While a mini-computer may seem to be a solution to the above problem, it can fail to adequately serve a large number of students each of whom could be developing his own particular operating system.

One useful alternative to the above problem is to consider the use of simulation as a means for implementation of a student-created operating system. Though it removes the student from contact with hardware, simulation is an effective means for introducing and implementing the various concepts necessary for understanding the principles of an operating system.

The purpose of this project is to develop a student-created operating system in a multiprogramming environment by simulating its behaviour using GPSS (General Purpose Simulation System V/6000 [CDC 71]). GPSS has the advantage of being a high level computer language specifically designed for the simulation of discrete systems. Consequently, the student does not have to become immersed in details such as generation of random numbers, queue discipline, statistical distributions, etc. The student can concentrate on specific operating system concepts due to the power of GPSS.

This project can be used as a teaching tool in a course on operating systems. Parts of the simulated operating system can be taken and demonstrated to the students. In addition various algorithms and techniques found in the hardware and software support of an operating system can be simulated with minor modifications. For example the static partition specification scheme used in the simulated model can be changed to a dynamic partition specification scheme. The statistical results produced by the model will indicate the improvement in the efficiency of the system.

What follows is a description of the design and implementation of an student-created operating system.

1.1 Operating Systems Terminology Used in this Report

The following are definitions for terms that will be used in later sections:

A user is anybody that desires work to be done by a computer system. A job is a computation requested by a user. A job may be divided into several steps (job steps). Job steps are units of work that must be done sequentially. For example, the three steps, compile, load, execute are job steps. Once the operating system accepts a user's job, it creates several processes. John J. Donovan and Stuart E. Madnick in their book [MAD 74] define the process to be a computation that may be done concurrently with other computations. Per. B. Hansen in his paper [BRI 70] defines the process (internal process) to be the execution of one or more interruptable programs in a given storage area.

A process created by the operating system may create other processes. When a process completes its task reports back to the father process, and it is then destroyed. Figure 1-1 depicts the relationship between user, job and processes. Job, job steps and processes are represented in the GPSS program by transactions. Transactions are units which drive and are modified by GPSS block structures. A multiprogrammed (multiprogramming) operating system is one that maintains more than one user program in main storage simultaneously, sharing processor time, storage space, and other resources among the active user jobs.

In the following sections we will describe the operating system, including a description of the hardware components and software modules required to support this system.

While it is impossible to describe all the GPSS blocks required to simulate all of the various actions of the operating system, a detailed description is presented in section 4.8 showing how GPSS is used to simulate the I/O Device Manager.

It is assumed that the reader is acquainted with the various commands in GPSS. Because of this, no discussion is given on describing properties of GPSS commands.

Additional information on GPSS can be found in [CDC 71], [IBM 74].

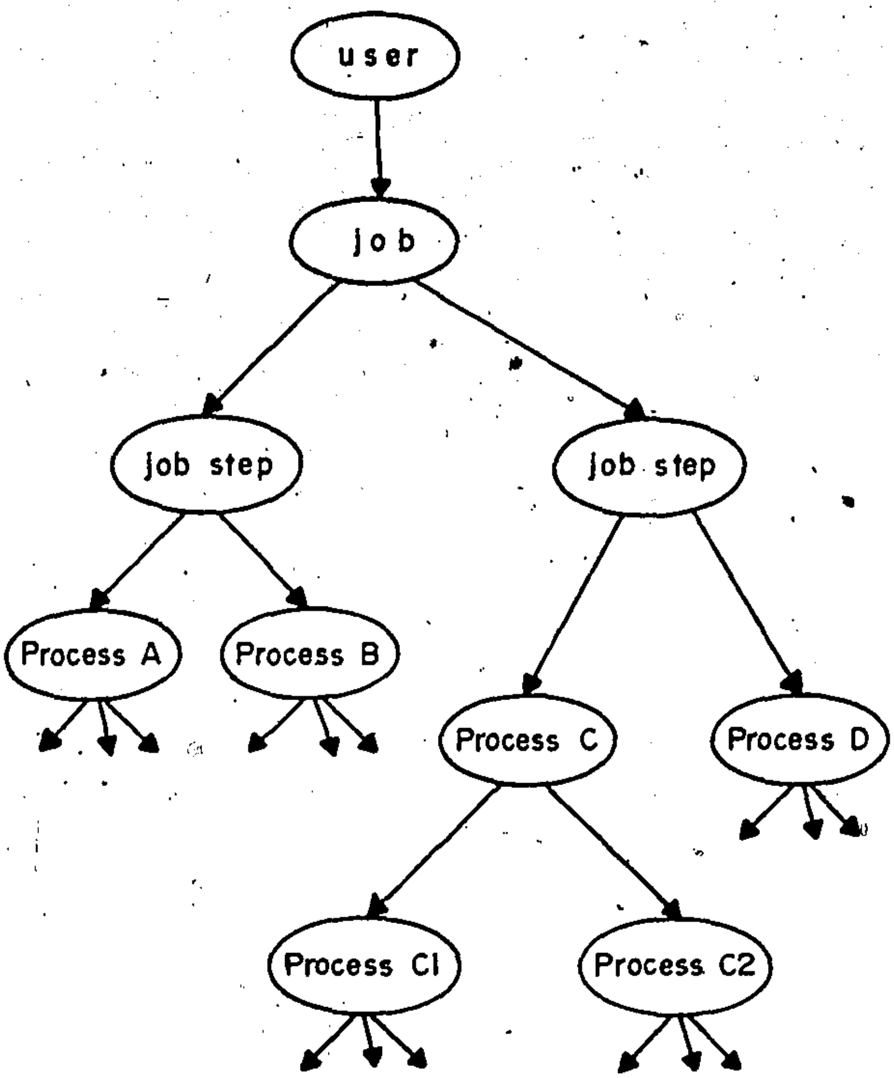


Figure 1-1. User, job, job step, process

2. DESCRIPTION OF THE OPERATING SYSTEM

The following is a brief discussion on the operating system shown in Figure 2-1 and Figure 2-2. We intend to trace a job, submitted to one of the input stations, as it moves through the operating system.

Let us assume that we have a source program written in a computer language X and that via the job cards the user has requested the system to compile, load and execute the job.

The card reader reads the source program while the channel transfers the contents of each source input card to the work area of the input spooler, where it is examined. If the card read is a job card, the input spooler enters the characteristics of the job (class, priority, ID etc.) into the system's input job queue, stored on the input drum, via the channel. If the card read is not a job card, the information contained on this card is stored on the input disk.

After the source program has been stored on the input disk, the job scheduler schedules the job for execution. A partition in the main memory is assigned to the job and a request is made to the proper compiler to compile the source program. If the requested compiler is not in the main memory, the system will access it from the input disk, where all the compilers are stored, and load the compiler into main memory. The compiler accesses the source code of the program from the input disk via the channel and compiles it. The object code (relocatable machine code), produced by the compiler, is stored on the input disk. The compiler also transfers via the channel the listing of the program onto output disk.

When the compilation of the job is over, a request is made to the loader to load the program into main memory. The loader will access the object code of the program from the input disk, produce the equivalent absolute machine code and load it into main memory. The job now is ready to be executed.

The CPU scheduler will assign the CPU to the job and the output data, produced by the job during execution, is transfered via the channel to the output disk.

When the execution of the job is over, the job enters the system's output-job queue. A line printer is assigned to the job and the output spooler accesses the information, to be printed, from the output disk and transfers it to the line printer from where it is printed.

A larger discussion on how jobs are treated by the software modules of the system is given in section 4.

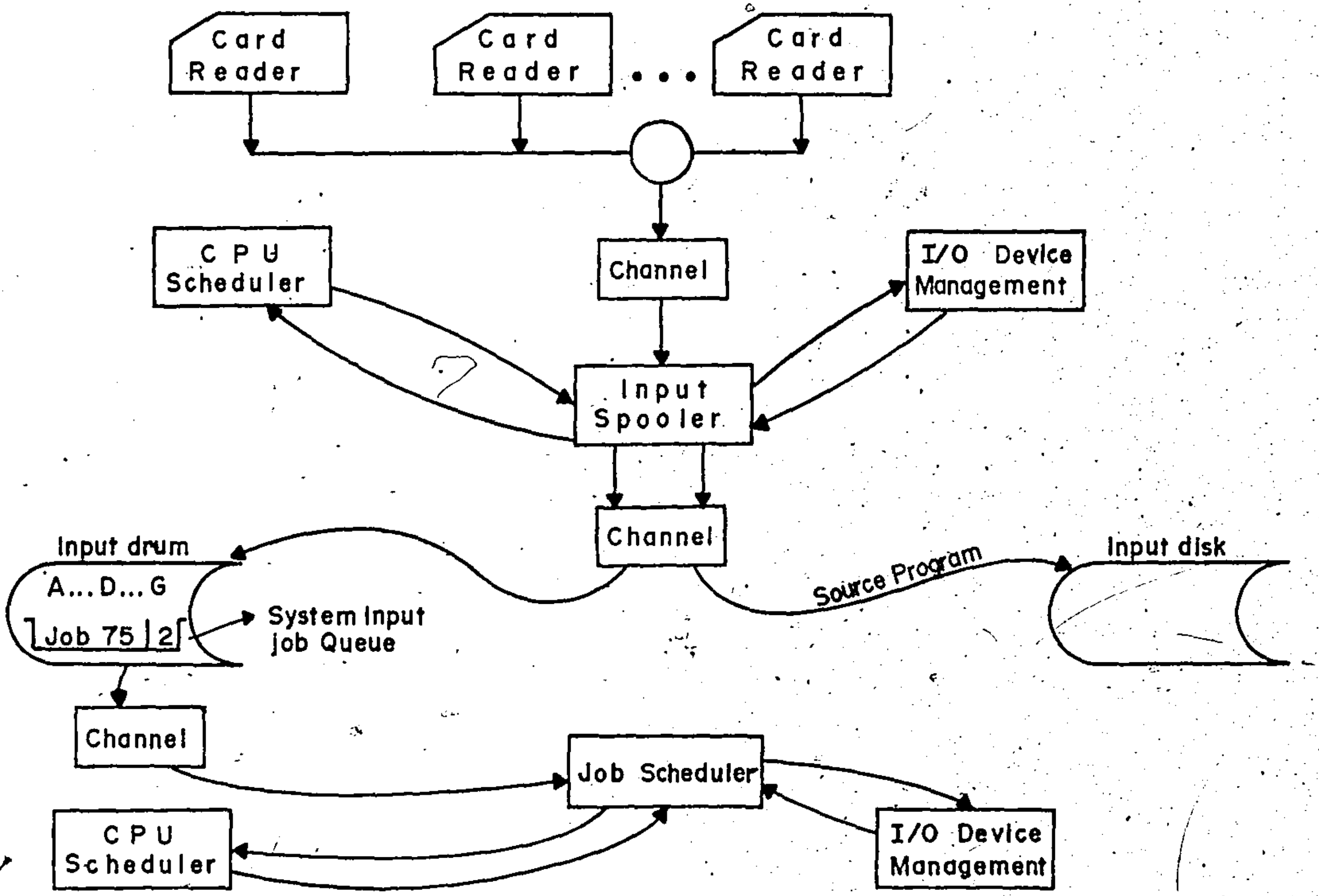


Figure 2-1. Diagram of the system (part one)

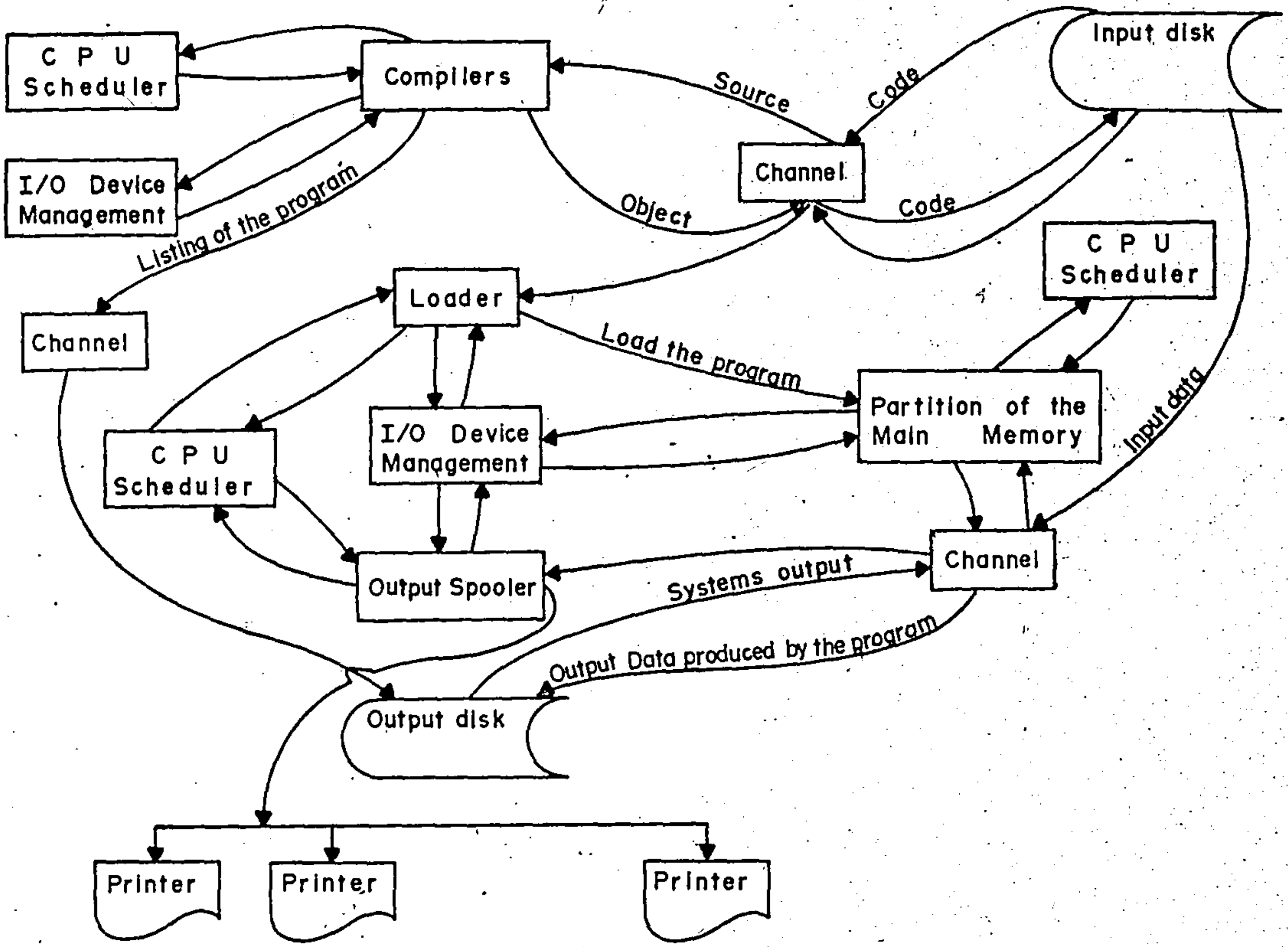


Figure 2-2. Diagram of the system (part two).

3. HARDWARE COMPONENTS OF THE SYSTEM

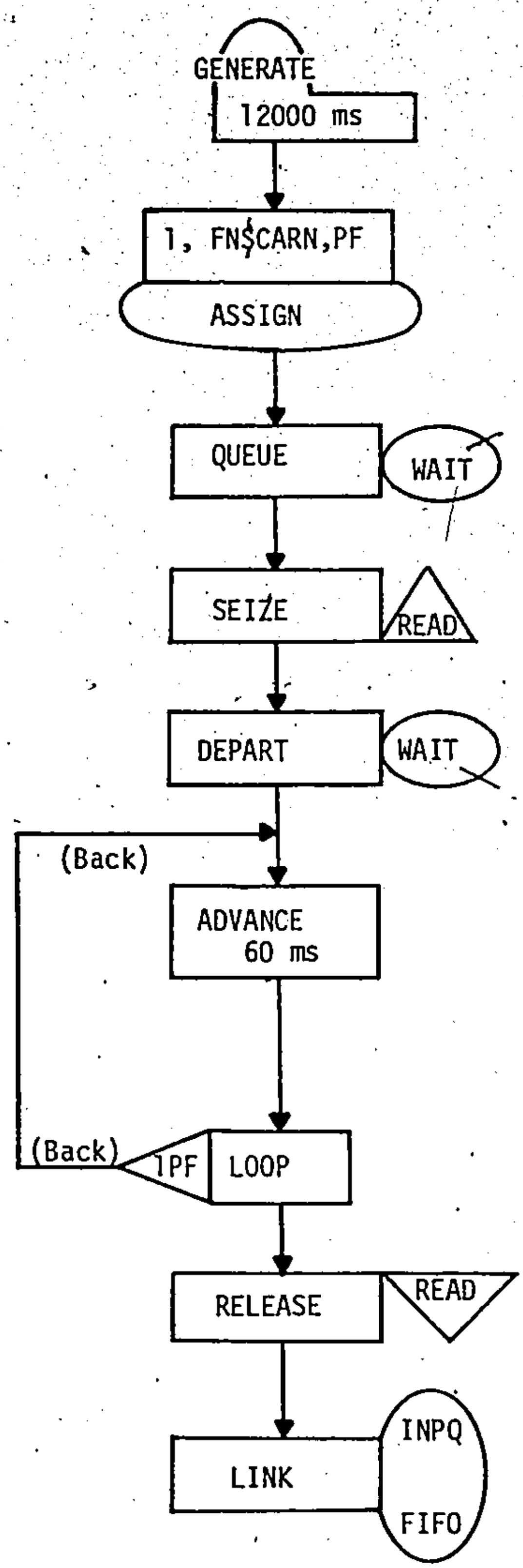
The following is a discussion on the hardware component that support the operating system. These components are composed of the card readers, line printers, CPU, channel, input drum and input and output disks.

3.1 Card Readers

As we can see from Figure 2-1 the input ports of the system are the card readers. We allow the system to have from one up to five card readers. Each card reader has a buffer which may hold the contents of one input card. The time taken by the card reader to read a card is assumed to be 60 ms [MAD 74]. A card reader will read a card and store the contents of the card in the buffer of the card reader. At this point it will signal the input spooler to examine and store the contents of the card into an appropriate auxiliary storage device. As soon as the card is stored, the input spooler signals the card reader to read the next card and the process is repeated until all the cards of a user deck are read into the system.

Figure 3-1 shows how a card reader can be simulated in GPSS. This is a simplistic model used to illustrate the power of GPSS. A more complicated model has been implemented in the student-created operating system (Appendix A, block no. 16 to block no. 37).

Each of the blocks in Figure 3-1 contains a comment that describes the effect of the block. The names of the blocks even seem



Programs arrive in the model every 12 sec.

Assign to parameter 1 the number of input cards of the program

Programs wait in line for card reader

Program gains control of card reader

Program departs from queue

Card reader reads a card

If more cards are to be read go back to read the next card

Card reader is finished with program

Enter the program in the system's input queue

Figure 3-1. GPSS model of a card reader

appropriate for the types of operations performed by a computer system. For example, the card reader is represented by a "facility" which can be under the control of only one program at a time. The SEIZE statement will enable a program to take control of the card reader only if the card reader is not under the control of another program. A program is blocked by the SEIZE block if another program is already occupying the card reader. The QUEUE/DEPART block pair is used to gather statistics of the queue that is formed in front of card reader.

By examining the standard output produced by the GPSS simulator, the utilization of the card reader and the average time programs spent in the queue can be determined.

3.2 Input Drum

The system has one magnetic drum called the "Input drum". This is a Fixed-Head drum having several hundred read/write heads. The input drum is a continuous spinning device which is used to store the system's input job queue.

Typical characterisitics of a drum [MAD,74] are:

rotation speed = 10 ms

maximum access = 10 ms

average access = 5 ms

serial access (depending on length of record) = 1 ms.

3.3 Input Disk and Output Disk

The system has two Moving-Head disks which are called the "input disk" and the "output disk". The head of a disk is physically moved from track to track. In order to identify a particular record stored on a moving-head disk, it is necessary to specify the arm position, track number and record number. The arm position is based upon radial movement whereas record number is based upon circumferential movement. Thus, to access a record, it is necessary to move to the correct radial position (if not already there) and then to wait for the correct record to rotate under the read/write head.

The purpose of the input disk is to store the source code, object code along with the input data for any program along with the five separate compilers. No more than 400 user programs, each with maximum of 1,000 cards may be stored on the input disk. If the input disk is full, the input spooler signals the card readers to stop reading any additional jobs.

The purpose of the output disk is to store the system output. System output here refers to printed output, including system messages and user job output printed by the high-speed printer. Typical characteristics of a disk [MAD 74] are:

rotation speed = 20 ms

maximum access = 75 ms

maximum arm movement = 55 ms

average access = 35 ms

serial access (depending on length of record) $\approx$ 1 ms.

3.4 Line Printers

The system may have from one up to five line printers and one auxiliary line printer. Each line printer has a buffer which may hold one output line. The time taken by the line printer to print a line is 60 ms. [MAD 74]. The auxiliary line printer handles any overflow when the system's output job queue contains a large number of entries.

No special action is needed in order to add a line printer to the system. Line printers are dynamically generated by the simulated model when they are requested from the input stations of the system.

3.5 Channels

The system has only one block multiplexor channel [MAD 74]. This channel is used to transfer information from the buffer of a card reader to main memory, from main memory to the buffer of a line printer, from main memory to a Direct Access Storage Device (DASD) and from a DASD to main memory.

Additional disks, drums, tapes and channels can be added to the system with minor modifications in the simulated model. As is shown by the statistics, produced by the GPSS simulator, the I/O devices and the channel are sufficient for this simulated model to operate efficiently.

A larger discussion on how drums, disks and channels are represented by the simulated system is presented in the next section.

4. DESCRIPTION OF THE SOFTWARE MODULES OF THE SYSTEM

This section contains a discussion on the software components of the operating system. These include the input spooler, memory management, job scheduler, compiler management and compilers, relocatable loader, CPU scheduler, I/O device management and output spooler.

4.1 Input Spooler

The input spooler or system reader is the software module of the simulated operating system whose primary objective is to improve the system's thruput. This is accomplished by overlapping job processing and system input. By system input we mean the job-input stream coming from the card readers.

The basic function of the input spooler is to move the job-input stream from the card reader to direct-access storage (input disk). The following discussion explains how this is performed. It is assumed here that the two tables shown in Figure 4-1, stored in the main memory, are used by the input spooler as a directory for storing the system input and the system output job queues onto input disk, input drum respectively. Table 1 in Figure 4-1 maintains indication of the status of each input disk storage area (usually called SPOOL areas) where input decks are stored. A given SPOOL area may be unused, in which case it is <u>available</u> for use. If it is being used to hold an input deck, it may be in either <u>input</u>, <u>hold</u>, or <u>run</u>

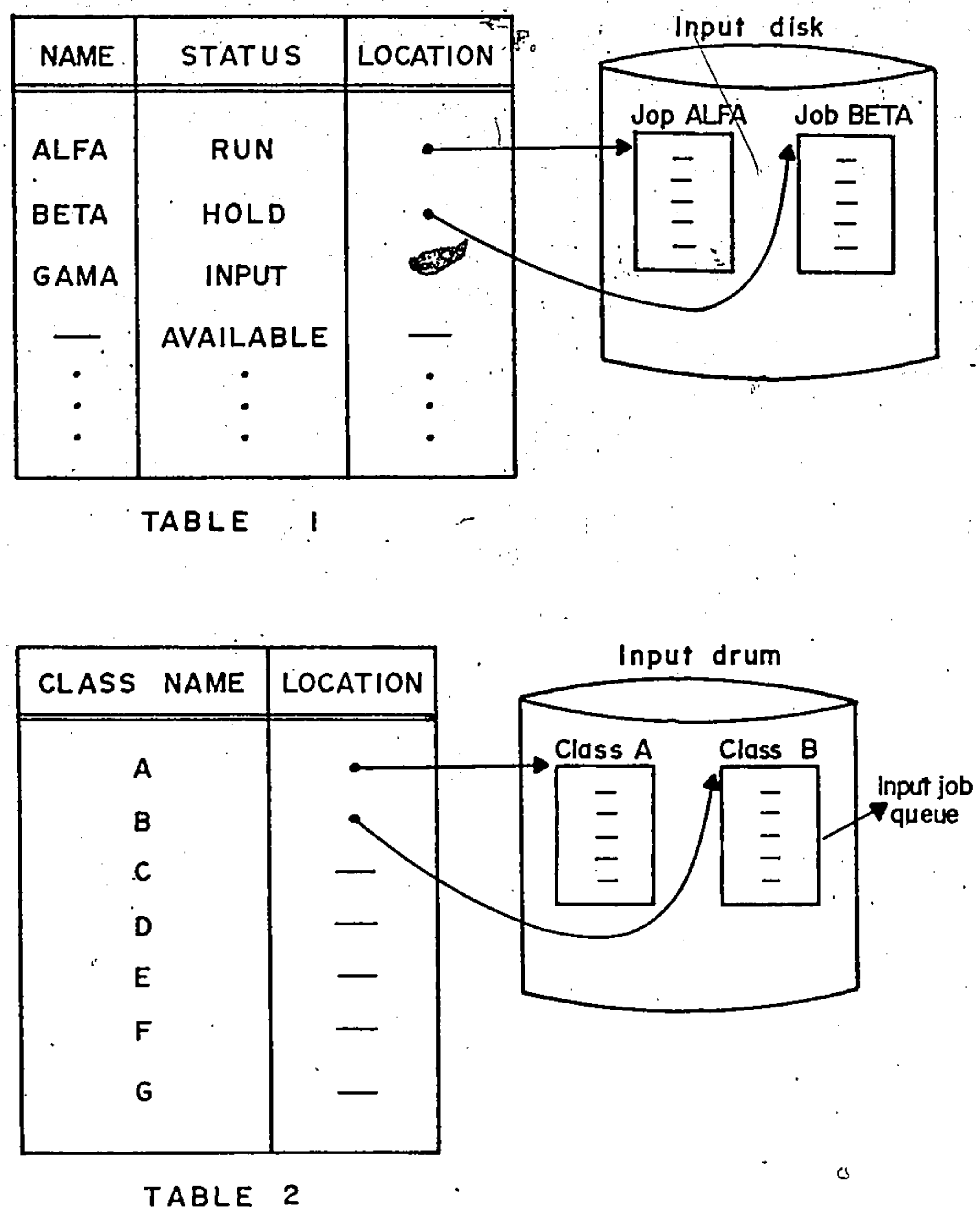


Figure 4-1. Tables used by the input spooler

status:

<u>input</u> : The input deck is still being read.

<u>hold</u> : The input deck has been completely copied onto the input disk but the corresponding job has not been started yet.

<u>run</u> : The corresponding job is currently running and is reading the input data from the SPOOL area.

In addition to the above status information, the SPOOL Table 1 also stores the user-assigned job name and the physical location of the SPOOL area on the input disk.

Table 2 stores the physical location, of the input queues for various job classes, on the input disk.

The input spooler has one buffer in main memory for each card reader. Each of those buffers is capable of holding the contents of 10 input cards.

The input spooler is activated by an I/O complete indication from the card reader. This I/O complete indication is issued whenever a card has been read. The input spooler transfers the contents of the card reader buffer, via channel, to the main memory. If the card is a special job card used to start each input deck, the previous input SPOOL area is known to be complete and is placed in "hold" status. An "available" entry in table 1 (Figure 4-1) and a free SPOOL area on the input disk must be found for the new input deck. The entry is then set to "input", and information, such as job name, is transfered from the job card to the Table 1 entry. Also the input spooler enters the new job into the system input job queue, stored on the input drum. While the input spooler performs the above tasks the card reader reads the next input card of the job.

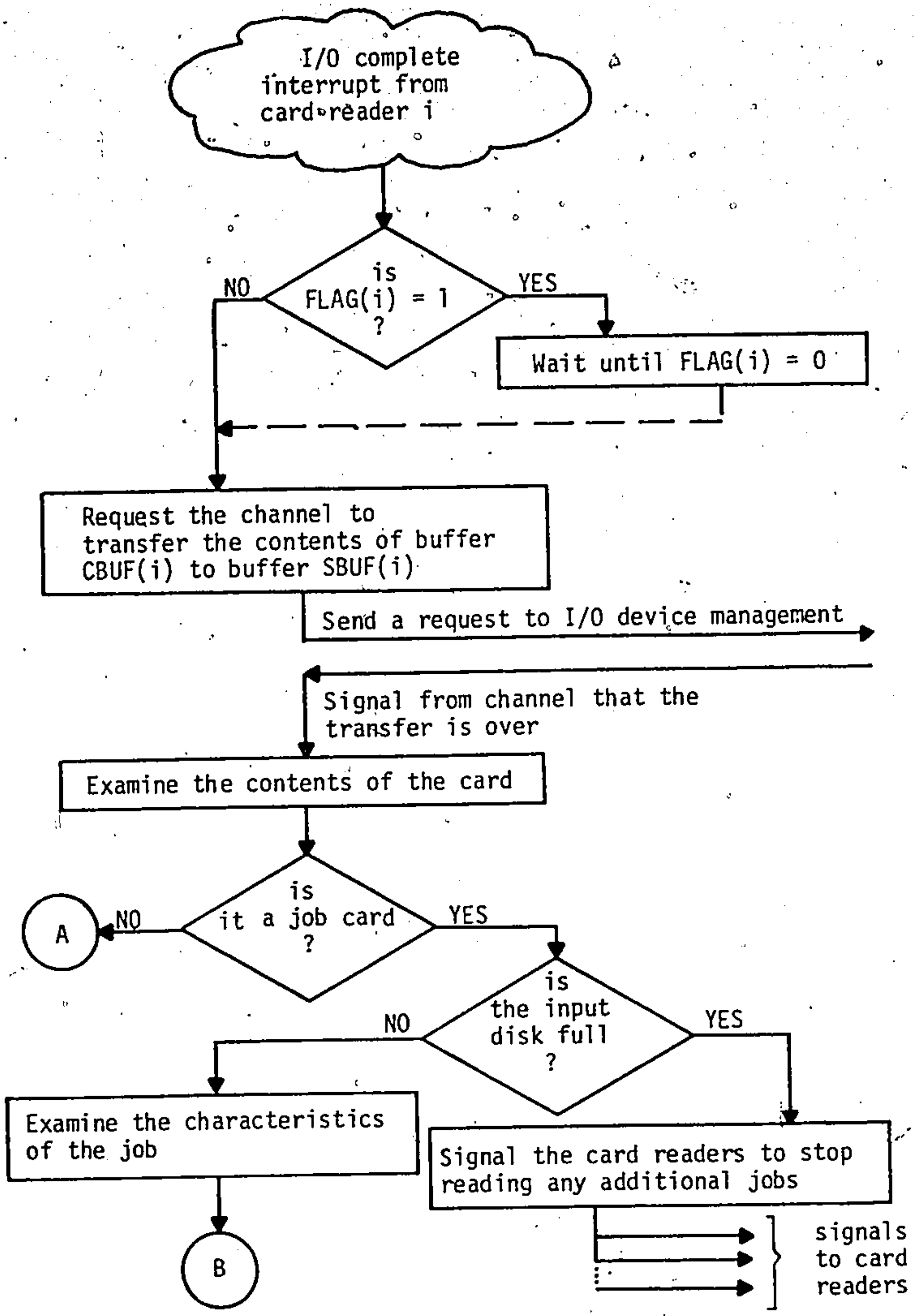
If the card read by the card reader is not a job card, the input spooler stores it into the spooler buffer that corresponds to this card reader. When the buffer is full the input spooler transfers, via channel, the contents of the buffer to the SPOOL area, of the input disk, which was assigned to the job earlier. The following algorithm gives a detailed description of the input spooler used in the simulated model.

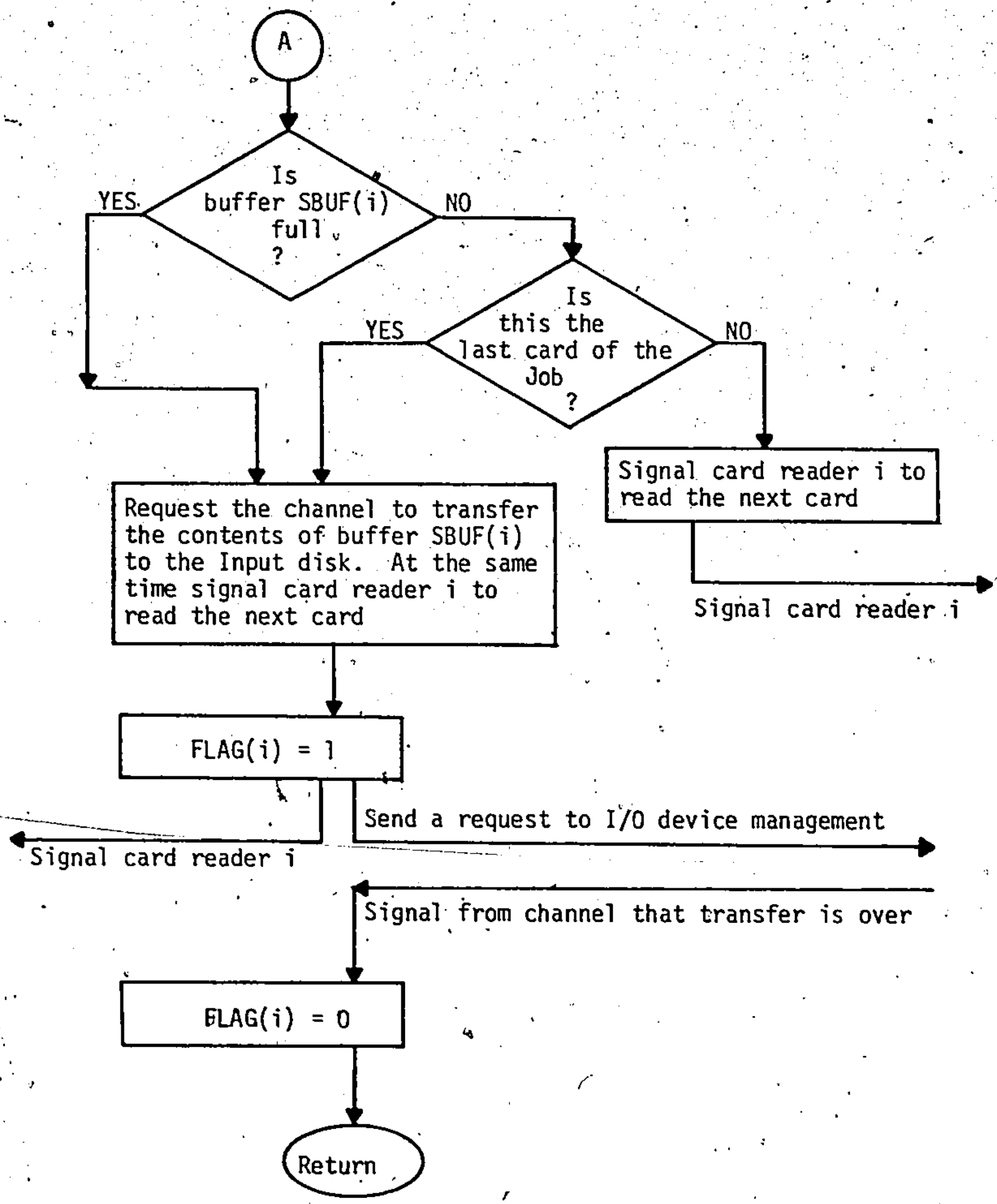
4.1.1 Algorithm of the Input Spooler

Initially FLAG(i) = 0, and i = 1, 2, ..., N where, N is the number of card readers in the system.

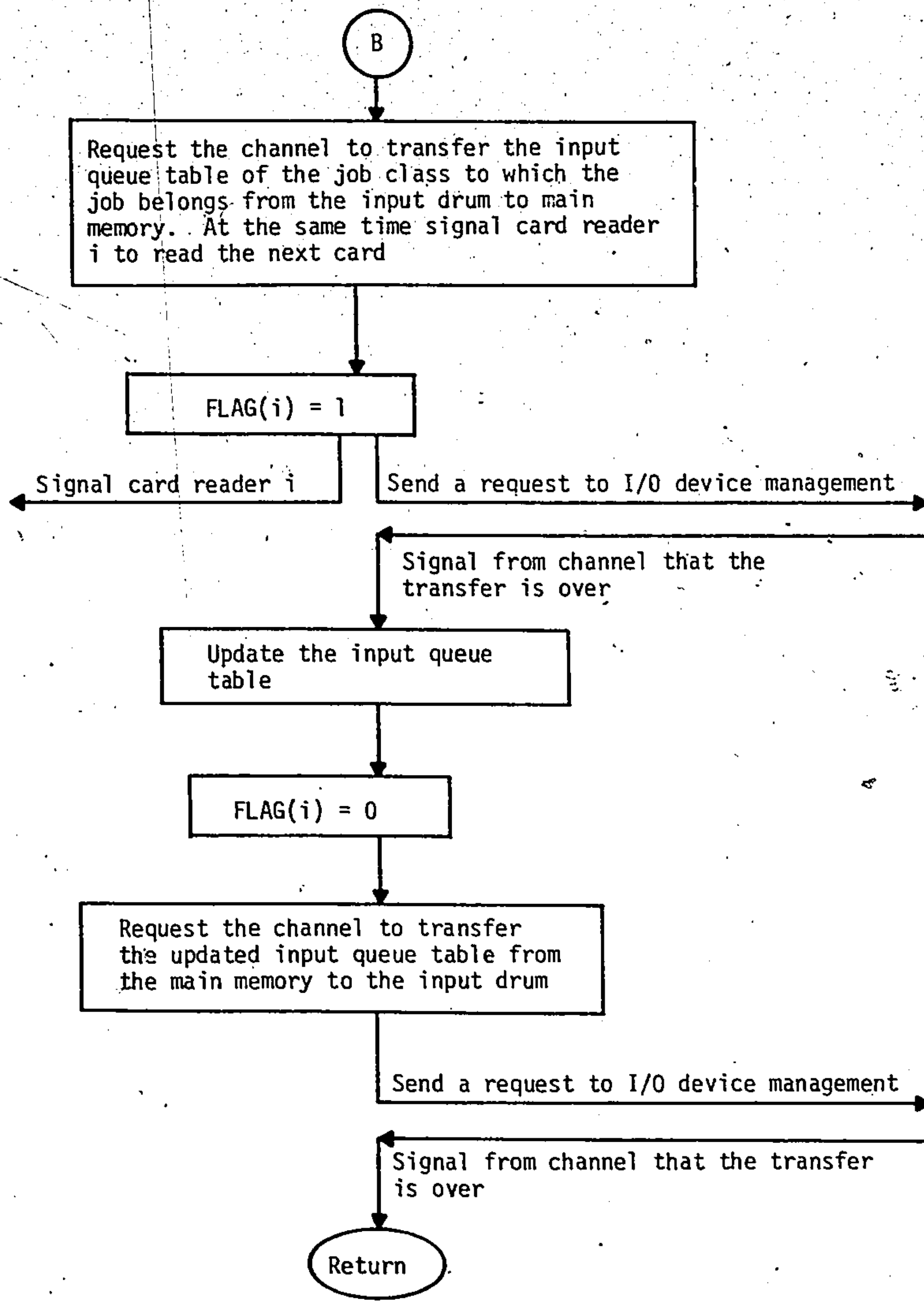
CBUF(i) is the buffer of card reader i

SBUF(i) is the buffer of the input spooler that corresponds to card reader i.


I/O complete interrupt from card reader i
is FLAG(i) = 1 ?
NO
YES
Wait until FLAG(i) = 0
Request the channel to transfer the contents of buffer CBUF(i) to buffer SBUF(i)
Send a request to I/O device management
Signal from channel that the transfer is over
Examine the contents of the card
is it a job card ?
NO
A
YES
is the input disk full ?
NO
YES
Examine the characteristics of the job
B
Signal the card readers to stop reading any additional jobs
signals to card readers



A
Is buffer SBUF(i) full ?
YES
NO
Is this the last card of the Job ?
YES
NO
Signal card reader i to read the next card
Signal card reader i
Request the channel to transfer the contents of buffer SBUF(i) to the Input disk. At the same time signal card reader i to read the next card
FLAG(i) = 1
Signal card reader i
Send a request to I/O device management
Signal from channel that transfer is over
FLAG(i) = 0
Return


B
Request the channel to transfer the input queue table of the job class to which the job belongs from the input drum to main memory. At the same time signal card reader i to read the next card
FLAG(i) = 1
Signal card reader i
Send a request to I/O device management
Signal from channel that the transfer is over
Update the input queue table
FLAG(i) = 0
Request the channel to transfer the updated input queue table from the main memory to the input drum
Send a request to I/O device management
Signal from channel that the transfer is over
Return

4.2 Memory Management

In the simulated operating system we have used the static partition specification version of the partition allocation technique [MAD 74]. By static specification, we mean that memory is divided into partitions prior to the processing of any jobs. This is similar to the technique used in IBM's OS/360 MFT (Multiprogramming with a Fixed number of Tasks [HEL 75]). The partition specification may be designated by the "computer operator". Computer operator here refers to the person who is using the simulated model.

In the main memory of the simulated model there are two types of partitions, the system partition and the problem-program partitions (PP partitions). The system partition is assigned to the operating system which is permanently resident in main memory. Each of the PP partitions holds a separate user job, as illustrated in Figure 4-2. The partition sizes have to be chosen to correspond closely to the more common job sizes. This is important since it effects the utilization of the main memory. A poor choice in the size of the partition may result in increasing the problem of fragmentation. A typical example is shown in Figure 4-3. In this example, all the partitions are assigned in the best possible way. Yet, only 209K bytes of the available 800K bytes of storage is actually used. Thus, over 73 percent of the available memory is wasted. While this may be an extreme case, similar situations can occur in actual systems.

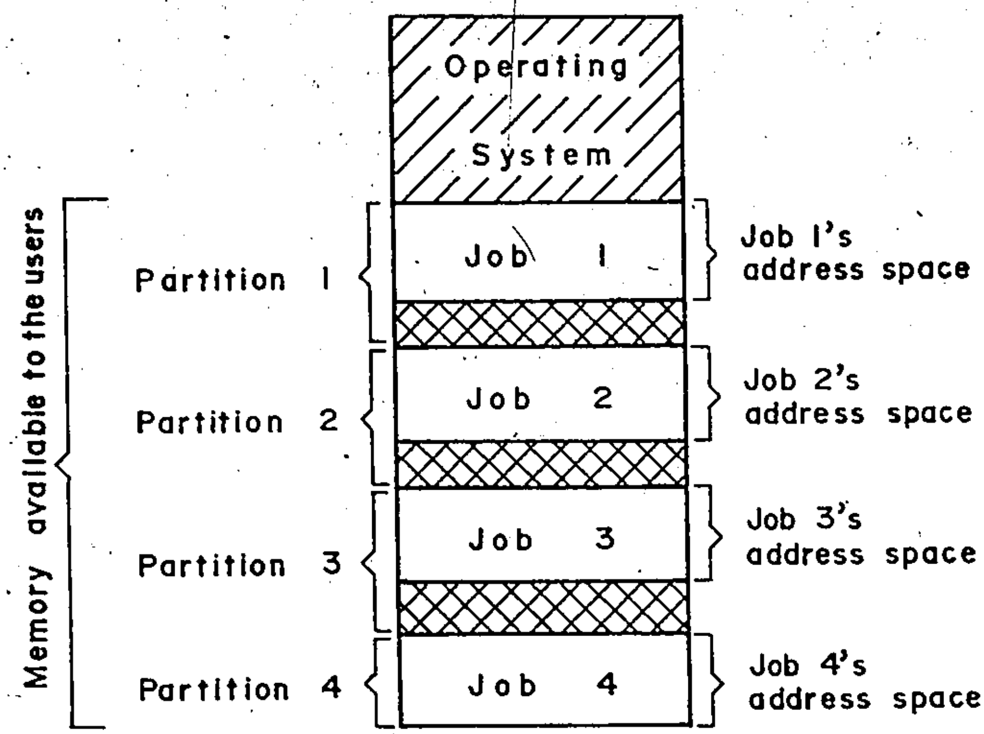


Figure 4-2. Partitioned allocation

Partition	Partition Size	Job Size	Wasted Space
1	8K	1K	7K
2	32K	9K	23K
3	120K	35K	85K
4	120K	39K	81K
5	520K	125K	395K
	800K	209K	591K

Figure 4-3. Fragmentation problem

In the simulated model, the memory available to the users is initialized by the GPSS statement INITIAL XF20, M, where M is an integer number expressed in K bytes. For example, INITIAL XF20, 180 assigns 180 K bytes of main memory for the user programs. Jobs that request more memory than 180 K bytes will be rejected by the system. The system may support from one up to seven PP partitions. The number of PP partitions in the main memory can be designated by the GPSS statement INITIAL XF8, N, where N is any integer number from one up to and including seven. The PP partition sizes and the job classes that each PP partition may serve are designated in the GPSS model by the fullword Matrix 1 (Appendix A, block no. 1.a). An example is given in Figure 4-4. The PP partitions are identified by the rows 1, 2, 3, 4, 5, 6, 7 of the matrix 1. The row numbers determine the job scheduling and CPU scheduling priority (as will be discussed later). The higher the number the lower will be the priority. Thus, PP partitions with high priority are assigned to small row numbers where each PP partition may serve from one up to four classes. The first four elements of each row are the classes that each partition may serve while the fifth element is the size of the partition. In the example in Figure 4-4 the main memory contains three PP partitions. Partition no. 1 may serve jobs that belong to classes B and A, while its size is 30K bytes. The size of the partition no. 2 is 60K bytes and may serve jobs that belong to classes A, B, E and G. The third partition may serve jobs that belong to class D and its size is 20K bytes.

ROW \ COLUMN	1	2	3	4	5
1	B	A	0	0	30
2	A	B	E	G	60
3	D	0	0	0	20
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0

Figure 4-4. Matrix 1

In the above discussion we used the term job class but we did not say anything about it. In the following section we will discuss the job classification scheme employed by the simulated system.

4.2.1 Job Classification

As was discussed earlier, the computer operator assigns job classes to each of the PP partitions, while users assign a class to their job on a job card. In the simulated system there are five classes, A - G, used to classify user jobs into I/O-oriented jobs, CPU-oriented job, jobs with small and jobs with large memory requirements. [HEL 75]. Figure 4-5 shows the seven classes, A - G, along with the CPU, memory and I/O categories M, L, S, L1, etc. The following is a discussion on these categories.

Memory categories S, M, L

Category S: Jobs that belong in this category require M_1K to M_2K bytes of main memory, where $M_1 < M_2$. Numbers in the range from M_1 to M_2 are given by function 4 (Appendix A, block no. 4.e).

Category M: Jobs that belong in this category require M_2K to M_3K bytes of main memory, where $M_2 < M_3$. Numbers in the range from M_2 to M_3 are given by function 5 (Appendix A, block no. 5.e).

Category L: Jobs that belong in this category require M_3K to M_4K bytes of main memory. Jobs that require more than M_4K bytes of main memory are considered non-standard jobs and are scheduled by the simulated system at the end of the simulated period. Numbers in the range from M_3 to M_4 are given by function 6 (Appendix A, block no. 6.e). M_4 is initialized by the statement INITIAL XF4, M_4.

The CPU and I/O categories are similarly defined and are summarized in Figure 4-6.

T_1 to T_5 are integer numbers expressed in milliseconds. For example, T_1 = 100 ms, T_2 = 1000 ms, etc.

$\emptyset_1$ to $\emptyset_{12}$ are integers which represent the number of output lines of a user job, without including the job listing and system messages. For example $\emptyset_5$ = 40 output lines, $\emptyset_7$ = 100 output lines etc.

I_1 to I_4 are integers which represent the input data cards as the percentage of the input cards of a user job. For example I_3 = 50%, I_4 = 90%.

CLASS	CPU	MEMORY	I/O
A	S1	S	S1
B	M	S	L2
C	M	S	M
	M	M	M
D	S2	M	L1
E	M	M	L2
F	L	L	L3
G	L	L	S2
	L	M	S2

Figure 4-5. Job classification

	S	M	L
CPU	$\underline{S1}$: $T_1 < T_2$ $\underline{S2}$: $T_2 < T_3$	$T_3 < T_4$	$T_4 < T_5$
I/O	S1: $Ø_1 < Ø_2$ S2: $Ø_3 < Ø_4$ $(Ø_1 < Ø_3 < Ø_2 < Ø_4)$ $\underline{S1}$: $I_1 < I_2$	$Ø_5 < Ø_6$ $(Ø_3 < Ø_5 < Ø_2 < Ø_4 < Ø_6)$ $I_2 < I_3$	$\underline{L1}$: $Ø_7 < Ø_8$ $\underline{L2}$: $Ø_9 < Ø_{10}$ $\underline{L3}$: $Ø_{11} < Ø_{12}$ $(Ø_4 < Ø_7 < Ø_6 < Ø_8)$ $(Ø_6 < Ø_9 < Ø_8 < Ø_{10})$ $(Ø_8 < Ø_{11} < Ø_{10} < Ø_{12})$ $I_3 < I_4$
MEMORY	$M1 < M2$	$M2 < M3$	$M3 < M4$

Figure 4-6. Job categories

4.3 Job Scheduling

In the simulated system there are two job schedulers. The first is used for scheduling regular jobs while the second for scheduling non-standard jobs.

4.3.1 Job Scheduler for Regular Jobs

Job scheduling for regular jobs consists of assigning jobs to problem-program partitions (PP partitions). Once a job is assigned to a PP partition it remains there until the execution of the job is over. Since no more than seven PP partitions can be specified in the system, seven is the maximum number of user jobs that can be multiprogrammed.

The job scheduler selects the "best" candidate from the system-input job queue. What scheduling rule does the job scheduler for regular jobs use to select the next job from the system-input job queue? In Figure 4-7 we give a specific example to show how the best job is selected. In Figure 4-7 there are four PP partitions, P1 through P4, and one partition P0 for the operating system.

Assume that P1, P3 and P4 have been scheduled with say Job 9, Job 7 and Job 8 and that P2 is now ready for another job. The job scheduler now has to schedule a job for this partition. Job scheduling is performed using the job-classification scheduling scheme. Job classes, represented by letters A thru G, are assigned to both PP partitions at system generation time by the operator, and to user jobs by the users. The job scheduler will schedule the "best"

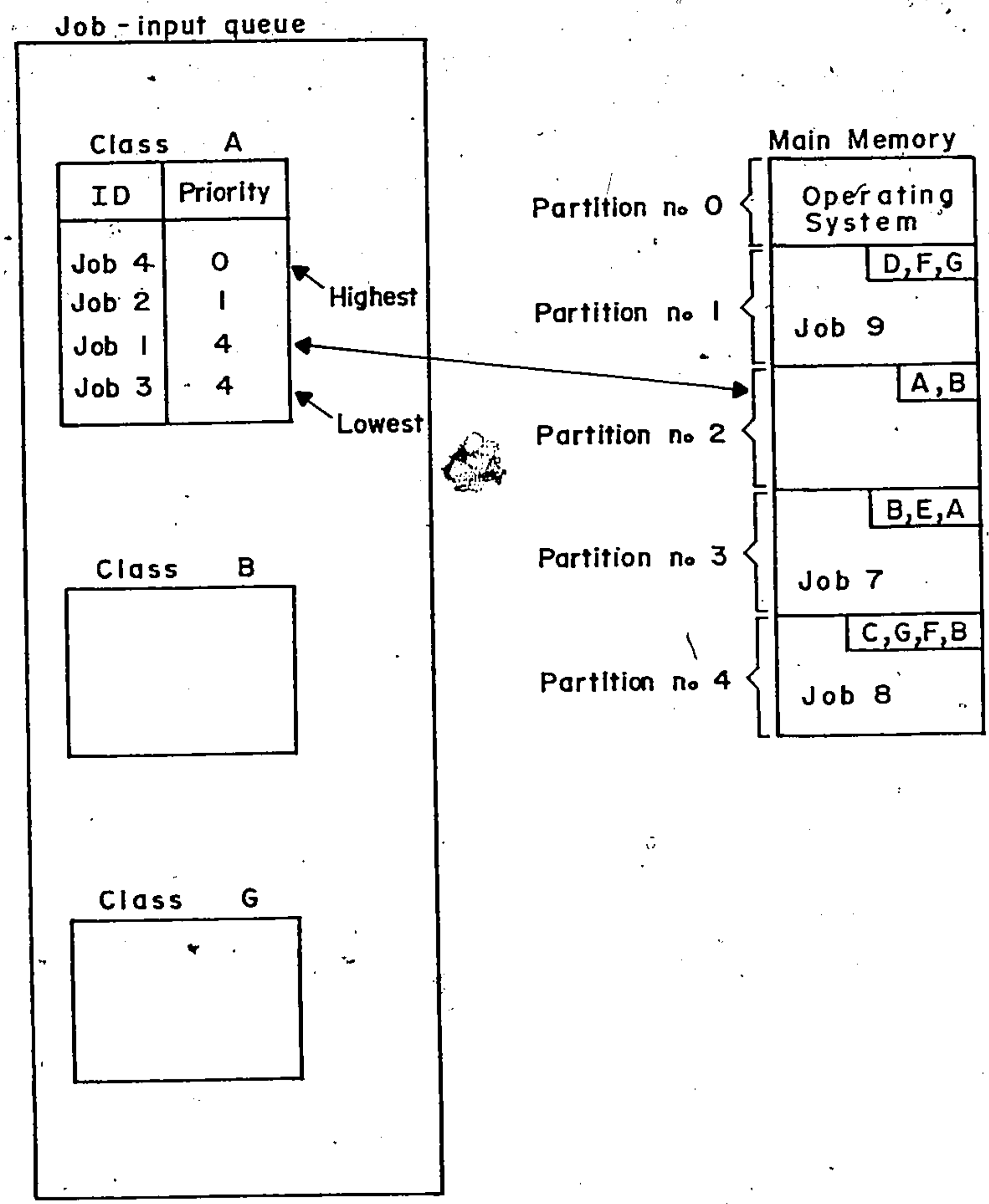


Figure 4-7. Job class and priority scheduling

candidate of all jobs in the input-job queue that have been assigned a class value that has also been assigned to a partition. Let us now examine in more detail the scheduling process.

The job's class is punched by the user on the job card with the CLASS = parameter (A, B, C, D etc.). The assignment of job classes to jobs can be considered to divide the input queue into several subqueues, one for each class. Since several jobs may have the same class, a tie-breaking scheme is needed. The first tie-breaking provision permits users also to supply, on the job card, a priority value ranging from 4 (lowest priority) through 0 (highest priority). If, using the priority value, the job scheduler still does not have a unique job, the remaining ties are broken by the order in which the jobs have entered each queue (i.e., FCFS within priority and class). From one to four of the seven possible A - G job classes may be assigned to each PP partition by the operator. Jobs for a partition may be scheduled only from queues whose class (es) corresponds to the class (es) assigned to the partition. The first job class assigned to a partition is given highest priority for job scheduling in that partition; if there is a second, it would be second highest, a third it would be third highest and a fourth would be lowest. Figure 4-7 shows a particular scheduling problem with job-class queues and classes assigned to partitions. Partition classes are indicated in the upper right-hand corner of each partition; class priority is indicated from left to right. From this example, several characterisitcs may be observed about class scheduling. The job scheduler may schedule jobs for partition 2 from either job-class queue A or job-class queue B. All jobs in queue A will be scheduled first, since A is the first entry in P2's assigned class list and therefore has a higher priority for P2 than queue B. Assuming that queue A has several jobs, the job scheduler then will schedule the job with the highest priority from queue A. Jobs in queues D, F and G may only be scheduled in P1. Notice also that the same job class may be assigned to different partitions. For example, P2 and P3 may both schedule class-A jobs, although in P2, class-A jobs have a higher scheduling priority. The major problem with job-classification scheduling is internal fragmentation of the main memory. An example of internal fragmentation was given earlier. In that example 74 percent of the main memory was fragmented. This "fitting" problem must be balanced with the objective of assigning I/O-bound jobs to high-priority partitions and CPU-bound jobs to lower-priority partitions, thus achieving a good overall use of system resources. This latter consideration will be discussed further in section 4.7.

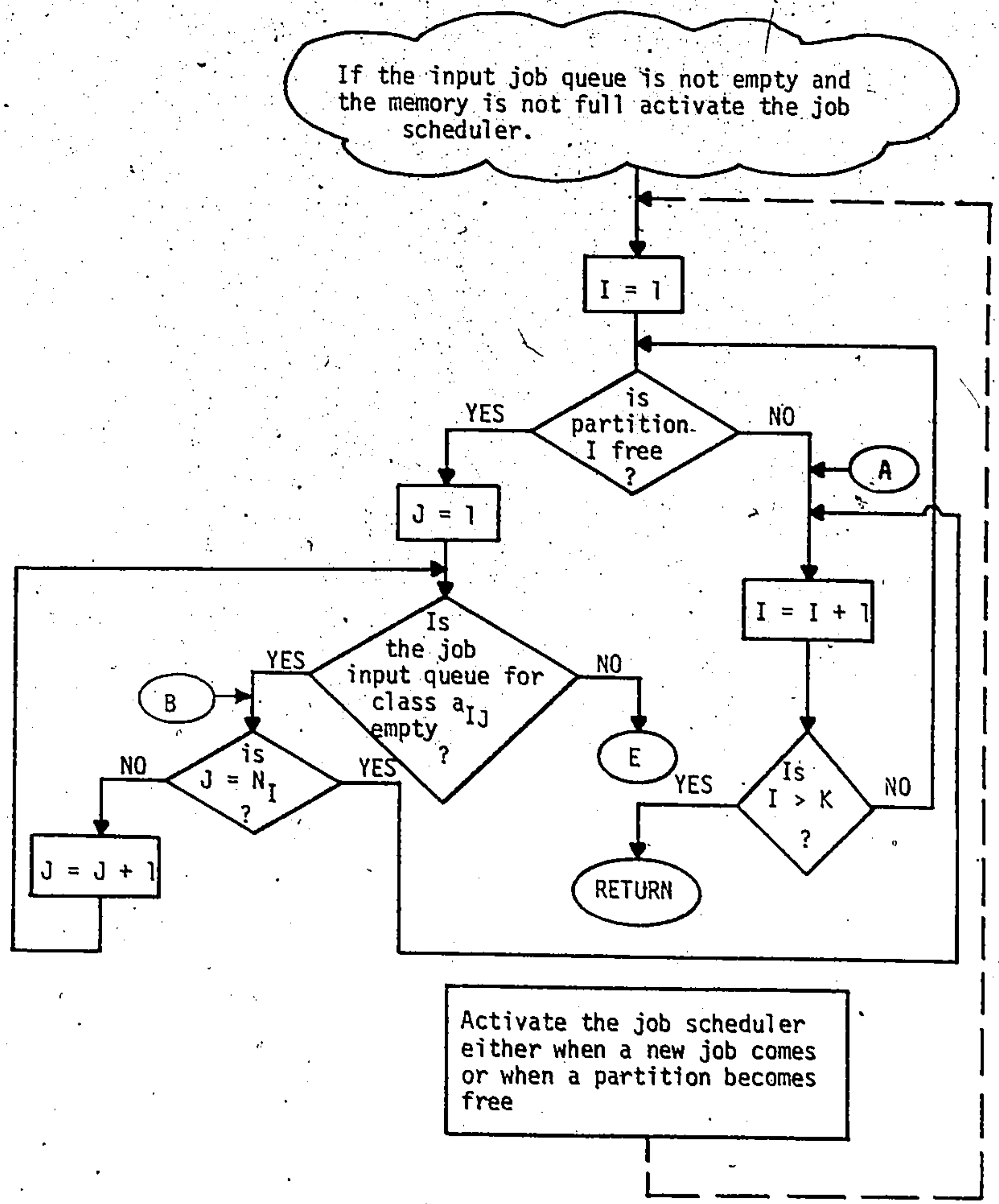
In the GPSS model of the student-created operating system (Appendix A) the job scheduler is activated by one transaction generated from GENERATE block 200. This transaction is brought into the system at the beginning of the simulation. It activates the job scheduler only when the input queue of the system is not empty and when there is at least one partition free in the main memory.

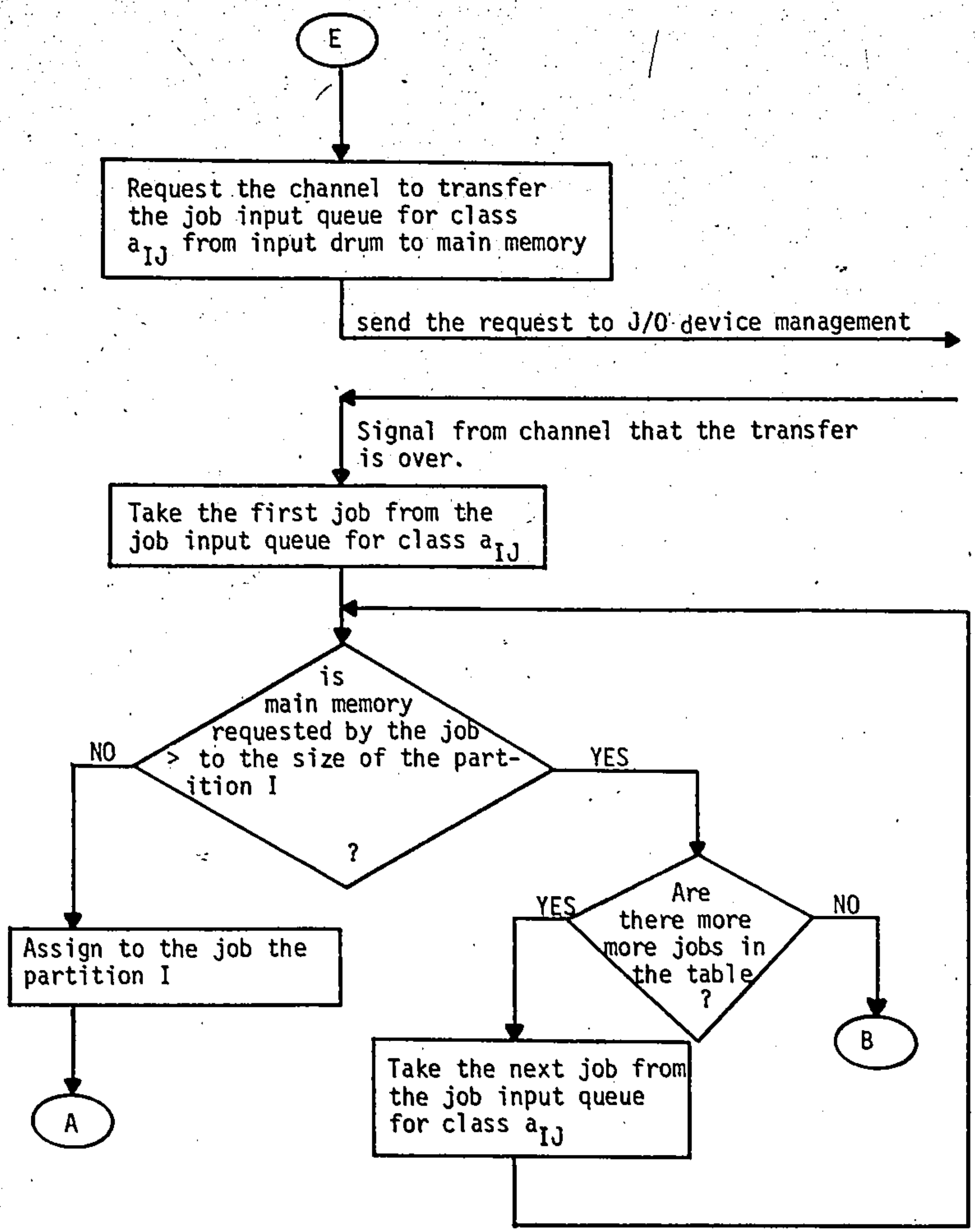
Assume that the above two conditions are satisfied. The job scheduler examines the partitions of the main memory starting from those with the highest priority, until it finds one that is free. Assume that this is partition no. 2 which may serve classes A or B. The job scheduler examines the table 2 of the input spooler to see if the input queue for class A is empty. If it is not empty the job scheduler selects the job with the highest priority from queue A and assigns partition 2 to the job. If the input queue for class A is empty, the input queue for class B is examined and the above process is repeated. If both input queues are empty and no other partition is free, the job scheduler will be activated again, either when a new job arrives at one of the input queues A or B, or when a new partition becomes free.

The following algorithm gives a detailed description of the job scheduler used in the simulated model.

4.3.1.1 Algorithm of the Job Scheduler for Regular Jobs

Assume that the class(es) $a_{I1}, a_{I2}, \ldots, a_{IN_I}$ are assigned to partition I, where $I = 1, 2, 3, \ldots, K$, $N_I = 1, 2, 3, 4$ and K is the number of partitions in the main memory.


If the input job queue is not empty and the memory is not full activate the job scheduler.
I = 1
is partition I free ?
YES
NO
A
J = 1
I = I + 1
Is the job input queue for class a$_{IJ}$ empty ?
YES
NO
B
E
is J = N$_I$?
NO
YES
Is I > K ?
YES
NO
J = J + 1
RETURN
Activate the job scheduler either when a new job comes or when a partition becomes free


E
Request the channel to transfer the job input queue for class a_{IJ} from input drum to main memory
send the request to J/O device management
Signal from channel that the transfer is over.
Take the first job from the job input queue for class a_{IJ}
is main memory requested by the job > to the size of the partition I ?
NO
YES
Assign to the job the partition I
A
Are there more more jobs in the table ?
YES
NO
B
Take the next job from the job input queue for class a_{IJ}

4.3.2 Job Scheduler for Non-Standard Jobs

As was mentioned earlier non-standard jobs are scheduled at the end of the day. The timer transaction, which simulates the end of the day event, arrives at GENERATE block 441, when the simulated period is over. This transaction takes the following steps.

1) It closes the gates at the input stations (LOGIC S 22), so no further jobs can enter into the system.

2) It waits until all the jobs that are already in the system are completed.

3) It activates the job scheduler for non-standard jobs. The characteristics of the non-standard jobs are stored by the input spooler in the job input queue WAIT1.

The rules used by the input spooler for entering non-standard jobs into queue WAIT1 are the same as those used for regular jobs (FCFS with priority). Before non-standard jobs are scheduled the main memory is divided into two contiguous regions. A portion of memory is permanently allocated to the operating system. All the remainder of memory is available to the single job being processed [MAD 74].

Job scheduling is based on FCFS with a prioirity scheme. The job scheduler takes the first job from the input queue WAIT1 and allocates it to the portion of the memory available for the users. When the job is completed, the next job is scheduled and the same process is repeated until all the non-standard jobs are served by the system.

4.4 Compiler Management and Compilers

4.4.1 Compilers

There are five compilers available in the simulated system. All of these have been chosen to be in relocatable machine code and stored on the input disk.

The user will request a compiler through a control card by specifying the code of the compiler (COMP = code). These codes are 17, 18, 19, 20, 21 for compilers 1, 2, 3, 4, 5 respectively. Each compiler is associated with a queue, where processes wait when the requested compiler is busy. The processes are stored in these queues according to FCFS with a priority scheme.

Only two compilers are allowed in main memory at any one time. Each of the compilers in main memory has two buffers labelled buffer no. 1 and buffer no. 2. The compiler via the channel transfers a block of source code from the input disk into buffer no. 1. This source code is then compiled and relocatable machine code is produced which is then stored into buffer no. 2. When buffer no. 2 is full, its contents are transfered via the channel onto the input disk. This process is repeated until the entire source program is compiled.

In the GPSS simulated model, the compilers are implemented by segment block no. 293 to block no. 323 (Appendix A).

4.4.2 Compiler Management

The basic functions of the compiler manager are:

1) to keep track of the status of the compilers which are in main memory,

2) to decide on a policy that will determine which of the compilers will be in main memory, for how long, and when,

3) to allocate a compiler to a job for as long as the job requires it,

4) to determine a deallocation policy. When the compilation of a job is over the compiler is released.

For each one of the five compilers the compiler manager maintains the following information.

i) Residence

Memory: a copy of the compiler is in the main memory.

Disk : the compiler is on the disk.

Load : the compiler is being loaded in the main memory.

ii) Status

Busy : the compiler is now compiling a program.

Free : the status of a compiler is free if a) the compiler is not in the main memory, b) the compiler is being loaded in the main memory and c) the compiler is in the main memory but is not compiling any program at this time.

The following is an example which illustrates how the compiler manager functions. Let us assume that the situation is as stated in Figure 4-8.

compiler no.	residence	status
1	Memory	Busy
2	Disk	Free
3	Load	Free
4	Disk	Free
5	Disk	Free

Figure 4-8. Compilers residence, status information

Compiler 1 is in the main memory and is compiling a program, compilers 2, 4 and 5 are on the input disk and compiler 3 is being loaded into memory. Also assume that the queue for compiler 1 is empty and that process no. 9 has requested compiler 4. The residence status (disk) for compiler 4 indicates that the compiler is not in the memory. Since there is no location of memory free (only two compilers may be put into memory at the same time) to load the compiler 4 the compiler manager enters process no. 9 into a queue called SPES. In this queue processes are waiting, according to FCFS within a priority scheme, if the compiler which have been requested cannot be loaded into main memory. Assume that after some time t the situation is as stated in Figure 4-9 and that there are no processes waiting for compiler 1.

compiler no.	residence	status
1	Memory	Free
2	Disk	Free
3	Memory	Busy
4	Disk	Free
5	Disk	Free

Figure 4-9. Compilers residence, status information (at time t).

The compiler manager checks the queue SPES and finds that process no. 9 has requested compiler 4 and is waiting. Since compiler 4 is not in main memory and compiler 1 is free, compiler 4 can be loaded into the location of the main memory where the compiler 1 is located. The compiler manager, before instructing the loader to load the compiler 4 into main memory, takes the following actions:

a) It changes the residence status of the compiler 1 from "Memory" to "Disk" since the copy of compiler 1, which is now in the main memory, is going to be overwritten by compiler 4.

b) It changes the residence status of compiler 4 from "Disk" to "Load". The residence status "Load" is necessary to be used when a compiler is being loaded into main memory.

Consider the situation where compiler 4 is being loaded into main memory and a new process request this compiler. If the residence status of compiler 4 had been marked "Memory" during loading, compiler manager would attempt to instruct the compiler to compile the process.

But the compiler is not even fully loaded yet!! If it had been marked "disk", the compiler manager would attempt to load compiler 4 again.

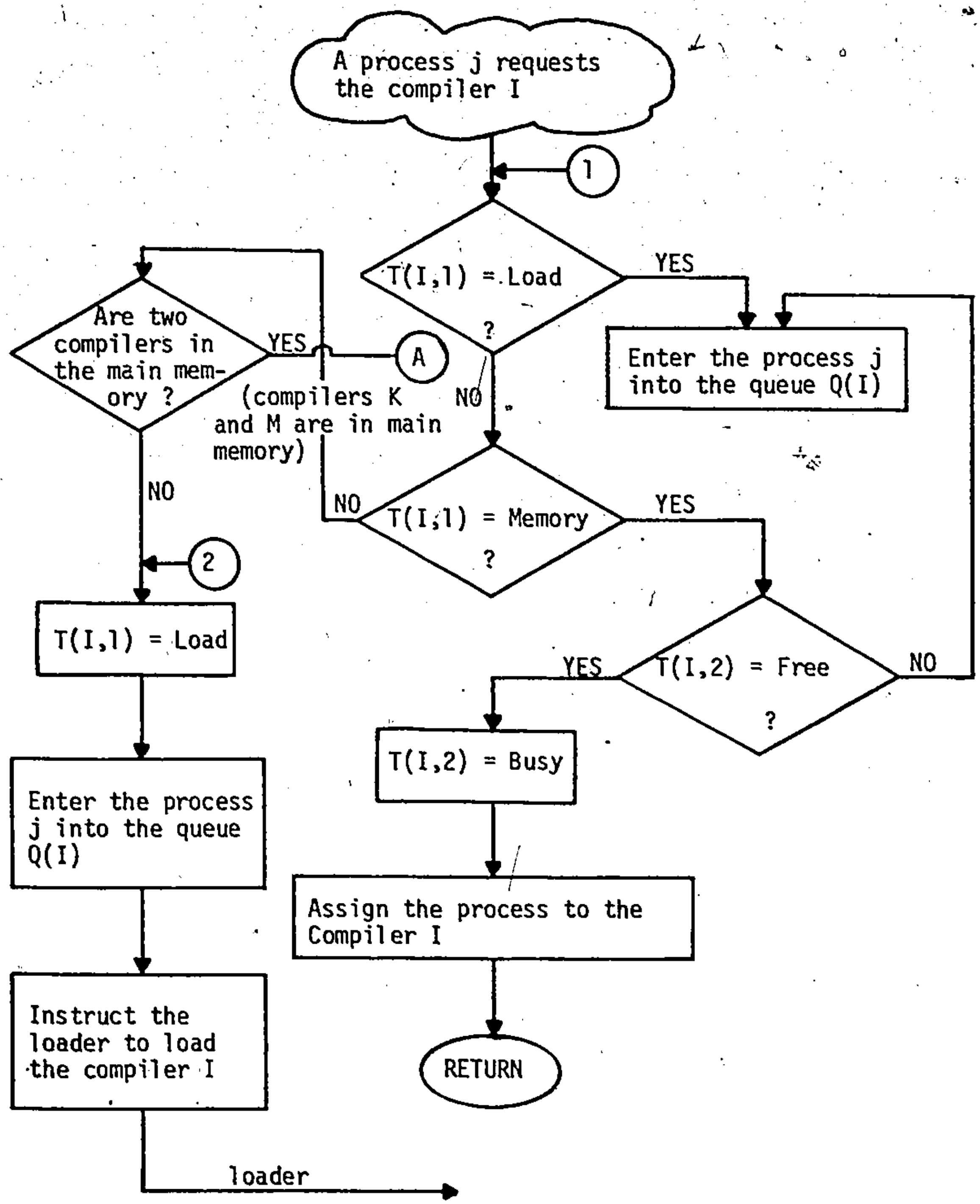
When a process requests a compiler and the residence status of the compiler is "Load" the compiler manager enters the process into the queue for the requested compiler. When the compiler has been loaded the compiler manager assigns to the compiler the process which is at the front of the queue associated with this compiler.

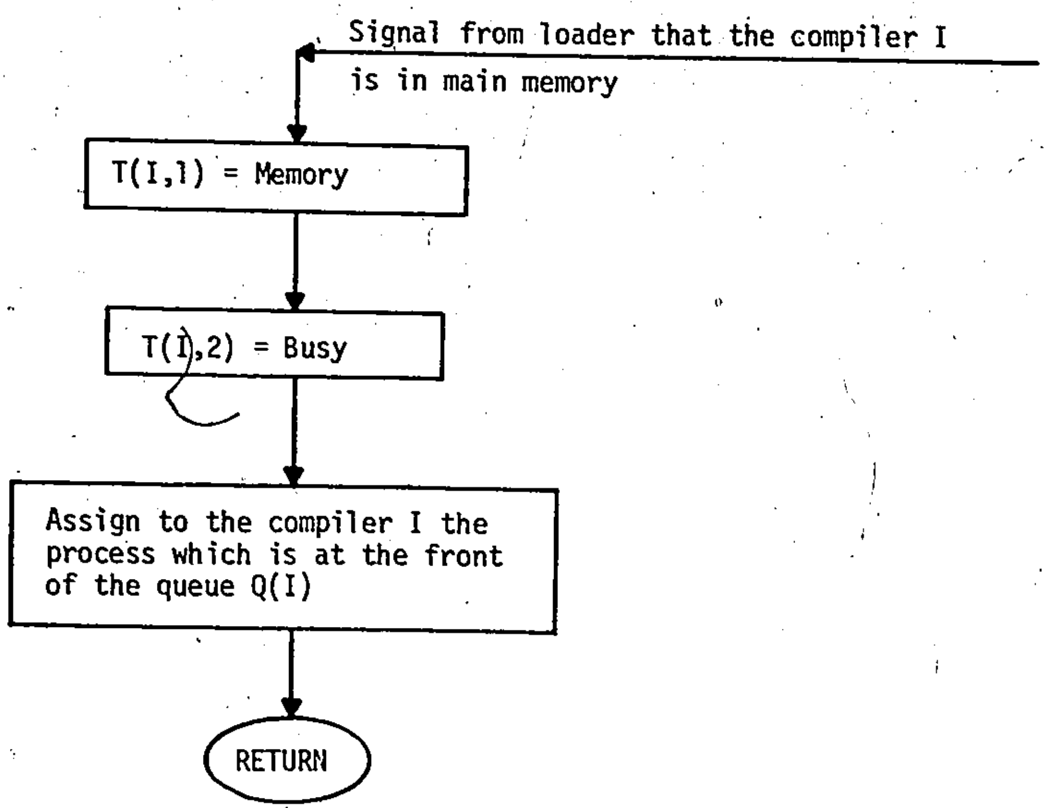
c) It creates a process whose task is to load into main memory compiler 4 and sends the process to the loader.

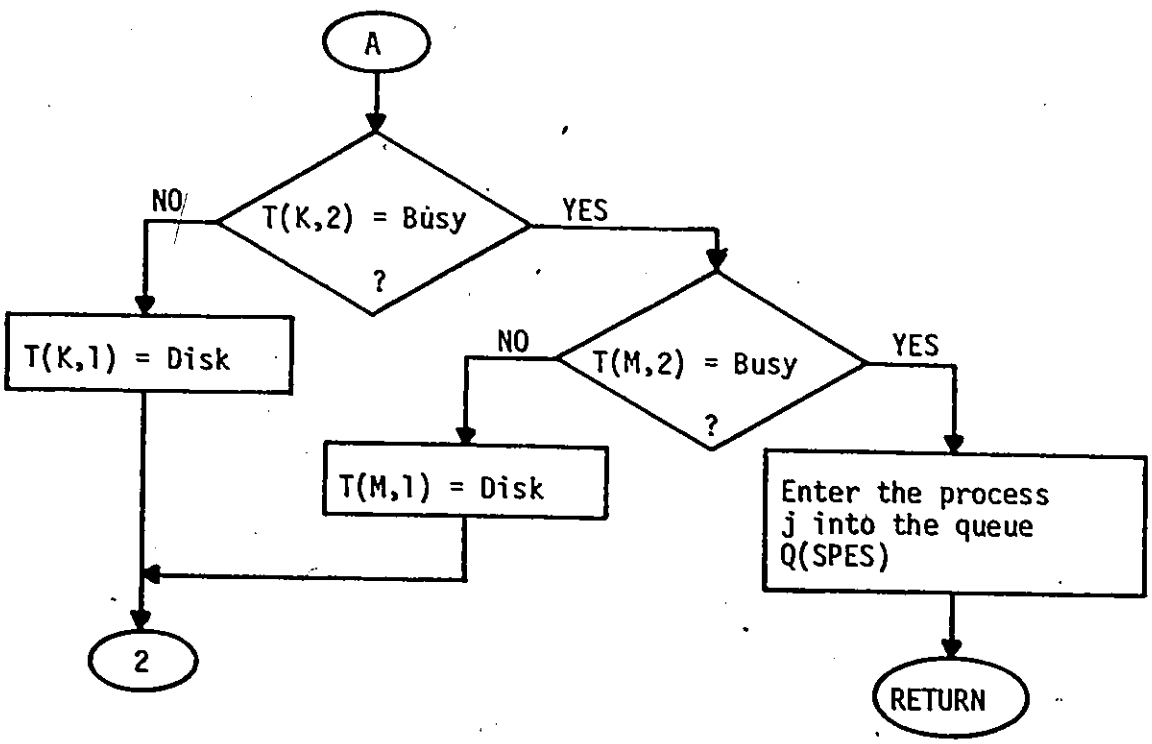
d) It enters process no. 9 into the queue for compiler 4. When compiler 4 has been loaded into the main memory, the compiler manager assigns to it the process which is at the front of the queue waiting for compiler 4 and changes it's status to in memory and busy.

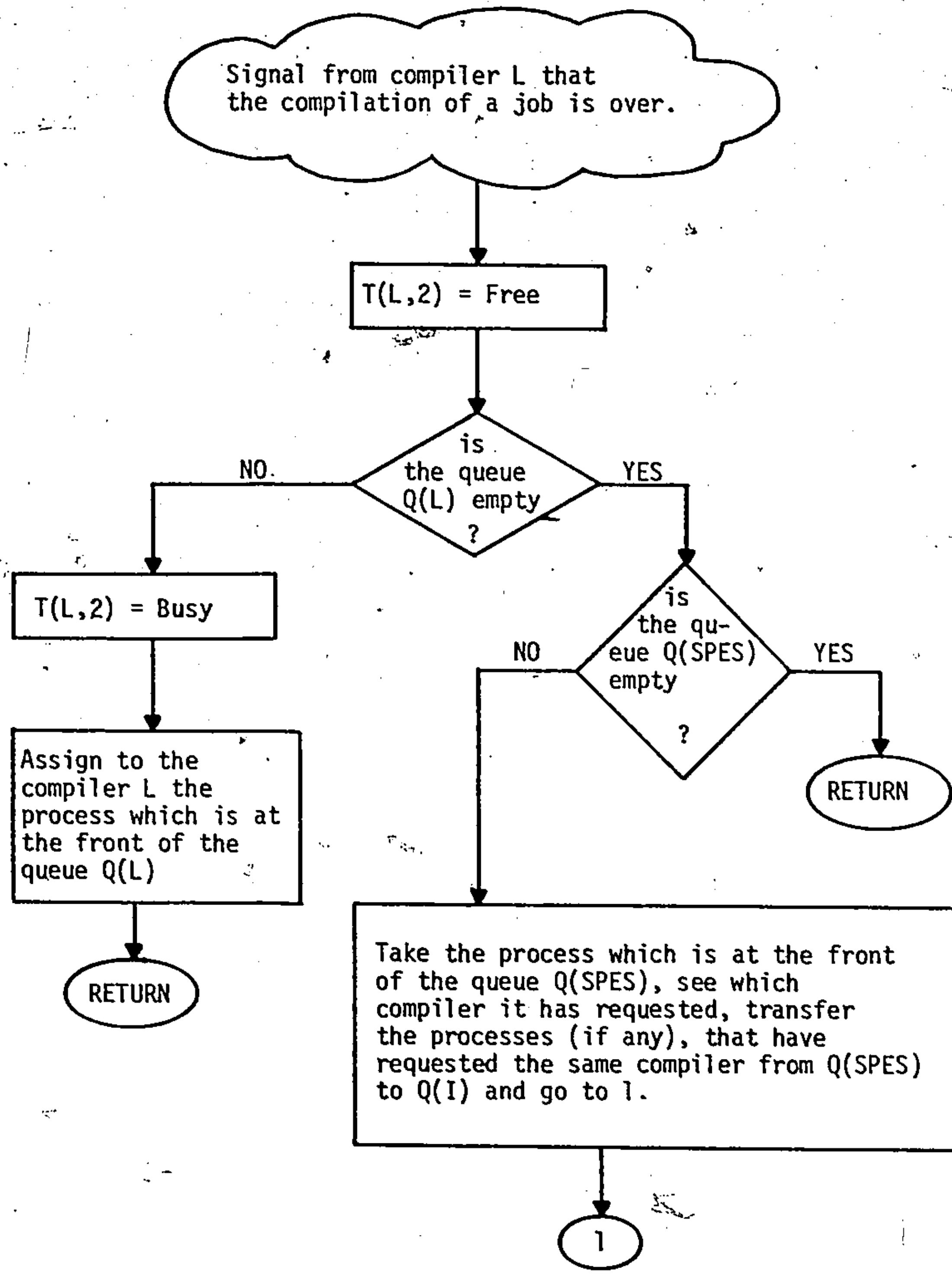
4.4.2.1 <u>Algorithm of the Compiler Manager</u>

Assume that T(I,1) is the residence status and T(I,2) the status of the compiler I, where T(I,1) = Memory, Disk, load, T(I,2) = Free, Busy, where I = 1, 2, 3, 4, 5. Also assume that Q(I) is the queue for compiler I and Q(SPES) is the queue "SPES" discussed earlier.


A process j requests the compiler I
1
T(I,1) = Load ?
YES
NO
Enter the process j into the queue Q(I)
Are two compilers in the main memory ?
YES
A
(compilers K and M are in main memory)
NO
T(I,1) = Memory ?
YES
NO
2
T(I,1) = Load
T(I,2) = Free ?
YES
NO
T(I,2) = Busy
Enter the process j into the queue Q(I)
Assign the process to the Compiler I
Instruct the loader to load the compiler I
RETURN
loader


Signal from loader that the compiler I is in main memory
T(I,1) = Memory
T(I,2) = Busy
Assign to the compiler I the process which is at the front of the queue Q(I)
RETURN


A
T(K,2) = Busy ?
NO
YES
T(K,1) = Disk
T(M,2) = Busy ?
NO
YES
T(M,1) = Disk
Enter the process j into the queue Q(SPES)
2
RETURN


Signal from compiler L that the compilation of a job is over.
T(L,2) = Free
is the queue Q(L) empty ?
NO.
YES
T(L,2) = Busy
Assign to the compiler L the process which is at the front of the queue Q(L)
RETURN
is the queue Q(SPES) empty ?
NO
YES
RETURN
Take the process which is at the front of the queue Q(SPES), see which compiler it has requested, transfer the processes (if any), that have requested the same compiler from Q(SPES) to Q(I) and go to 1.
1

4.5 Relocatable Loader

There is one relocatable loader in the simulated system. This is used to load into main memory compilers and the object code for the user programs.

The loader can load only one program at the time. If a process requests the loader while some other program is being loaded, the operating system will enter the process into a waiting queue "LOADQ". Here processes wait when the loader is busy. Processes are entered into the queue according to FCFS within a priority scheme.

As discussed earlier, the compilers and object user programs are in relocatable machine code stored on the input disk. Thus, the loader transfers, via channel, a block of relocatable machine code from the input disk into main memory, produces the equivalent absolute machine code and loads it into a PP partition of the main memory. This process is repeated until the complete code of the program is loaded into main memory.

The relocatable loader has been simulated by the segment block no. 327 to block no. 339, Appendix A.

4.6 Internal Priority

When a PP partition is assigned to the job by the job scheduler, the operating system assigns to the job a priority called the "internal priority". The internal priority of a job depends on the partition which has been assigned to the job. In turn, the assigned partition depends on the class to which the job belongs. Jobs that have been assigned to partitions 1, 2, 3, 4, 5, 6, 7 have internal priorities of 37 (highest), 36, 35, 34, 33, 32, 31 (lowest) respectively. A user may indirectly influence the actual internal priority. This is because his primary constraint is that he must assign a job class to the job that has been assigned to a partition large enough to run the job. In the event that only one partition has been assigned this class, his job must execute in that partition and the job's internal priority is then completely determined by the partition number. If, however, the job is assigned a class that can be serviced by more than one partition, then the job will encounter a different scheduling priority, depending on the partition in which it runs.

In order to achieve faster service to user programs, the software modules of the operating system (input spooler, output spooler, I/O device manager, job scheduler, CPU scheduler ect.) are assigned a higher internal priority (60) than the user programs. If, for example, the input spooler has an internal priority lower than the user programs, it will function very slowly. This results in a small number of jobs being placed into the system which in turn results in a poor utilization of the system resources. Figure 4-10 schematically represents the internal priority levels.

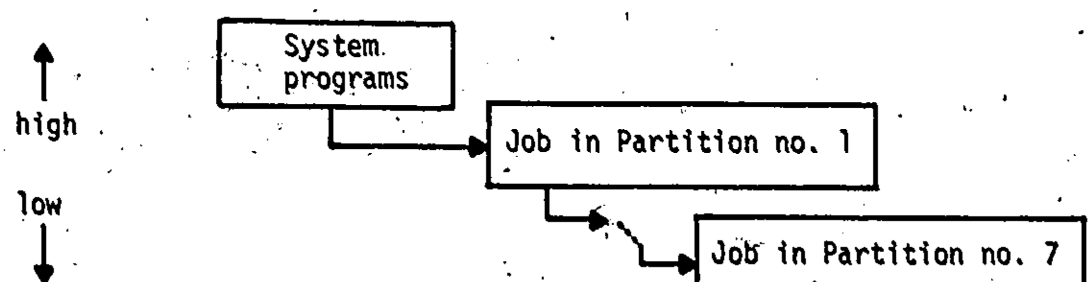


Figure 4-10. Internal priority levels

4.7 CPU Scheduler

The CPU scheduler is a software module in the operating system. It's purpose is to schedule the CPU among the active processes that are resident in the operating system partition and problem-program partitions, and performs the following functions:

1) Keep track of the state of processes.

2) Decide which process is to get the CPU, when, and for how long.

3) Allocate the CPU to processes.

4) Deallocate the CPU from a process.

The CPU scheduler in the simulated system uses a <u>H</u>ighest-<u>S</u>tatic-priority-<u>F</u>irst-<u>S</u>erved (HSFS) technique with pre-emption [MAD 74]. The CPU is assigned to a process with the highest internal priority among those waiting for the CPU. If a new process arrives with a higher internal priority than the one currently using the CPU, a pre-emption occurs.

To demonstrate how the user jobs and processes, created by the operating system, move from one state to another in the operating system, the process state diagram in Figure 4-11 is used.

<u>Hold state</u>: Here the user's job has been converted into an internal machine readable form, but no memory has been assigned to the job (the job has been SPOOLED onto disk).

<u>Ready state</u>: At some time, the job scheduler scans the input job queue on the input drum and picks a job to be admitted into the system. A partition of the main memory and an internal priority is

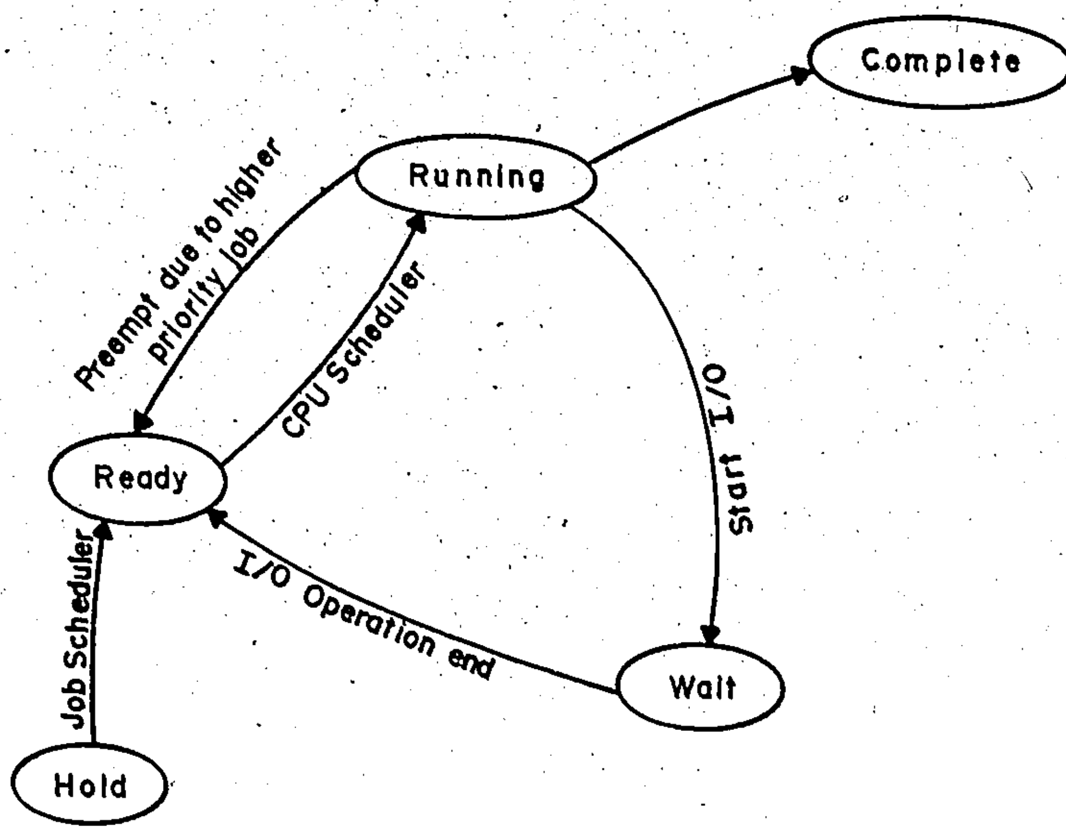


Figure 4-11. Process state diagram

assigned to the job. When the job has been compiled and loaded into main memory it becomes a process which is ready to run.

Running state: When the CPU becomes free, the CPU scheduler scans the list of ready processes, chooses the process with the highest internal priority and assigns the CPU to this process. Suppose for example, that the CPU has been assigned to process A. The process will hold the CPU until one of the following events occur:

1) The process requests an I/O operation. In this case process A will create a process A1 whose task is to satisfy the I/O request of the father process A. The CPU scheduler moves process A to the Wait state; then moves process A1 to the Ready state and assigns the CPU to another process. When process A1 satisfies the father's request, it will be destroyed. Process A is then moved to the Ready state.

2) The process experience an error condition. If the error cannot be recovered by the system the process is moved to the Complete state.

3) A process B with higher priority than process A requests the CPU. In this case a pre-emption occurs i.e., the CPU scheduler deallocates the CPU from process A, moves process A to Ready state and assigns the CPU to process B.

4) The process has completed its computation. The CPU scheduler moves the process to the Complete state.

Because of the HSFS scheduling rule, CPU scheduling can result in "poor" or "good" use of the system's CPU and I/O resources. Poor scheduling results when a CPU-bound job has been scheduled into the highest priority partition 1. CPU-bound jobs rarely go into the wait state and I/o-bound jobs in lower priority partitions will have little opportunity to execute. With a similar job mix, "good" scheduling can be achieved if I/O-bound jobs are scheduled in higher-priority partitions. They will frequently go into the wait state (while processing I/O) and allow time for lower-priority CPU-bound jobs to execute. More system activity, and thus a higher thruput, will result than with the previous "poor" job-scheduling arrangement.

The "pre-emption" characteristic of the HSFS with preemption technique used by the CPU-scheduler has been implemented in the GPSS program by associating the CPU with a PREEMPT block.

Through HSFS with preemption technique, we have accomplished two major objectives:

1) good CPU scheduling, resulting in improved system thruput and high utilization of the CPU,

2) good use of the main memory resource with little main memory fragmentation.

4.8 I/O Device Management

This section focuses on the management of I/O devices. These include the I/O devices such as the input drum, input disk, output disk, and the supporting device control channel. The block diagram of the GPSS model of the I/O device manager is presented with a detailed discussion.

The basic functions of I/O device Management are:

1) To keep track of the status of all devices (input drum, input disk, output disk, and channel).

2) To decide on the policy to determine which process gets a device, when and for how long.

3) To assign a device to a process. A device is assigned for as long as the process needs the device.

4) To deallocate a device.

All I/O operations in the simulated operating system are divided into two categories. To the first category belong those that involve a direct access storage device (input disk, input drum and output disk). The corresponding I/O operations are to transfer information, via channel, from main memory to a direct access storage device (DASD) or from a DASD to main memory. The second category consists of I/O operations that do not involve any DASD. Those operations are to transfer information from the buffer of a card reader to main memory or from main memory to the buffer of a line printer.

Each of the I/O devices is associated with a queue where a process is placed when the device, which has been requested by the process, is busy. The queue ordering is based on FCFS with a priority scheme. Figure 4-12 shows how the I/O device management is modelled.

When a process needs an I/O operation, it creates an I/O process, say B, and sends it to the I/O device manager, this process carries a message from the father process for the I/O device manager.

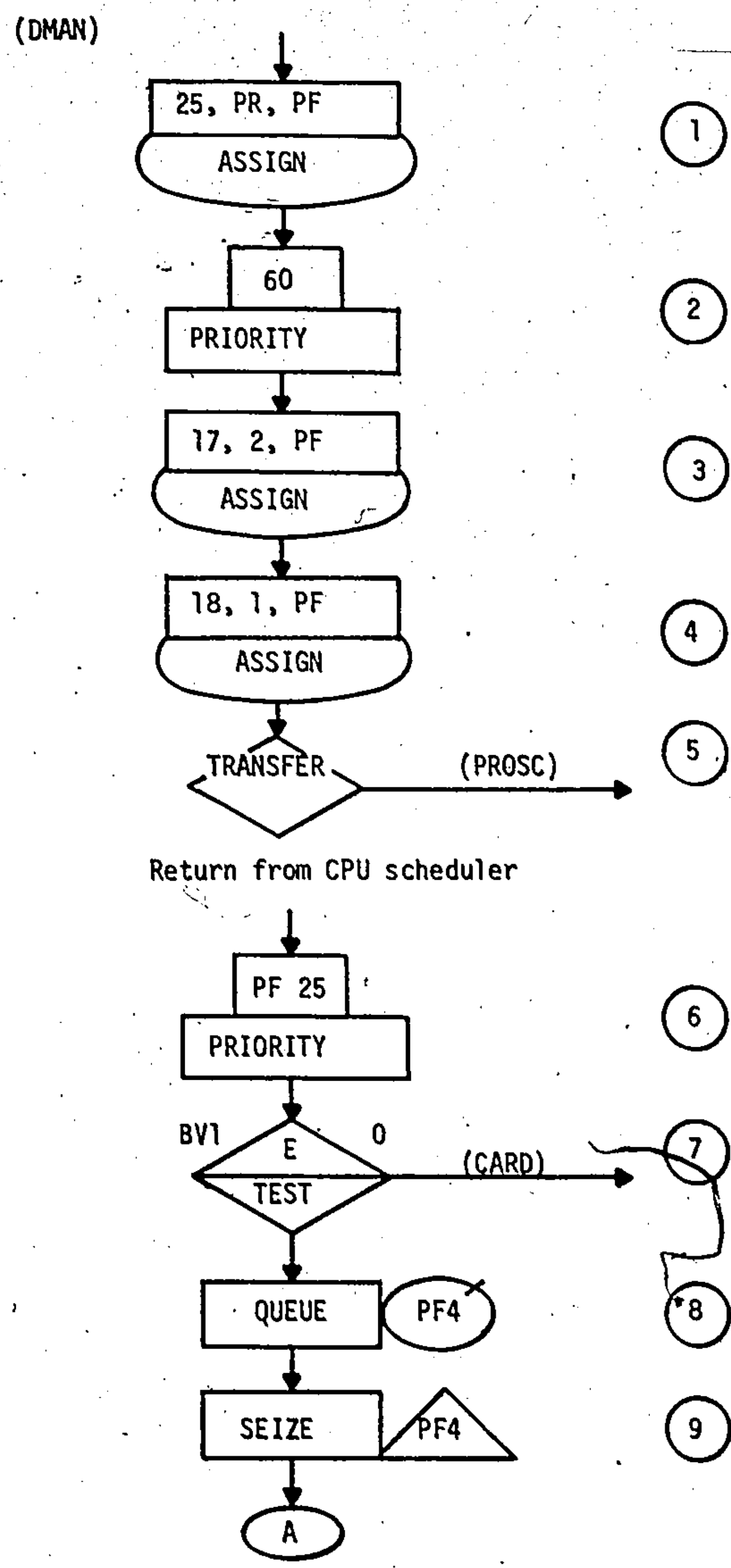
This message contains such information as, the location of the stored data, where the data is to be transfered, the size of the data block and the address to which the son process should be returned. Process B is represented in GPSS by a transaction whose message is stored in the transaction fullword parameters. The transaction (process B) enters the I/O device management model (Figure 4-12) at block 1 where it's internal priority is stored in the fullword parameter 2.5. This is necessary since the original form of the transaction is going to be changed as it moves through the various blocks of the I/O device management model. The I/O device manager needs the CPU in order to examine the message carried by the transaction (process B). Thus, it creates a process say A which will be sent to the CPU scheduler. The transaction now changes form and becomes a second process, A. Block no. 2 assigns to the transaction (process A) the internal priority 60, because the I/O device manager is one of the operating system programs. Block no. 3 assigns to the fullword parameter 17, the return address form the CPU scheduler, while block no. 4 assigns to the fullword parameter 18, the time for which process A needs the CPU. TRANSFER block no. 5 sends this transaction (process A) to the CPU scheduler. When the message is examined, the transaction (process A) returns to location AAA2, where process A is destroyed. At this time the transaction takes its original form (process B). Block no. 6 restores the internal priority of the transaction (process B) and if the I/O operation requested by the process B belongs to category two the transaction (process B) is sent by the TEST block no. 7 to the location

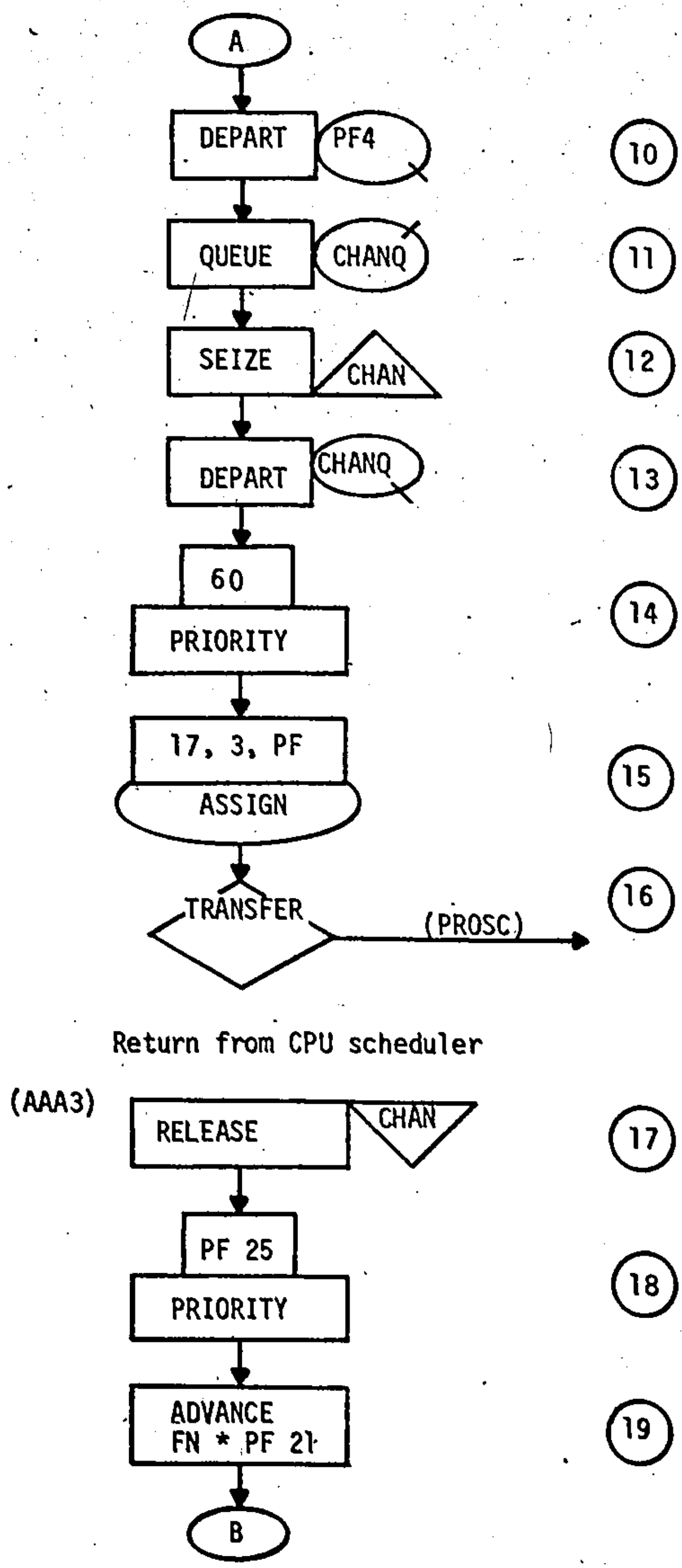
CARD.

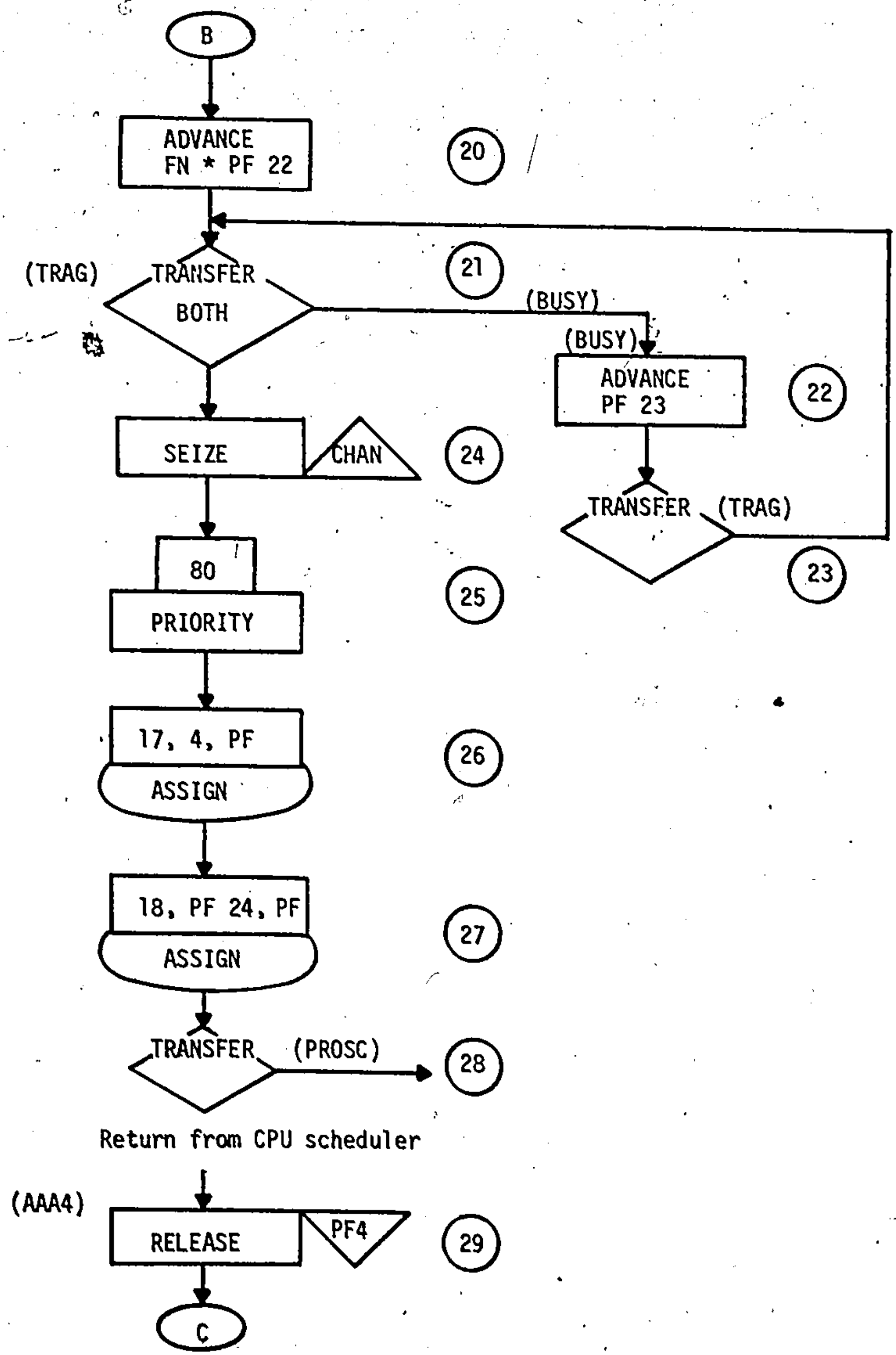
Let us assume that the I/O operation requested belongs to category 1 i.e., that the I/O operation involves a Direct Access Storage Device (DASD). Since a DASD (disk or drum) can handle only one data transfer at a time; it is represented by a facility (block 9).

If the DASD requested by the process B is busy the I/O device Manager places process B into a queue represented by the QUEUE block no. 8. When the DASD becomes free the transaction (process B) will "Seize" it by entering into the SEIZE block no. 9. Next, the transaction (process B) enters the DEPART block no. 10 which simulates the event "process B is leaving the queue". At this point the channel is required to transfer the information requested by process B. The channel is represented by a facility (block no. 12) since it can handle only one data transfer at a time. If the channel is busy at this time, the transaction (process B) has to wait in the queue CHANQ (block no. 11). When the channel becomes free the transaction will "seize" the channel (block no. 12) and it will leave the queue CHANQ (block 13). So far the I/O device manager has assigned to process B the requested DASD and the channel. The CPU is required to issue a command to the DASD. The command is to start the input-output operation by signaling to the DASD the track it is to locate. Thus, the I/O device manager creates another process C which is to be sent to the CPU scheduler. The transaction changes its original form to become process C. Block no. 14 assigns to transaction (process C) the internal priority 60. TRANSFER block no. 16 sends the transaction to the CPU scheduler.

When the start command is issued by the CPU the transaction (process C) returns to location AAA3 where, process C is destroyed. While the DASD is searching to find the proper track the channel is released (block no. 17) for other tasks. If process B has requested a disk (move head-disk), block no. 19 provides a delay while the reading head moves to the proper track. If process B has requested a drum (fixed head-drum) then this delay time is zero. A further delay is provided by block no. 20 while the beginning of the track record comes under the reading head. When this last delay is finished, the channel must be available for transmission of data. A test at the conditional TRANSFER block no. 21 checks the availability of the channel. If the channel is busy, it sends the transaction (process B) to ADVANCE block no. 22 to return for another try after one full revolution time. When the channel is available at the appropriate time, the CPU has to be interrupted for a time equal to the data transmission time. Thus, the I/O device management creates a new process D which is to be sent to the CPU scheduler. The PRIORITY block no. 25 assigns to transaction (process D) the highest internal priority 80 in the simulated system. This highest internal priority is required in order to interrupt the CPU. This interruption is represented by a PREEMPT block in priority mode. Block no. 26 assigns to the fullword parameter 17, the return address from the CPU scheduler, while block 27 assigns to the fullword parameter 18, the data transmission time. The data transmission time depends upon the data size and the channel speed. TRANSFER block no. 28 sends the transaction (process D) to the CPU scheduler. When the operation has been completed, transaction (process D) is returned to I/O device manager where process D is destroyed. The transaction here takes its original form (process B). The I/O device manager deallocates the DASD (block no. 29) and the channel (block no. 30), restores the internal priority of process B (block no. 31) and returns process B (block no. 32) to the point from where it was originated.


(DMAN)
25, PR, PF
ASSIGN
60
PRIORITY
17, 2, PF
ASSIGN
18, 1, PF
ASSIGN
TRANSFER
(PROSC)
Return from CPU scheduler
PF 25
PRIORITY
BV1
E
0
TEST
(CARD)
QUEUE
PF4
SEIZE
PF4
A
1
2
3
4
5
6
7
8
9


A
DEPART
PF4
10
QUEUE
CHANQ
11
SEIZE
CHAN
12
DEPART
CHANQ
13
60
PRIORITY
14
17, 3, PF
ASSIGN
15
TRANSFER
(PROSC)
16
Return from CPU scheduler
(AAA3)
RELEASE
CHAN
17
PF 25
PRIORITY
18
ADVANCE
FN * PF 21
19
B


B
ADVANCE
FN * PF 22
20
(TRAG)
TRANSFER
BOTH
21
(BUSY)
(BUSY)
ADVANCE
PF 23
22
SEIZE
CHAN
24
TRANSFER
(TRAG)
23
80
PRIORITY
25
17, 4, PF
ASSIGN
26
18, PF 24, PF
ASSIGN
27
TRANSFER
(PROSC)
28
Return from CPU scheduler
(AAA4)
RELEASE
PF4
29
C

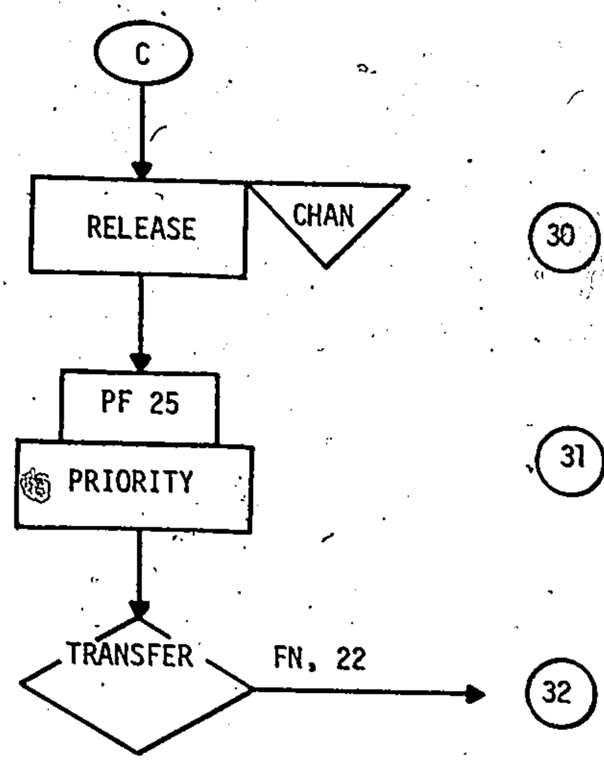


Figure 4-12 Simulating model of I/O device management

4.9 Output Spooler

The output spooler or system writer is the software module of the simulated operating system whose primary objective is to improve the system's thruput. This is accomplished by overlapping job processing, and system output. System output here refers to print output, including system messages, and user job output printed by the high-speed printer.

The basic function of the output spooler is to print the system output stored on one of the direct-access storage devices (output disk). The following discussion explains how this is accomplished.

When the execution of a job has been completed the output spooler inserts the job into the system's output job queue. This queue is divided into subqueues, one for each line printer. Jobs having only a small number of output lines to be printed will get the line printer earlier than jobs having a large number of output lines. If the line printer requested by the input station from which the job entered the system is busy and the queue associated with the line printer has more than six entries, the job is entered into a queue setaside for the auxiliary line printer, provided that the auxiliary queue does not have more than four entries.

The output spooler has one buffer in main memory for each line printer. Each buffer is capable of holding a block of 10 output lines.

When a line printer has been assigned to a job, the output spooler transfers, via the channel, a block of 10 output lines from the output disk to the buffer in main memory corresponding to the line printer. Each output line is then transferred, via the channel from the buffer of the spooler to the buffer of the line printer from where it is printed. While the line printer is printing the tenth line, the output spooler transfers, via the channel, the next block into buffer area in main memory. Again the process is repeated until the job has completed printing.

In executing the above steps we require 11 transfers to print a block of 10 lines. The first is to transfer a block of 10 output lines from the output disk to the buffer of the output spooler, while the rest are to transfer the 10 lines one at a time from the buffer of the output spooler to the buffer of the line printer.

An alternative to the above method is to transfer one output line at a time from the output disk to the buffer of the line printer. Though this would require only 10 transfers in order to print 10 lines, the first alternative is faster. The proof to this argument is shown by the following typical example. In this example we will calculate the time required to print a job consisting of 30 lines, using both methods and compare the results.

Let us assume that,

T1 (3 ms) represents the time required by the CPU to issue a start input-output command to the channel,

T2 (25 ms) represents the average time taken by the arm of the disk to move into the required track, T3 (10 ms) represents the average time taken for the beginning of the track record to come under the reading head,

T4 (4 ms) represents the time taken by the channel to transfer a block of 10 lines,

T5 (3 ms) represents the time taken by the channel to transfer one line,

T6 = T1+T2+T3+T4 (42 ms) represents the time required to transfer a block of 10 lines from the input disk to the buffer of the output spooler,

T7 = T1+T5 (6 ms) represents the time required to transfer a line from the buffer of the output spooler to the buffer of the line printer,

T8 = T1+T2+T3+T5 (41 ms) represents the time required to transfer a line from the output disk to the buffer of the printer,

and T9 (60 ms) represents the time taken by the line printer to print a line.

Using the first method to print a job consisting of 30 lines we need: T6+30(T7+T9) or 2022 ms. With the second method, we would require 30(T8+T9) or 3030 ms. From these simple calculations it is obvious that the first method is preferable since we save 1008 ms.

In the previous discussion we did not take into consideration the total waiting time for the queues of the output disk, channel and CPU. If we take the waiting queues into consideration it can be shown that the first method is better than the second.

From the GPSS segment of the I/O device management in Figure 4-12 we can see that a process with the task to transfer information, via channel, from the output disk to the main memory or to a line printer, has to pass through the following queues:

a) the queue for the output disk,

b) the queue for the channel,

c) the queue for the CPU, and

d) then again, the queue for the channel.

Thus the process experiences a total of four waiting queues.

A process with the task to transfer information, via channel, from main memory to the line printer, has to pass through the queue for the channel. Thus, using the first method to print a block of 10 lines can require a total of 14 waiting queues, while the second method can require a total of 40 waiting queues. In general, it can be assumed that the first method will have less total waiting time than the second method.

To support our argument we simulated the SCOS for a period of 16 minutes using both methods. By using the second method, the average time required to print a job increased from 13845 ms to 21177 ms. In addition, the utilization of the line printer was increased from 64 percent to 89 percent. The average waiting time of the jobs in the output queue was increased from 15600 ms to 83363 ms and the auxiliary line printer was required to assist in the printing process of the system. The use of the second method also effected the average turn-around time by increasing it from 0.017 hours to

0.025 hours.

This concludes the discussion on software-modules of the simulated operating system. In the next section we present results and conclusions.

5. RESULTS AND CONCLUSIONS

In this section we will present the major characteristics of the Student-Created Operating System (SCOS), the statistical results obtained from the simulated model indicating utilization of various devices and ending with a brief discussion on the advantages and disadvantages of this approach.

The following discussion is based upon the statistics obtained by simulating SCOS for a period of seven hours. The model operates with a basic time unit of 1 ms. A 60 second simulation of the SCOS model requires approximately 13 seconds of CPU time on CDC/6400 machine. The GPSS entities required to simulate the SCOS are given in Figure 5-1.

The simulated system has two card readers and one line printer. Jobs arrive at card readers one and two in two exponential streams with average interarrival times of 15 seconds and 2 minutes respectively. Figure 5-2 shows the characteristics of the user jobs that arrive at card readers one and two. These characteristics are given to the jobs that arrive to the card readers one and two by the functions no. 1 and no. 2 (Appendix A) respectively.

The main memory available to the users is 180K bytes and is divided into seven PP partitions. Figure 5-3 shows the job classes that each PP partition may serve and the sizes of the partitions. As discussed earlier this is represented in the GPSS model by the matrix no. 1.

GPSS ENTITYS	NUMBER
Blocks	482
Facilities	22
Storage	1
Queues	11
Logics	30
Tables	14
Functions	45
Variables	17
Fullword savevalues	29
Halfword savevalues	7
Fullword Matrixes	2
Chains	20

Figure 5-1. GPSS entities used by the simulated model

JOB CLASSES	CARD READER 1	CARD READER 2
A	35 %	8%
B	24 %	5 %
C	22 %	6 %
D	8 %	11%
E	7 %	16%
F	2 %	25%
G	2 %	29%

Figure 5-2. Job class distribution among card readers one and two

PARTITION	JOB CLASSES	SIZE (in K bytes)
1	A, B, C	10
2	A, B, C	20
3	C, B, A	20
4	C, D, E, G	25
5	D, C, E, G	30
6	F, G, D, C	35
7	F, G	40

Figure 5-3. Partition specifications

5.1 Equipment Statistics

The GPSS simulator has gathered statistics for CPU, card readers, line printer, channel, input disk, output disk, and input drum. Figure 5-4 shows the statistics gathered by the GPSS "Facilities" which represent the various equipment in the simulated model of SCOS, while Figure 5-5 shows the statistics gathered by the GPSS "Queues" which represent the waiting lines formed in front of the equipment. For the reader who is not familiar with GPSS terminology used in the tables shown in Figure 5-4 and 5-5, we give a brief definition. Additional information can be found in [GOR 75], [SCH 74]. Using the terminology given in [SCH 74], for the table in Figure 5-4 we define the average utilization as the fraction of time that the corresponding devices (Facilities) were in use during the simulation. We define the number entries as the number of captures during the simulation and the average time/tran as the average holding time per capture.

For the table in Figure 5-5, we define the maximum contents as the largest number of items which were simultaneously waiting during simulation, total entries as the total number of items which entered the Queue, zero entries as the number of items which were not delayed, the average time/tran as the average time that each item waited in the Queue (zero entries are included in this average), and the $average time/tran as the average time that each delayed item waited in the Queue (zero entries are excluded from this average).

DEVICE	AVERAGE UTILIZATION (In parts per hundred)	NUMBER ENTRIES	AVERAGE TIME/TRAN (in milliseconds)
C P U	99	1410720	8.161
Card reader 1	61	811	8578.977
Card reader 2	17	97	19543.979
Line printer	75	907	15051.731
Channel	19	382008	5.485
Input disk	20	42992	52.671
Output disk	9	19494	51.290
Input drum	2	5253	16.254

Figure 5-4. Statistics of the devices

QUEUE FOR THE DEVICE	MAXIMUM CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	AVERAGE TIME/TRAN (in milliseconds)	$ AVERAGE TIME/TRAN (in milliseconds)
C P U	6	1410721	1306700	1.57	21.32
Card reader 1	6	811	318	7001.97	11518.45
Card reader 2	2	97	72	3536.68	14142.71
Line printer	7	907	424	—	22688.65
Channel	3	314296	310431	0.05	4.04
Input disk	3	42992	37892	4.39	37.05
Output disk	2	19494	18697	1.11	27.15
Input drum	1	5253	5202	0.11	10.82

Figure 5-5. Statistics of the waiting queues for the devices

From the equipment statistics we can conclude the following:

1) The simulated CPU was performing useful work; that is, idling less than 1% of the simulated run time. From this information the operating system, and particularly, the CPU scheduler has succeeded in scheduling efficiently the CPU among the system and user programs. This high utilization of the CPU also indicates that the system was overloaded during the simulation.

2) The number of I/O devices and channel are sufficient for the simulated system to operate efficiently, since the majority of the processes that had requested these devices did not have to wait in order to get the requested device. For example, from the statistics for the queue of the input disk (Figure 5-5) we can see that 88 per-cent of the processes (the ratio of zero entries to total entries) that had requested the input disk did not have to wait in order to capture the disk. Also, the maximum contents of the queue for the input disk was only three processes.

5.2 Main Memory Statistics and Fragmentation

The statistics gathered by the GPSS simulator show that utilization of the main memory was 91 percent, while its average contents were 6.4 jobs. Figure 5-6 shows the fragmentation of main memory calculated by the GPSS program whenever all of the seven PP partitions were occupied by user jobs. Figure 5-7 shows the mean and standard deviation of the GPSS weighted tables used to tabulate the fragmentation of each PP partition weighted by the time that a job occupied a partition. Figure 5-8 shows the mean and standard deviation of the GPSS non-weighted tables used to tabulate the fragmentation of each pp partition.

From the above statistics we can conclude that the main memory was well utilized with relatively small fragmentation.

The mean arguments and standard deviations shown in Figure 5-7 are very similar to those shown in Figure 5-8, with the only exception being that of partition six. This is due to the fact that most of the user jobs that require a large amount of execution time (i.e., jobs that belong to classes G and F), were scheduled in partition six. Thus, these jobs held partition six longer than jobs scheduled in other partitions, and since the fragmentation in Figure 5-7 is weighted by the time that a job occupies a partition, the mean and standard deviation for partition six in Figure 5-7 is greater than the corresponding mean and standard deviation shown in Figure 5-8.

The problem of fragmentation cannot be completely solved with a static partition specification scheme. No matter how the main memory is partitioned, it still will be fragmented. This problem can be partially solved by other schemes, such as dynamic partition specification, relocatable partitioned memory, paging etc., [MAD 74], [SHA 74], [TSI 74].

MEMORY FRAGMENTATION (in parts per hundred)	TIMES OF OCCURRENCE
15	6
16	13
17	24
18	37
19	22
20	34
21	34
22	39
23	26
24	4
25	6

Figure 5-6. Memory fragmentation

PARTITION		MEAN ARGUMENT (in K bytes)	STANDARD DEVIATION (in K bytes)
No	SIZE (in K bytes)		
1	10	0.995	0.822
2	20	5.244	3.024
3	20	4.545	2.978
4	25	1.988	1.475
5	30	6.250	2.716
6	35	8.328	5.638
7	40	2.983	0.446

Figure 5-7. Fragmentation for each PP partition (weighted)

PARTITION No.	PARTITION SIZE (in Kbytes)	MEAN ARGUMENT (in K bytes)	STANDARD DEVIATION (in K bytes)
1	10	1.000	0.814
2	20	5.134	2.921
3	20	4.545	3.032
4	25	1.679	1.528
5	30	2.365	1.805
6	35	1.659	2.068
7	40	2.211	1.512

Figure 5-8. Fragmentation for each PP partition (non-weighted)

5.3 Job Statistics

In the simulation period of seven hours, 907 user jobs were processed by the SCOS. The number of jobs in each job class is given in Figure 5-9, while Figure 5-10 shows the average contents of the job input queues for various classes.

For each of the user jobs in the system the GPSS program gathers statistics, such as job class, external and internal priority, the code of the compiler that a job has requested, main memory and CPU time used by a job, job turn-around time, etc. Figures 5-11 through 5-15 show the statistics for the first five jobs processed by the system. As was expected, jobs with higher internal priority got better service by the operating system than jobs with lower internal priority. The job turn-around time is defined as, $T_{j\ out} - T_{j\ in}$, while the average job turn-around time is defined as,

$$\frac{\sum_{i=1}^{N_c} (T_{j\ out} - T_{j\ in})_i}{N_c} \quad \text{where,}$$

$T_{j\ in}$ is the time at which the job j arrives at the card reader,

$T_{j\ out}$ is the time at which the line printer has completed printing a job,

N_c is the total number of jobs processed to completion during a period of time.

The average job turn-around time in a period of four hours of simulation computed by the GPSS program, was 0.06 hours. In this

CLASS	NUMBER OF JOBS
A	309
B	203
C	171
D	81
E	66
F	35
G	42

Figure 5-9. Number of jobs in various classes

JOB INPUT QUEUE FOR CLASS	AVERAGE CONTENTS (No. of jobs waiting)
A	0.48
B	3.30
C	35.60
D	30.09
E	28.84
F	21.53
G	26.41

Figure 5-10. Average number of jobs waiting in the input queue for various classes

```
-----------------------------------------------
-                                             -
-  THIS IS A REPORT FOR THE JOB NO.     1     -
-                                             -
-----------------------------------------------

THE CLASS OF THE JOB IS    C
THE EXTERNAL PRIORITY ASSIGNED BY THE USER IS  2
THE INTERNAL PRIORITY ASSIGNED BY THE SYSTEM IS  36
THIS JOB HAD REQUESTED THE COMPILER  19
THE MAIN MEMORY USED BY THE JOB IS   16  K  BYTES
THE CPU TIME USED BY THE JOB IS     22256  MS
THE JOB ARRIVED AT INPUT STATION  1  AT THE TIME        22495  MS
THE JOB CAPTURED THE CARD READER AT THE TIME        22495  MS
THE TIME TAKEN BY THE CARD READER TO READ THE JOB IS   11115  MS
THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR           34 MS
THE TIME TAKEN BY THE COMPILER TO COMPILE THE JOB IS     810  MS
THE JOB DID NOT HAVE TO WAIT IN THE OUTPUT QUEUE
THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE JOB IS   15334  MS
THE TURNAROUND TIME OF THE JOB IS     5.617  HRS
```

Figure 5-11. Statistics for job no. 1

```
-----------------------------------------------
-                                             -
-  THIS IS A REPORT FOR THE JOB NO.     4     -
-                                             -
-----------------------------------------------


THE CLASS OF THE JOB IS     4
THE EXTERNAL PRIORITY ASSIGNED BY THE USER IS  4
THE INTERNAL PRIORITY ASSIGNED BY THE SYSTEM IS  36
THIS JOB HAD REQUESTED THE COMPILER    19
THE MAIN MEMORY USED BY THE JOB IS   19  K  BYTES
THE CPU TIME USED BY THE JOB IS        665  MS
THE JOB ARRIVED AT INPUT STATION  1  AT THE TIME          38269  MS
THE JOB CAPTURED THE CARD READER AT THE TIME         47949  MS
THE TIME TAKEN BY THE CARD READER TO READ THE JOB IS    3531  MS
THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR         17148 MS
THE TIME TAKEN BY THE COMPILER TO COMPILE THE JOB IS     437  MS
THE JOB HAD TO WAIT IN THE OUTPUT QUEUE FOR    9427  MS
THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE JOB IS     8765  MS
THE TURNAROUND TIME OF THE JOB IS      6.015  MS
```

Figure 5-12. Statistics for job no. 4

```
------------------------------------------------
-                                              -
-  THIS IS A REPORT FOR THE JOB NO.       5    -
-                                              -
------------------------------------------------

THE CLASS OF THE JOB IS     A
THE EXTERNAL PRIORITY ASSIGNED BY THE USER IS  4
THE INTERNAL PRIORITY ASSIGNED BY THE SYSTEM IS  36
THIS JOB HAD REQUESTED THE COMPILER   19
THE MAIN MEMORY USED BY THE JOB IS   13  K  BYTES
THE CPU TIME USED BY THE JOB IS        672  MS
THE JOB ARRIVED AT INPUT STATION  1  AT THE TIME          56128  MS
THE JOB CAPTURED THE CARD READER AT THE TIME          56128  MS
THE TIME TAKEN BY THE CARD READER TO READ THE JOB IS     9945  MS
THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR          8525 MS
THE TIME TAKEN BY THE COMPILER TO COMPILE THE JOB IS     1545  MS
THE JOB HAD TO WAIT IN THE OUTPUT QUEUE FOR   14275  MS
THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE JOB IS   13436  MS
THE TURNAROUND TIME OF THE JOB IS      6.814  HRS
```

Figure 5-13. Statistics for job no. 5

```
-----------------------------------------------
-                                             -
-  THIS IS A REPORT FOR THE JOB NO.      6    -
-                                             -
-----------------------------------------------

THE CLASS OF THE JOB IS     C
THE EXTERNAL PRIORITY ASSIGNED BY THE USER IS  4
THE INTERNAL PRIORITY ASSIGNED BY THE SYSTEM IS  35
THIS JOB HAD REQUESTED THE COMPILER   19
THE MAIN MEMORY USED BY THE JOB IS   14  K  BYTES
THE CPU TIME USED BY THE JOB IS      25623  MS
THE JOB ARRIVED AT INPUT STATION  1  AT THE TIME          69372  MS
THE JOB CAPTURED THE CARD READER AT THE TIME          69372  MS
THE TIME TAKEN BY THE CARD READER TO READ THE JOB IS   10127  MS
THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR              49 MS
THE TIME TAKEN BY THE COMPILER TO COMPILE THE JOB IS     675  MS
THE JOB DID NOT HAVE TO WAIT IN THE OUTPUT QUEUE
THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE JOB IS  14693  MS
THE TURNAROUND TIME OF THE JOB IS     [illegible]17  MS
```

Figure 5-14. Statistics for job no. 6

```
THIS IS A REPORT FOR THE JOB NO.     3

THE CLASS OF THE JOB IS    C
THE EXTERNAL PRIORITY ASSIGNED BY THE USER IS  4
THE INTERNAL PRIORITY ASSIGNED BY THE SYSTEM IS  35
THIS JOB HAD REQUESTED THE COMPILER   17
THE MAIN MEMORY USED BY THE JOB IS   16  K  BYTES
THE CPU TIME USED BY THE JOB IS      33928  MS
THE JOB ARRIVED AT INPUT STATION  1  AT THE TIME         25341  MS
THE JOB CAPTURED THE CARD READER AT THE TIME         43327  MS
THE TIME TAKEN BY THE CARD READER TO READ THE JOB IS     4941  MS
THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR           47 MS
THE TIME TAKEN BY THE COMPILER TO COMPILE THE JOB IS     791  MS
THE JOB DID NOT HAVE TO WAIT IN THE OUTPUT QUEUE
THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE JOB IS  12066  MS
THE TURNAROUND TIME OF THE JOB IS      6.039  HRS
```

Figure 5-15. Statistics for job no. 3

period the simulated system processed 507 user jobs where the job mix for the system consisted of small jobs (i.e., jobs that belong to classes A, B and C). The average job turn-around time in a period of seven hours of simulation was 1.1 hours. This increase in job turn-around time is due to the fact that after four hours of simulation, the system processed jobs having large turn-around times (i.e., jobs that belonged to classes D, E, F and G). The large turn-around time for these jobs is caused by the following facts:

a) Jobs that belong to classes D, E, F and G use more CPU time with lengthier output. Thus, they remain in main memory and in the output job queue longer than the other jobs.

b) Jobs that belong to classes D, E, F and G are scheduled in PP partitions with low priorities, which in turn results in low internal priorities. In general, the CPU was busy over the four hour period processing jobs with high internal priorities that were continuously coming into the system. Thus, jobs with a low internal priority had a poor chance in using the CPU, and therefore increasing their turn-around time. After four hours of simualtion the system did not accept any additional jobs submitted to the input stations. At this point, jobs already in the system were executed.

5.4 Summary

The primary goal of this project was to simulate the behavior of a student-created operating system (SCOS) using GPSS. The GPSS model for the SCOS behaves quite efficiently, on the machine and the environment that we have chosen, as reflected in the statistical results. Even though the SCOS is not a real system, it has the major characteristics of a small computer which can support multiprogramming. The SCOS has deliberately been designed in a modular and structured manner. As a result, the software modules of the operating system can easily be identified. In addition SCOS can easily be modified. More devices, such as card readers, line printers, disks, drums, tapes and channels can be added into the system. In Appendix B we list the GPSS statements that have to be added in order to supply the GPSS model with one additional input station. The size of the main memory and the partition specification can be changed by modifying only three to four statements of the GPSS program.

One obstacle in using GPSS to simulate operating systems is in learning the GPSS language. Since students usually have taken several programming courses before studying operating systems, learning the fundamentals of another language such as GPSS would not be difficult.

Operating systems can be simulated to any degree of detail desired. Entire systems, as the one that we have implemented, portions of systems, or various implementation strategies can be simulated. This flexibility makes it possible to study and analyze concepts such as multiprogramming scheduling methods, page fetch schemes, page replacement techniques, etc. Furthermore, GPSS can be used in conjunction with most text books in operating systems.

Problems could be taken directly from the textbook and solutions presented in GPSS. This approach has the advantage of allowing the students to investigate concepts as they are encountered rather than waiting until some later time to integrate them in a semester project. When the semester project is assigned, the amount of time required to supervise the students would be considerably less than most non-GPSS approaches since the students would not be trying to cope with the idiosyncrasies of a particular programming language or hardware system.

Using GPSS can be economical. Similar statistical results to those obtained with the simulation time of seven hours, can be achieved with a smaller simulation time of 15 minutes.

Simulation using GPSS can be an effective learning aid because it enables the student to grasp an overall view of operating system. The flexibility and ease of use of GPSS makes it possible to implement many concepts that might be impossible otherwise.

APPENDIX A

A Listing of the GPSS Program SCOS

```
      SUBROUTINE REPORT(I)
      DIMENSION I(6)
      K=I(1)
      GO TO (1,2,3,4,5,6,7,8,9,10,11),K
1     PRINT100,I(2),I(3),I(4)
100   FORMAT (//////*0*,50X,*TO JOB NO.*,I5/* *,57X,*20(*-*)/*0*,8X,*SOR
     1RY WE CANNOT RUN YOUR JOB. YOU HAVE REQUESTED*,I6,* KBYTES MAIN ME
     1MORY*/* *,8X,*THE MAIN MEMORY AVAILABLE TO THE USERS  IS*,I6,* K B
     1YTES*//////////)
      RETURN
2     PRINT 110,I(2),I(3),I(4),I(5),I(6)
110   FORMAT (*1*,27X,56(*+*)/* *,27X,*+*,54X,*+*/* *,27X,*+*,54X,*+*/*
     1*,27X,*+*,54X,*+*/* *,27X,*+*,2X,*THIS IS THE FRAGMENTATION MATRIX
     1 .*,18X,*+*/* *,27X,*+*,2X,*(THE ELEMENTS ARE EXPRESSED IN PARTS P
     1ER THOUSAND).*,1X,*+*/* *,27X,*+*,54X,*+*/* *,27X,*+*,54X,*+*/* *,
     127X,*+*,54X,*+*/* *,27X,56(*+*)///////* *,30X,5(I3,7X)//)
      RETURN
3     PRINT 120,I(2)
120   FORMAT(*1*,50X,45(*-*)/* *,50X,*-*,43X,*-*/* *,50X,*-*,43X,*-*/* *
     1,50X,*-*,43X,*-*/* *,50X,*-*,2X,*THIS IS A REPORT FOR THE JOB NO.*
     1,I6,3X,*-*/* *,50X,*-*,43X,*-*/* *,50X,*-*,43X,*-*/* *,50X,*-*,43X
     1,*-*/* *,50X,45(*-*)////////)
      IF(I(3).NE.1) GOTO 51
      CHAR=5H    A
      GO TO 121
51    IF(I(3).NE.2) GOTO 52
      CHAR=5H    B
      GO TO 121
52    IF(I(3).NE.3) GOTO 53
      CHAR=5H    C
      GO TO 121
53    IF(I(3).NE.4) GOTO 54
      CHAR=5H    D
      GO TO 121
54    IF(I(3).NE.5) GOTO 55
      CHAR=5H    E
      GO TO 121
55    IF(I(3).NE.6) GOTO 56
      CHAR=5H    F
      GO TO 121
56    CHAR=5H    G
121   PRINT 122,CHAR,I(4),I(5),I(6)
122   FORMAT (*0*,43X,*THE CLASS OF THE JOB IS*,A5,/*0*,43X,*THE EXTERNA
     1L PRIORITY ASSIGNED BY THE USER IS*,I3/*0*,43X,*THE INTERNAL PRIOR
     1ITY ASSIGNED BY THE SYSTEM IS*,I4/*0*,43X,*THIS JOB HAD REQUESTED
     1THE COMPILER*,I5)
      RETURN
4     IF(I(4).EQ.33) I(4)=3
      IF(I(4).EQ.34) I(4)=4
      IF(I(4).EQ.35) I(4)=5
      PRINT 130,I(2),I(3),I(4),I(5),I(6)
130   FORMAT(*0*,38X,*      THE MAIN MEMORY USED BY THE JOB IS*,I5,*  K
     1BYTES*/*0*,38X,*      THE CPU TIME USED BY THE JOB IS*,I10,*  MS*/*
     10*,43X,*THE JOB ARRIVED AT INPUT STATION*,I3,* AT THE TIME*,I13,*
     1  MS*/*0*,43X,*THE JOB CAPTURED THE CARD READER AT THE TIME*,I13,*
     1  MS*)
      RETURN
```

```
5       PRINT 140,I(2),I(3),I(5)
140     FORMAT(*0*,43X,*THE TIME TAKEN BY THE CARD READER TO READ THE JOB
       1IS*,I8,*  MS*/*0*,43X,*THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR*
       1,I13,* MS*/*0*,43X,*THE TIME TAKEN BY THE COMPILER TO COMPILE THE
       1JOB IS*,I8,*  MS*)
        IF(I(4).NE.0) GO TO101
        PRINT 102
102     FORMAT(*0*,43X,*THE JOB DID NOT HAVE TO WAIT IN THE OUTPUT QUEUE*)
        GO TO103
101     PRINT 104,I(4)
104     FORMAT(*0*,43X,*THE JOB HAD TO WAIT IN THE OUTPUT QUEUE FOR*,I8,*
       1 MS*)
103     PRINT 105,I(6)
105     FORMAT(*0*,43X,*THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT TH
       1E JOB IS*,I8,*  MS*)
        RETURN
6       X1=FLOAT(I(2))/3600000.
        PRINT 150,X1
150     FORMAT(*0*,43X,*THE TURNAROUND TIME OF THE JOB IS*,F10.3,*  HRS*)
        RETURN
7       PRINT 160
160     FORMAT (*1*,50X,45(*-*)//*0*,53X,*THIS IS A NON STANDARD JOB*//*0*
       1,50X,45(*-*)//)
        RETURN
8       Y1=FLOAT(I(3))/(FLOAT(I(2))*3600000.)
        PRINT 170,Y1
170     FORMAT(*1*//*0*,40X,52(*-*)//*0*,44X,*THE AVERAGE TURNAROUND TIME
       1IS*,F10.3,*  HRS*//*0*,40X,52(*-*)////////)
        RETURN
9       PRINT 520,I(2),I(3),I(4),I(5),I(6)
520     FORMAT(*0*,30X,5(I3,7X)//)
        RETURN
10      L=6
        DO 999 K=2,6
        M=9-K
        IF(I(M).NE.0) GO TO 992
        L=M-1
999     CONTINUE
992     IF(L.EQ.1) RETURN
        PRINT 521,(I(K),K=2,L)
521     FORMAT(*0*,30X,5(I3,7X)//)
        RETURN
11      PRINT 661
661     FORMAT (*1*,70X,*++++ ERROR ++++*/////*0*,60X,* SUM OF THE MEMORY
       1PARTITION SIZE REQUESTED EXCEEDS THE MAIN MEMORY AVAILABLE TO THE
       1USERS*)
        RETURN
        END
```

OMPILER SPACE

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *
        *
        *
        *
        *
        *
        *
        *
        *
        *
               SIMULATE
        *
        *
        *
        *
        *
        *  THIS IS A LIST OF THE MAJOR PARAMETERS (FULLWORD PARAMETERS) USED
        *  IN THIS PROGRAM.
        ***********************************************************************
        *
        *
        *
        *  P1  REPRESENTS THE NUMBER OF THE INPUT CARDS OF THE JOB.
        *
        *
        *  P2  IDENTIFIES THE CARD READER THAT IS READING A JOB
        *      (P2 TAKES ONE OF THE VALUES 3,4,5,6,7 FOR ONE OF THE CARD
        *      READERS 1,2,3,4,5 RESPECTIVELY).
        *
        *
        *  P3  IS USED BY THE INPUT SPOOLER AND TAKES ONE OF THE VALUES 1,2,3,4,
        *      5 IF THE JOB COMES FROM ONE OF THE INPUT STATIONS 1,2,3,4,5
        *      RESPECTIVELY.
        *
        *
        *  P4  TAKES ONE OF THE VALUES 1,2,8 IF INPUT DRUM,INPUT DISK,OUTPUT
        *      DISK IS REQUESTED BY A PROCESS.
        *
        *
        *  P5  HOLDS THE CLASS IN WHICH THE JOB BELONGS AND TAKES ONE OF THE
        *      VALUES 1,2,3,4,5,6,7 FOR ONE OF THE CLASSES A,B,C,D,E,F,G
        *      RESPECTIVELY.
        *
        *
        *  P6  HOLDS THE EXTERNAL PRIORITY (PRIORITY ASSIGNED BY THE USER) FOR
        *      THE JOB.
        *
        *
        *  P7  IS THE MAIN MEMORY REQUESTED BY THE USER.
        *
        *
        *  P8  IS THE TOTAL CPU TIME USED BY THE JOB.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  P9  IS THE TOTAL NUMBER OF OUTPUT LINES FOR A JOB EXCLUDING THE
        *      SOURCE LISTING.
        *
        *  P10  IS AN AUXILIARY PARAMETER USED BY THE INPUT SPOOLER
        *
        *  P11 IS THE TOTAL NUMBER OF DATA CARDS FOR A JOB.
        *
        *  P12 IS THE TOTAL NUMBER OF OUTPUT DATA BLOCKS FOR A JOB.
        *      OUTPUT DATA BLOCKS CONTAIN INFORMATION THAT IS TO BE PRINTED AND
        *      GENERATED DURING EXECUTION OF THE JOB (EACH DATA BLOCK IS 800
        *      BYTES).
        *
        *  P13 IS THE TOTAL NUMBER OF INPUT STATEMENTS OF A JOB.
        *
        *  P14 IS THE TOTAL NUMBER OF INPUT DATA BLOCKS FOR A JOB.
        *      INPUT DATA BLOCKS CONTAIN INFORMATION AND ARE GENERATED DURING
        *      EXECUTION OF A CARD READER.
        *
        *  P15 IS THE TOTAL NUMBER OF OUTPUT LINES WHICH INCLUDES THE LISTING
        *      OF THE JOB.
        *
        *  P16  MEAN OF I/O INTERREQUEST INTERVALS.
        *
        *  P17 SAVES THE ADDRESS OF THE LOCATION TO WHICH THE PROCESS WILL BE
        *    RETURNED FROM THE CPU SCHEDULER.
        *
        *  P18 IS THE TIME THAT THE PROCESS WILL HOLD THE CPU.
        *
        *  P19 SAVE THE PRIORITY OF A PROCESS WHICH ENTERS THE CPU
        *      SCHEDULER.
        *
        *  P20 SAVES THE ADDRESS OF THE LOCATION TO WHICH THE PROCESS WILL
        *      BE RETURNED FROM THE I/O DEVICE MANAGEMENT.
        *
        *  P22 IS USED BY THE I/O DEVICE MANAGEMENT AND TAKES THE VALUE 23 IF A
        *      DRUM IS REQUESTED, 26 IF A DISK IS REQUESTED.
        *
        *  P23  IS THE TIME TAKEN BY THE DIRECT ACCESS STORAGE DEVICE (DISK OR
        *      DRUM) TO PERFORM ONE REVOLUTION.
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *   P24 IS THE TIME TAKEN BY THE CHANNEL TO TRANSFER A BLOCK OF
        *       INFORMATION. THIS TIME DEPENDS ON THE SIZE OF THE BLOCK.
        *
        *   P25 SAVES THE PRIORITY OF A PROCESS WHICH ENTERS THE I/O DEVICE
        *       MANAGEMENT.
        *
        *   P31 IDENTIFIES THE LINE PRINTER REQUESTED BY THE INPUT STATION.
        *       IT CAN TAKE ONE OF THE VALUES 10,11,12,13,14.
        *
        *   P32 TAKES THE VALUE 9 IF THE AUXILIARY LINE PRINTER IS REQUESTED, 0
        *       IF  IT IS NOT REQUESTED.
        *
        *   P33 IDENTIFIES THE COMPILER REQUESTED BY THE JOB. IT CAN TAKE ONE
        *       OF THE VALUES 17,18,19,20,21.
        *
        *   P35 SAVES THE ADDRESS OF THE LOCATION TO WHICH THE PROCESS WILL BE
        *       RETURNED FROM THE RELOCATABLE LOADER.
        *
        *   P36  IS THE TIME AT WHICH THE JOB CAPTURES THE CARD READER.
        *
        *   P37  IS THE TIME AT WHICH THE JOB LEAVES THE CARD READER.
        *
        *   P38  IS THE TIME AT WHICH THE JOB ENTERS THE INPUT QUEUE.
        *
        *   P39  IS THE TIME THAT THE JOB HAD TO WAIT IN THE INPUT QUEUE.
        *
        *   P40  IS THE TIME AT WHICH A PARTITION IS ALLOCATED TO THE JOB.
        *
        *   P41  IS THE NUMBER OF THE PARTITION THAT HAS BEEN ALLOCATED TO
        *        THE JOB.
        *
        *   P46  IS THE TURNAROUND TIME OF THE JOB.
        *
        *   P47  IS THE TIME AT WHICH THE JOB ARRIVES AT THE SYSTEM.
        *
        *   P48  IS THE TIME AT WHICH THE JOB CAPTURES THE COMPILER.
        *
        *   P49  IS THE TIME TAKEN BY THE COMPILER TO COMPILE A JOB.
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *  PE0  IS THE TIME AT WHICH THE JOB CAPTURES THE LINE PRINTER.
        *
        *  PE1  IS THE TIME TAKEN BY THE LINE PRINTER TO PRINT THE JOB.
        *
        *  PE2 IDENTIFIES THE NUMBER OF THE JOB IN THE AUXILIARY REPORT
        *      GENERATION.
        *
        *  PE3 IS THE TIME AT WHICH THE JOB ENTERS THE OUTPUT QUEUE.
        *
        *  PE4 IS THE TIME THAT A JOB HAS TO WAIT IN THE OUTPUT QUEUE.
        *
        *
        *
        *      EQUIVALENT STATEMENTS
        *
        *  THESE ARE DECLARED EXPLICITY TO AVOID INTERFERENCE WITH THE BLOCKS
        *
        CHANQ  EQU         15,Q
        CHAN   EQU         15,F
        WAIT1  EQU         15,C
        READY  EQU         16,Q
        CPU    EQU         16,F
        AVG1   EQU         5,X
        AVG2   EQU         6,X
        TIME   EQU         7,X
        LOADQ  EQU         25,Q
        LOADF  EQU         25,F
        SPES   EQU         28,C
        *
        *
        *       SAVEVALUE   INITIALIZATIONS
        *
        *
        *  AVERAGE JOBS ARRIVAL TIME AT INPUT STATION 1.
        *
               INITIAL     XF$AVG1,15000
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  AVERAGE JOBS ARRIVAL TIME AT INPUT STATION 2.
        *
               INITIAL    XF$AVG2,120000
        *
        *
        *  THE TIMER TRANSACTION ARRIVES AT TIME XF$TIME.
        *
               INITIAL    XF$TIME,1151400
        *
        *
        * JOBS WHICH REQUEST CPU TIME OF MORE THAN 100000MS WILL BE SCHEDULED
        *  AT THE END OF THE DAY.
        *
               INITIAL    XF3,100000
        *
        *
        *  JOBS WHICH REQUEST MAIN MEMORY OF MORE THAN 40K WILL BE SCHEDULED AT
        *  THE END OF THE DAY.
        *
               INITIAL    XF4,40
        *
        *
        *  XF8 IS THE NUMBER OF PARTITIONS IN THE MAIN MEMORY.
        *
               INITIAL    XF8,7
        *
        *
        *  XF20 IS THE MAIN MEMORY AVAILABLE TO THE USERS.
        *
               INITIAL    XF20,180
        *
        *
        *  XF30 IS THE ROW INDEX OF THE MATRIX 2. MATRIX 2 IS USED TO STORE THE
        *  FRAGMENTATION OF THE MAIN MEMORY.
        *
               INITIAL    XF30,1
        *
        *
        *  XF31 IS THE COLUMN INDEX OF THE MATRIX 2.
        *
               INITIAL    XF31,1
        *
        *
        *
        *       MATRIX DEFINITION STATEMENT
        *
        *
        *
1.α     1      MATRIX     MX,7,5
        *
        *
2.α     2      MATRIX     MX,20,5
        *
```

```
      BLOCK
      NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
              *
              *
              *  MATRIX SAVEVALUE INITIALIZATIONS
              *
              *
              *
                     INITIAL    MX1(1,1),1/MX1(1,2),2/MX1(1,3),3/MX1(1,5),10
              *
                     INITIAL    MX1(2,1),1/MX1(2,2),2/MX1(2,3),3/MX1(2,5),20
              *
                     INITIAL    MX1(3,1),3/MX1(3,2),2/MX1(3,3),1/MX1(3,5),23
              *
                     INITIAL    MX1(4,1),3/MX1(4,2),4/MX1(4,3),5/MX1(4,5),25
                     INITIAL    MX1(4,4),7
              *
                     INITIAL    MX1(5,1),4/MX1(5,2),3/MX1(5,3),5/MX1(5,5),30
                     INITIAL    MX1(5,4),7
              *
                     INITIAL    MX1(6,1),6/MX1(6,2),7/MX1(6,3),4/MX1(6,4),3
              *
                     INITIAL    MX1(6,5),35
              *
                     INITIAL    MX1(7,1),6/MX1(7,2),7/MX1(7,5),40
              *
              *
              *
              *  THE MAIN MEMORY HAS 7 PARTITIONS.
              *
                     STORAGE    S$MEMOR,7
              *
              *
              *
              *         VARIABLE DEFINITIONS
              *
              *
              *
              *
              *  V1 IS ASSIGNED THE NUMBER OF INPUT DATA CARDS.
  1.b         1      VARIABLE   (PF1*FN*PF10)/100
              *
              *
              *  V2 IS ASSIGNED THE NUMBER OF OUTPUT BLOCKS.
  2.b         2      VARIABLE   PF9/10+1
              *
              *
              *
              *  V4 IS ASSIGNED THE NUMBER OF INPUT STATEMENTS.
              *
  4.b         4      VARIABLE   PF1-PF11
              *
              *
              *
```

```
  BLOCK
  NUMBER   *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
           *   V5 IS ASSIGNED THE NUMBER OF INPUT BLOCKS.
           *
  5.b      5      VARIABLE   PF11/10+1
           *
           *
           *   V6 IS ASSIGNED THE NUMBER OF OUTPUT LINES WHICH INCLUDES THE SOURCE
           *   LISTING OF THE PROGRAM.
           *
  6.b      6      VARIABLE   PF9+PF13+50
           *
           *
           *   V7 IS THE MEAN OF I/O INTEREQUEST INTERVALS.
           *
  7.b      7      VARIABLE   PF8/(PF12+PF14)
           *
           *
           *   V8 IS THE MAIN MEMORY THAT IS NOT USED IN THE PARTITION XF9.
           *
  8.b      8      VARIABLE   XF11-PF7
           *
           *
           *   V9 IS THE FRAGMENTATION OF THE MAIN MEMORY EXPRESSED IN PARTS PER
           *   THOUSAND.
           *
  9.b      9      VARIABLE   (XF13+XF14+XF15+XF16+XF17+XF18+XF19)*1000/XF20
           *
           *
           *   V10 IS ASSIGNED THE NUMBER OF INPUT BLOCK TRANSFERS THAT THE COMPILER
           *   NEEDS IN ORDER TO COMPILE THE PROGRAM.
           *
  10.b     10     VARIABLE   PF13/50+1
           *
           *
           *   V12 IS THE NUMBER OF STANDARD JOBS WHICH HAVE BEEN ENTERED INTO THE
           *   SYSTEM.
           *
  12.b     12     VARIABLE   N$WAIT-CH$WAIT1
           *
           *
  14.b     14     VARIABLE   MX1(1,5)+MX1(2,5)+MX1(3,5)+MX1(4,5)+MX1(5,5)+MX1(6,5)
           *
           *
           *   V13 IS THE TOTAL CORE SIZE AVAILABLE TO THE USERS.
           *
  13.b     13     VARIABLE   V14+MX1(7,5)
           *
           *
```

```
BLOCK
NUMBER   *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J         COMMENTS
         *
         *
         *           BOOLEAN VARIABLE DEFINITIONS
         *
         *
         *
  1       1     BVARIABLE  PF20≠E≠1+PF20≠E≠11
         *
         *
  2       2     BVARIABLE  XH*PF3≠E≠10+PF1≠E≠1
         *
         *
  3       3     BVARIABLE  CH9≠LE≠5*PF32≠E≠9
         *
         *
  4       4     BVARIABLE  WSWAIT≠E≠0*F3≠E≠0*F4≠E≠0
         *
         *
  5       5     BVARIABLE  PF30≠E≠1*PF29≠NE≠0
         *
         *
         *                TABLE DEFINITIONS
         *
         *
         *  TABLE 1 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 1
         *  WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  1       1     TABLE      PF42,0,2,W42
         *
         *  TABLE 2 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 2
         *  WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  2       2     TABLE      PF42,0,2,W42
         *
         *  TABLE 3 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 3
         *  WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  3       3     TABLE      PF42,0,2,W42
         *
         *  TABLE 4 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 4
         *  WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  4       4     TABLE      PF42,0,2,W42
         *
         *  TABLE 5 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 5
         *  WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
```

99

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
  5      5        TABLE      PF42,0,2,W42
         *
         *   TABLE 6 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 6
         *   WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  6      6        TABLE      PF42,0,2,W42
         *
         *   TABLE 7 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 7
         *   WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
         *
  7      7        TABLE      PF42,0,2,W42
         *
         *   TABLE 8 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 1
         *   (IT IS NOT WEIGHTED).
         *
  8      8        TABLE      PF42,0,2,42
         *
         *   TABLE 9 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 2
         *   (IT IS NOT WEIGHTED).
         *
  9      9        TABLE      PF42,0,2,42
         *
         *   TABLE 10 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 3
         *   (IT IS NOT WEIGHTED).
         *
 10     10        TABLE      PF42,0,2,42
         *
         *   TABLE 11 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 4
         *   (IT IS NOT WEIGHTED).
         *
 11     11        TABLE      PF42,0,2,42
         *
         *   TABLE 12 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 5
         *   (IT IS NOT WEIGHTED).
         *
 12     12        TABLE      PF42,0,2,42
         *
         *   TABLE 13 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 6
         *   (IT IS NOT WEIGHTED).
         *
 13     13        TABLE      PF42,0,2,42
         *
         *   TABLE 14 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 7
         *   (IT IS NOT WEIGHTED).
         *
 14     14        TABLE      PF42,0,2,42
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *
        *
        *              FUNCTION DEFINITIONS
        *
        *
        *
        *
        *
        *  FUNCTION 1 RETURNS THE CLASS FOR A JOB ARRIVING AT INPUT STATION 1.
        *
  1      1       FUNCTION    FN1,D7
  1     .35,1/.59,2/.81,3/.89,4/.96,5/.98,6/1.,7
        *
        *
        *  FUNCTION 2 RETURNS THE CLASS FOR A JOB ARRIVING AT INPUT STATION 2.
        *
  2      2       FUNCTION    FN2,D7
  2     .29,7/.54,6/.71,5/.81,4/.87,3/.92,2/1.,1
        *
        *
        *  EXTERNAL PRIORITY OF THE JOB ASSIGNED BY THE USER.
        *
  3      3       FUNCTION    FN3,D5
  3     .40,4/.75,3/.94,2/.97,1/1.,0
        *
        *
        *  MAIN MEMORY USED BY THE JOB (S).
        *
  4      4       FUNCTION    FN4,D4
  4     0.0,8/.40,12/.75,16/1.,22
        *
        *
        *  MAIN MEMORY USED BY THE JOB (M).
        *
  5      5       FUNCTION    FN5,D3
  5     0.0,22/.65,26/1.,32
        *
        *
        *  MAIN MEMORY USED BY THE JOB (L).
        *
  6      6       FUNCTION    FN6,D3
  6     0.0,32/.70,36/1.,42
        *
        *
        *  CPU TIME USED BY THE JOB (S1).
        *
  7      7       FUNCTION    FN7,D1?
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
   7           0.0,100/.05,200/.11,300/.19,400/.29,500/.44,600/.64,700/.89,800/.95,900
   7           1.,1001
               *
               *
               *
               *  CPU TIME USED BY THE JOB (S2).
               *
   8           8       FUNCTION   FN8,C11
   8           0.0,1000/.05,3000/.11,5000/.24,7000/.44,9000/.69,11000/.89,13000
   8           .94,15000/.97,17000/1.,20001
               *
               *
               *  CPU TIME USED BY THE JOB (M).
               *
   9           9       FUNCTION   FN1,C11
   9           0.0,20001/.02,22000/.06,24000/.11,26000/.41,28000/.61,30000/.76,32000
   9           .87,34000/.90,36000/.94,38000/1.,40001
               *
               *
               *  CPU TIME USED BY THE JOB (L).
               *
  10           10      FUNCTION   FN2,C13
  10           0.0,40001/.02,45000/.05,50000/.10,55000/.19,60000/.34,65000/.48,70000
  10           .51,75000/.79,80000/.87,85000/.93,90000/.97,95000/1.,100000
               *
               *
               *  NUMBER OF OUTPUT LINES (S1).
               *
  11           11      FUNCTION   FN3,C6
  11           0.0,0/.15,10/.50,20/.75,30/.95,40/1.,51
               *
               *
               *  NUMBER OF OUTPUT LINES (S2).
               *
  12           12      FUNCTION   FN4,C7
  12           0.0,10/.02,20/.10,30/.25,40/.50,50/.95,60/1.,71
               *
               *
               *  NUMBER OF OUTPUT LINES (M)
               *
  13           13      FUNCTION   FN5,C10
  13           0.0,40/.02,50/.07,60/.15,70/.25,80/.45,90/.70,100/.85,110/.93,120/1.,131
               *
               *
               *  NUMBER OF OUTPUT LINES (L1)
               *
  14           14      FUNCTION   FN6,C11
  14           0.0,100/.02,110/.07,120/.12,130/.20,140/.30,150/.55,160/.83,170
  14           .93,180/.98,190/1.,201
```

```
 BLOCK
 NUMBER   *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J         COMMENTS
          *
          *
          *
          *  NUMBER OF OUTPUT LINES (L2)
          *
  15      15      FUNCTION   RN7,D13
  15      0.0,15/.02,160/.05,170/.09,180/.16,190/.24,200/.39,210/.59,220
  15      .74,230/.84,240/.92,250/.97,260/1.,271
          *
          *
          *  NUMBER OF OUTPUT LINES (L3)
          *
  16      16      FUNCTION   RN8,D9
  16      0.0,180/.08,190/.13,200/.23,210/.42,220/.67,230/.87,240/.97,250/1.,281
          *
          *
          *  NUMBER OF DATA CARDS GIVEN AS THE PER CENTAGE OF INPUT CARDS (S1).
          *
  17      17      FUNCTION   RN2,D9
  17      0.0,0/.15,5/.20,10/.42,15/.67,20/.82,25/.90,30/.95,35/1.,40
          *
          *
          *  NUMBER OF DATA CARDS GIVEN AS THE PER CENTAGE OF INPUT CARDS (M).
          *
  18      18      FUNCTION   RN3,D6
  18      0.0,40/.20,45/.55,50/.80,55/.92,60/1.,65
          *
          *
          *  NUMBER OF DATA CARDS GIVEN AS THE PER CENTAGE OF INPUT CARDS (L).
          *
  19      19      FUNCTION   RN4,D5
  19      0.0,65/.15,70/.38,75/.68,80/1.,85
          *
          *
          *  FUNCTION 20 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 5
          *  AND IS USED BY THE CPU SCHEDULER.
          *
  20      20      FUNCTION   P5,L7
  20      1,CLASA/2,CLASB/3,CLASC/4,CLASD/5,CLASE/6,CLASF/7,CLASG
          *
          *
          *  FUNCTION 21 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 17
          *  AND IS USED BY THE PROCESS SCHEDULER.
          *
  21      21      FUNCTION   P17,L14
  21      1,AAA1/2,AAA2/3,AAA3/4,AAA4/5,AAA5/6,AAA6/7,AAA7/8,AAA8/9,AAA9/10,AAA10
  21      11,AAA11/12,AAA12/13,AAA13/14,AAA14
          *
          *
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *
         *
         *  FUNCTION 22 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 20
         *  AND IS USED BY THE I/O DEVICE MANAGEMENT.
         *
  22     22     FUNCTION   PF20,L15
  22     1,BBB1/2,BBB2/3,BBB3/4,BBB4/5,BBB5/6,BBB6/7,BBB7/8,BBB8/9,BBB9
  22     10,BBB10/11,BBB11/12,BBB12/13,BBB13/14,BBB14/15,BBB15
         *
         *
         *  FUNCTION 23 RETURNS THE ACCESS TIME FOR THE DRUM.
         *
  23     23     FUNCTION   RN2,C2
  23     0.0,0/1.,11
         *
         *
         *  DUMMY FUNCTION WHICH RETURNS THE VALUE ZERO.
         *
  24     24     FUNCTION   1,D2
  24     1,0/2,0
         *
         *
         *  FUNCTION 25 RETURNS THE TIME FOR THE HEAD OF THE DISK TO MOVE INTO
         *  THE REQUIRED POSITION.
         *
  25     25     FUNCTION   RN3,C2
  25     0.0,5/1.,56
         *
         *
         *  FUNCTION 26 RETURNS THE TIME REQUIRED FOR THE BEGINING OF THE TRACK
         *  WILL TAKE TO COME UNDER THE HEAD OF THE DISK.
         *
  26     26     FUNCTION   RN4,C2
  26     0.0,0/1.,21
         *
         *
         *
  27     27     FUNCTION   XF9,L7
  27     1,13/2,14/3,15/4,16/5,17/6,18/7,19
         *
         *
         *
  28     28     FUNCTION   XF9,L7
  28     1,37/2,36/3,35/4,34/5,33/6,32/7,31
         *
         *
         *  FUNCTION 29 IS USED TO CHANGE THE LEVEL OF PRIORITIES.
         *
```

```
 BLOCK
 NUMBER   *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
  29      29       FUNCTION   PF6,D5
  29      0,4/1,3/2,2/3,1/4,0
          *
          *
          *
          *  FUNCTION 30 RETURNS THE TIME REQUIRED BY THE COMPILER TO COMPILE A
          *  BLOCK OF SOURCE CODE.
          *
  30      30       FUNCTION   RN4,C2
  30      .0,100/1.,300
          *
          *
          *
          *  FUNCTION 31 IS USED BY THE INPUT SPOOLER..
          *
  31      31       FUNCTION   PF3,D5
  31      1,16/2,17/33,18/34,19/35,20
          *
          *
          *
          *
          *
          *  FUNCTION SHCRS PROVIDES A CORRESPONDENCE BETWEEN THE CARD READERS
          *  AND THEIR RESPECTIVE CONTROL SWITCHES.
          *
  32      SHCRS    FUNCTION   PF3,D5
  32      1,28/2,29/33,30/34,31/35,32
          *
          *
          *
          *  FUNCTION LOGIC IS USED BY THE JOB SCHEDULER TO RETURN VALUES FOR THE
          *  LOGIC SWITCHES.
          *
  33      LOGIC    FUNCTION   XF9,L7
  33      1,8/2,9/3,10/4,11/5,12/6,13/7,14
          *
          *
          *
          *  FUNCTION NCAR1 RETURNS THE NUMBER OF INPUT CARDS FOR JOBS THAT
          *  ARRIVE AT INPUT STATION 1.
          *
  34      NCAR1    FUNCTION   RN7,C7
  34      0.0,10/.01,50/.43,100/.75,150/.92,200/.97,250/1.,301
          *
          *
          *
          *  FUNCTION NCAR2 RETURNS THE NUMBER OF INPUT CARDS FOR JOBS THAT
          *  ARRIVE AT INPUT STATION 2.
          *
  35      NCAR2    FUNCTION   RN8,C8
  35      0.0,50/.03,100/.08,150/.16,200/.30,250/.65,300/.95,350/1.,400
          *
          *
          *
```

```
 BLOCK
 NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *  FUNCTION MORSH IS USED BY THE JOBS IN CLASS C.
         *
  36     MORSH  FUNCTION    RN1,D2
  36     .60,4/1.,5
         *
         *
         *
         *  FUNCTION LORMM IS USED BY JOBS IN CLASS G.
         *
  37     LORMM  FUNCTION    RN2,D2
  37     .65,6/1.,5
         *
         *
         *
         *  FUNCTION SOURC IS USED BY THE COMPILER.
         *
  38     SOURC  FUNCTION    RN1,D6
  38     .05,2/.2,4/.3,6/.55,8/.85,10/1.,12
         *
         *
         *
  39     PLOCK  FUNCTION    PF33,D5
  39     17,20/18,10/19,50/20,150/21,100
         *
         *
         *
         *  FUNCTION ADDRS RETURNS A LOCATION BASED UPON THE VALUE OF PARAMETER
         *  3F AND IS USED BY THE LOADER.
         *
  40     ADDRS  FUNCTION    PF35,L2
  40     1,NEXBL/2,XXX2
         *
         *
         *
         *  FUNCTION COMP RETURNS THE NAME OF THE COMPILER REQUESTED BY A JOB.
         *
  41     COMP   FUNCTION    RN5,D5
  41     .50,17/.52,18/.92,19/.97,20/1.,21
         *
         *
         *
         *  FUNCTION TABL IS USED TO GIVE THE NAMES TO THE TABLES 8,9,10,11,12,
         *  13,14.
         *
  42     TABL   FUNCTION    PF41,L7
  42     1,8/2,9/3,10/4,11/5,12/6,13/7,14
         *
         *
         *
         *  FUNCTION COPSP PROVIDES A CORRESPONDENCE BETWEEN THE COMPILERS AND
         *  THEIR RESPECTIVE CONTROL SWITCHES.
         *
  43     COPSP  FUNCTION    PF33,D5
  43     17,23/18,24/19,25/20,25/21,27
         *
```

106

```
     BLOCK
     NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
             *
             *
             *   FUNCTION LINPE PROVIDES A CORRESPONDENCE BETWEEN THE LINE PRINTERS
             *   AND THEIR RESPECTIVE CONTROL SWITCHES.
             *
 44          LINPE  FUNCTION   PF31,D6
 44          9,39/10,40/11,41/12,42/13,43/14,44
             *
             *
             *
             *   EXPONENTIAL DISTRIBUTION FUNCTION.
             *
 45          XPDIS  FUNCTION   RN8,C24
 45          0.0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
 45          0.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99
 45          0.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
             *
             *
             *
             *
             *
             *
             *
             *
             ***********************************************
             *             INPUT STATION 2.
             ***********************************************
             *
             *
             *
             *
             *   JOBS ARRIVE AT INPUT STATION 2.
             *
         1         GENERATE    X$AVG2,FN$XPDIS,,,,54PF,4PH
             *
             *
             *
             *   IF IT IS TIME TO CLOSE THE COMPUTING CENTER, DO NOT ALLOW ANY FURTHER
             *   JOBS TO COME INTO THE SYSTEM.
             *
         2         GATE LR  , 22
             *
             *
             *
             *   ASSIGN THE NUMBER OF INPUT CARDS.
             *
         3         ASSIGN      1,FN$NCAR2,PF
             *
             *
             *
             *   ASSIGN AN IDENTIFICATION NUMBER TO PF2 AND PF3.
             *
         4         ASSIGN      2,4,PF
             *
             *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
  5               ASSIGN     3,2,PF
        *
        *
        *
        *  ASSIGN LINE PRINTER 1 FOR PRINTING.
        *
  6               ASSIGN     31,10,PF
        *
        *
        *
        *  ASSIGN THE AUXILIARY LINE PRINTER IF REQUESTED.
        *
  7               ASSIGN     32,9,PF
        *
        *
        *
        *
        *
        *
        *  GO TO WAIT FOR THE CARD READER.
        *
  8               TRANSFER   ,WAIT
        *
        *
        *
        *
        ***********************************************
        *      INPUT STATION 1.
        ***********************************************
        *
        *
        *
        *  JOBS ARRIVE AT INPUT STATION 1.
        *
  9               GENERATE   X$AV31,FN$XPDIS,,,,54PF,4PH
        *
        *
        *
        *  IF IT IS TIME TO CLOSE THE COMPUTING CENTER, DO NOT ALLOW ANY FURTHER
        *  JOBS TO COME INTO THE SYSTEM.
        *
 10               GATE LR    22
        *
        *
        *
        *  ASSIGN THE NUMBER OF INPUT CARDS.
        *
 11               ASSIGN     1,FN$NCAR1,PF
        *
        *
        *
        *  ASSIGN AN IDENTIFICATION NUMBER TO PF2 AND PF3.
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
  12             ASSIGN     2,3,PF
        *
        *
        *
  13             ASSIGN     3,1,PF
        *
        *
        *
        *  ASSIGN LINE PRINTER 1 FOR PRINTING.
        *
  14             ASSIGN     31,10,PF
        *
        *
        *
        *  ASSIGN THE AUXILIARY LINE PRINTER IF REQUESTED.
        *
  15             ASSIGN     32,9,PF
        *
        *
        *
        ***************************************************
        *       THE CARD READERS.
        ***************************************************
        *
        *
        *
        *  WAIT FOR THE CARD READER.
        *
  16    WAIT     QUEUE      PF2
        *
        *
        *
  17             SAVEVALUE  25+,1,XF
        *
        *
        *
        *  ASSIGN TO PF52 THE ID OF THE JOB.
        *
  18             ASSIGN     52,XF25,PF
        *
        *
        *
        *  MARK THE TIME AT WHICH THE JOB ARRIVES.
        *
  19             MARK       47PF
        *
        *
        *
        *  CAPTURE THE CARD READER.
        *
  20             SEIZE      PF2
        *
        *
        *
        *  MARK THE TIME AT WHICH THE JOB CAPTURED THE CARD READER.
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
         *
   21           MARK       36PF
         *
         *
         *
         *  SAVE THE NUMBER OF INPUT CARDS.
         *
   22           ASSIGN     26,PF1,PF
         *
         *
         *
         *  INDICATES THAT WE ARE READING THE JOB CARD.
         *
   23           LOGIC S    PF2
         *
         *
         *
         *  LEAVE THE QUEUE.
         *
   24           DEPART     PF2
         *
         *
         *
         *  THE CARD READER HAS PRIORITY 30.
         *
   25    AGAIN  PRIORITY   30
         *
         *
         *
         *  TIME TAKEN BY THE CARD READER TO READ A CARD.
         *
   26           ADVANCE    60,1
         *
         *
         *
         *  IF LOGIC SWITCH IS SET, THEN WAIT SINCE THE INPUT DRUM IS FULL.
         *
   27    BACK   GATE LR    FN$SWCRS
         *
         *
         *
         *  GOTO INPUT SPOOLER.
         *
   28           TRANSFER   ,INPSP
         *
         *
         *
         *  RETURN FROM INPUT SPOOLER.
         *
         *  IF THERE ARE MORE CARDS LEFT GO BACK TO READ THE NEXT CARD.
         *
   29    CARDF  LOOP       1PF,AGAIN
         *
         *
         *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *  LEAVE THE CARD READER.
         *
  30            RELEASE     PF2
         *
         *
         *
         *  ASSIGN TO PF 37 THE TIME TAKEN BY THE CARD READER TO READ A JOB.
         *
  31            ASSIGN      37,MP36PF,PF
         *
         *
         *
         *  MARK THE TIME AT WHICH THE JOB ENTERS THE INPUT QUEUE.
         *
  32            MARK        38PF
         *
         *
         *
         *  IF THE JOB HAS REQUESTED MORE MEMORY OR CPU TIME THAN IS  ALLOWED
         *  SEND THE JOB TO THE CHAIN WAIT1. THESE JOBS WILL BE SCHEDULED
         *  AT THE END OF THE DAY.
         *
  33            TEST NE     PF10,0,FFF
         *
         *
         *
         *  OPEN THE GATE FOR THE JOB SCHEDULER TRANSACTION.
         *
  34            LOGIC R     15
         *
         *
         *
         *  XF2 IS A COUNTER THAT COUNTS THE NUMBER OF JOBS IN THE CHAINS 1,2,3,4
         *  5,6,7.
         *
  35            SAVEVALUE   2+,1,X
         *
         *
         *
         *  INSERT THE JOB INTO THE CHAIN PF5.
         *
  36            LINK        PF5,5PF
         *
         *
         *
         *  THIS CHAIN IS FOR NON STANDARD JOBS.
         *
  37     FFF    LINK        WAIT1,6PF
         *
         *
         *
         *
         *
         ****************************************************
         *          INPUT SPOOLER.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *************************************************
        *
        *  THE INPUT SPOOLER HAS PRIORITY 60.
        *
  38    INPSP  PRIORITY   60
        *
        *
        *  IF AT THIS TIME WE ARE TRANSFERING THE CONTENTS OF BUFFER OF THE
        *  INPUT SPOOLER I TO THE INPUT DISK, THE TRANSFER OF THE NEXT CARD
        *  READ BY CARD READER I TO THE BUFFER OF THE INPUT SPOOLER IS DELAYED.
        *
  39           GATE LR    FN31
        *
        *
        *
        *  CREATE AN I/O PROCESS. ITS TASK IS TO TRANSFER THE CONTENTS OF THE
        *  BUFFER OF THE CARD READER INTO THE WORK AREA OF THE INPUT SPOOLER IN
        *  ORDER TO EXAMINE THE CONTENTS OF THE CARD.
        *
        *  SAVE RETURN ADDRESS.
        *
  40           ASSIGN     20,1,PF
        *
        *
        *
        *  GOTO I/O DEVICE MANAGEMENT.
        *
  41           TRANSFER   ,DMAN
        *
        *
        *
        *  REQUEST THE CPU TO EXAMINE THE CONTENTS OF THE CARD.
        *
        *  SAVE RETURN ADDRESS.
        *
  42    BBB1   ASSIGN     17,1,PF
        *
        *
        *  PF18 IS THE TIME TAKEN BY THE INPUT SPOOLER TO EXAMINE THE
        *  CONTENTS OF THE CARD.
        *
  43           ASSIGN     18,1,PF
        *
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
  44           TRANSFER   ,PROSC
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER AND I/O DEVICE MANAGEMENT.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *  NOW EXAMINE TO SEE IF THE CARD READ IS THE JOB CARD.
        *
  45    AAA1   GATE LR    PF2,FIRSC
        *
        *
        *
  46           ASSIGN     4,2,PF
        *
        *
        *
        *  SIGNAL THE CARD READER TO READ THE NEXT CARD AND AT THE SAME TIME IF
        *  THE BUFFER OF THE INPUT SPOOLER IS FULL TRANSFER ITS CONTENTS TO THE
        *  INPUT DISK.
        *
  47           SPLIT      1,GAMA
        *
        *
        *
        *  SIGNAL TO THE CARD READER TO READ THE NEXT CARD.
        *
  48           TRANSFER   ,CARDR
        *
        *
        *
        *  INCREASE BY ONE THE CONTENTS OF THE BUFFER OF THE SPOOLER WHICH
        *  CORRESPONDS TO THE CARD READER UNTER EXAMINATION.
        *
  49    GAMA   SAVEVALUE  PF3+,1,XH
        *
        *
        *
        *  IF THIS IS THE LAST CARD OF THE CURRENT JOB OR IF THE BUFFER OF THE
        *  INPUT SPOOLER IS FULL GO TO QQQ.
        *
  50           TEST E     BV2,0,QQQ
        *
        *
        *
        *  DESTROY THE PROCESS.
        *
  51           TERMINATE
        *
        *
        *
        *  CREATE AN I/O PROCESS WHOSE TASK IS TO TRANSFER THE CONTENTS OF THE
        *  BUFFER OF THE INPUT SPOOLER INTO THE INPUT DISK.
        *  CLOSE THE DOOR UNTIL THE TRANSFER IS OVER.
        *
  52    QQQ    LOGIC S    FN31
        *
        *
        *
        *  SAVE RETURN ADDRESS.
        *
  53           ASSIGN     20,9,PF
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  THE TRANSFER TIME IS 4MS.
        *
  54           ASSIGN     24,4,PF
        *
        *
        *
  55           TRANSFER   ,DISK
        *
        *
        *
        *  RETURN FROM THE I/O DEVICE MANAGEMENT.
        *
        *  OPEN THE GATE SINCE THE TRANSFER IS OVER.
        *
  56    BB39   LOGIC R    FN31
        *
        *
        *
        *  NOW THE BUFFER OF THE INPUT SPOOLER IS EMPTY.
        *
  57           SAVEVALUE  PF3,0,XH
        *
        *
        *
        *  DESTROY THE PROCESS.
        *
  58           TERMINATE
        *
        *
        *
        *  TRANSFER TO FIRSC WHENEVER THE CARD UNDER EXAMINATION IS THE JOB
        *  CARD.
        *
  59    FIRSC  ASSIGN     4,1,PF
        *
        *
        *
        *  TEST TO SEE IF THE INPUT DISK IS FULL.
        *
  60           TEST G     XF1,400,NFULL
        *
        *
        *
        *  IF THE INPUT DISK IS FULL CLOSE THE GATE.
        *
  61           LOGIC S    FN3SHOES
        *
        *
        *
        *  RETURN TO THE CARD READER.
        *
  62           TRANSFER   ,BACK
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *  INCREASE THE ENTRIES IN THE INPUT DISK BY ONE.
        *
  63    NFULL  SAVEVALUE  1+,1,X
        *
        *
        *  PF5 IS THE CLASS OF THE JOB.
        *
  64           ASSIGN     5,FN*PF3,PF
        *
        *
        *
        *  PF6 IS THE PRIORITY OF THE JOB .THIS IS AN  EXTERNAL PRIORITY.
        *
  65           ASSIGN     6,FN3,PF
        *
        *
        *
        *  PF33 IS THE NAME OF THE COMPILER REQUESTED BY THE JOB.
        *
  66           ASSIGN     33,FN$COMP,PF
        *
        *
        *
        *  TRANSFER TO THE CLASS IN WHICH THE JOB BELONGS .
        *
  67           TRANSFER   FN,20
        *
        *
        *
        *  THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS A.
        *-------------------------------------------------------------------
        *  PF7 IS THE MAIN MEMORY USED BY THE JOB.
  68    CLASA  ASSIGN     7,FN4,PF
        *
        *
        *
        *  PF8 IS THE CPU TIME USED BY THE JOB.
        *
  69           ASSIGN     8,FN7,PF
        *
        *
        *
        *  PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
  70           ASSIGN     9,FN11,PF
        *
        *
        *
  71           ASSIGN     10,17,PF
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *
  72           TRANSFER    ,AAA
        *
        *
        *  THIS SECTION IS FOR THE JOBS WHICH BELONG TO CLASS B.
        *-----------------------------------------------------------------
        *
        *  PF7 IS THE MAIN MEMORY USED BY THE JOB.
  73    CLASB  ASSIGN      7,FN4,PF
        *
        *
        *
        *  PF8 IS THE CPU TIME USED BY THE JOB.
        *
  74           ASSIGN      8,FN3,PF
        *
        *
        *
        *  PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
  75           ASSIGN      9,FN15,PF
        *
        *
        *
  76           ASSIGN      10,19,PF
        *
        *
        *
  77           TRANSFER    ,AAA
        *
        *
        *
        *  THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS C.
        *-----------------------------------------------------------------
        *
  78    CLASC  ASSIGN      10,FNMMORSM,PF
        *
        *
        *
        *  PF7 IS THE MAIN MEMORY USED BY THE JOB.
        *
  79           ASSIGN      7,FN*PF10,PF
        *
        *
        *
        *  PF8 IS THE CPU TIME USED BY THE JOB.
        *
  80           ASSIGN      8,FN3,PF
        *
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *  PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
  81           ASSIGN      9,FN13,PF
        *
        *
        *
  82           ASSIGN      10,18,PF
        *
        *
        *
  83           TRANSFER    ,AAA
        *
        *
        *
        *  THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS D.
        *------------------------------------------------------------------
        *
        *  PF7 IS THE MAIN MEMORY USED BY THE JOB.
        *
  84    CLASD  ASSIGN      7,FN5,PF
        *
        *
        *
        *  PF8 IS THE CPU TIME USED BY THE JOB.
        *
  85           ASSIGN      8,FN8,PF
        *
        *
        *
        *  PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
  86           ASSIGN      9,FN14,PF
        *
        *
        *
  87           ASSIGN      10,13,PF
        *
        *
        *
  88           TRANSFER    ,AAA
        *
        *
        *
        *  THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS E.
        *------------------------------------------------------------------
        *
        *  PF7 IS THE MAIN MEMORY USED BY THE JOB.
        *
  89    CLASE  ASSIGN      7,FN5,PF
        *
        *
        *
        *  PF8 IS THE CPU TIME USED BY THE JOB.
        *
  90           ASSIGN      8,FN3,PF
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *
        *   PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
   91           ASSIGN     9,FN15,PF
        *
        *
        *
   92           ASSIGN     10,19,PF
        *
        *
        *
   93           TRANSFER   ,AAA
        *
        *
        *
        *   THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS F.
        *---------------------------------------------------------------
        *
        *   PF7 IS THE MAIN MEMORY USED BY THE JOB.
        *
   94   CLASF   ASSIGN     7,FN5,PF
        *
        *
        *
        *   PF8 IS THE CPU TIME USED BY THE JOB.
        *
   95           ASSIGN     8,FN10,PF
        *
        *
        *
        *   PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
        *
   96           ASSIGN     9,FN16,PF
        *
        *
        *
   97           ASSIGN     10,13,PF
        *
        *
        *
   98           TRANSFER   ,AAA
        *
        *
        *
        *   THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS G.
        *---------------------------------------------------------------
        *
        *
   99   CLASG   ASSIGN     10,FN$LORMM,PF
        *
        *
        *
        *   PF7 IS THE MAIN MEMORY USED BY THE JOB.
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J         COMMENTS
         *
  100           ASSIGN      7,FN*PF10,PF
         *
         *
         *
         *   PF8 IS THE CPU TIME USED BY THE JOB.
         *
  101           ASSIGN      8,FN10,PF
         *
         *
         *
         *   PF9 IS THE NUMBER OF OUTPUT RECORDS(OUTPUT LINES).
         *
  102           ASSIGN      9,FN12,PF
         *
         *
         *
  103           ASSIGN      10,17,PF
         *
         *
         *
         *   CHECK TO SEE IF THE JOB HAS REQUESTED MORE MEMORY THAN AVAILABLE TO
         *   THE USERS.
         *
  104    AAA    TEST LE     PF7,XF20,REGEC
         *
         *
         *
         *   PF11 IS THE NUMBER OF DATA CARDS.
         *
  105           ASSIGN      11,V1,PF
         *
         *
         *
         *   PF12 IS THE NUMBER OF THE OUTPUT BLOCKS
         *
  106           ASSIGN      12,V2,PF
         *
         *
         *
         *   PF13 IS THE NUMBER OF INPUT STATEMENTS.
         *
  107           ASSIGN      13,V4,PF
         *
         *
         *
         *   PF14 IS THE NUMBER OF INPUT BLOCKS.
         *
  108           ASSIGN      14,V5,PF
         *
         *
         *
         *   PF15 IS THE NUMBER OF THE OUTPUT LINES WHICH INCLUDES THE SOURCE
         *   LISTING OF THE PROGRAM.
         *
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
  109            ASSIGN     15,V6,PF
        *
        *
        *
        *  ASSIGN THE I/O INTEREQUEST INTERVALS.
        *
        *  PF16 IS THE MEAN OF THE I/O INTEREQUEST INTERVALS.
        *
  110            ASSIGN     16,V7,PF
        *
        *
        *
        *  IF THE MEAN IS LESS THAN ONE GOTO BBB
        *
  111            TEST GE    PF16,1,BBB
        *
        *
        *
        *  IF THE JOB HAS REQUESTED CPU TIME OF MORE THAN X3, THIS JOB WILL NOT
        *  BE SCHEDULED AT THIS TIME.
        *
  112   CCC      TEST LE    PF8,X3,EEE
        *
        *
        *
        *  IF THE JOB HAS REQUESTED MAIN MEMORY OF MORE THAN X20, THIS JOB WILL
        *  NOT BE SCHEDULED AT THIS TIME.
        *
  113            TEST LE    PF7,X4,EEE
        *
        *
        *
  114   DDD      LOGIC R    PF2
        *
        *
        *
        *  SIGNAL THE CARD READER TO READ THE NEXT CARD.
        *  AT THE SAME TIME REQUEST THE CHANNEL TO TRANSFER  THE INPUT QUEUE
        *  TABLE OF THE CLASS IN WHICH THE JOB BELONGS FROM THE INPUT DRUM TO
        *  THE WORK AREA OF THE INPUT SPOOLER.
        *
  115            SPLIT      1,GOB
        *
        *
        *
        *  SIGNAL THE CARD READER TO READ THE NEXT CARD.
        *
  116            TRANSFER   ,CARDR
        *
        *
        *
        *  CLOSE THE GATE UNTIL THE TRANSFER IS OVER.
        *
  117   GOB      LOGIC S    FN31
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  REQUEST THE CHANNEL.
        *
        *  SAVE RETURN ADDRESS.
        *
  118           ASSIGN     20,12,PF
        *
        *
        *
  119           ASSIGN     24,4,PF
        *
        *
        *
  120           TRANSFER   ,DRUM
        *
        *
        *
        *  REQUEST THE CPU TO ENTER THE CHARACTERISTICS OF THE JOB INTO THE
        *  INPUT QUEUE TABLE.
        *
        *  SAVE RETURN ADDRESS.
        *
  121   BB312   ASSIGN     17,11,PF
        *
        *
        *
        *  PF18 IS THE TIME TAKEN BY THE INPUT SPOOLER TO UPDATE THE TABLE.
        *
  122           ASSIGN     18,2,PF
        *
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
  123           TRANSFER   ,PROSC
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER.
        *
        *  REQUEST THE CHANNEL TO TRANSFER THE UPDATED TABLES BACK TO THE
        *  INPUT DRUM.
        *
        *  SAVE RETURN ADDRESS.
        *
  124   AAA11   ASSIGN     20,2,PF
        *
        *
        *
  125           ASSIGN     24,4,PF
        *
        *
        *
  126           TRANSFER   ,DRUM
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *
        *  THE TRANSFER IS OVER AND OPEN THE GATE.
        *
 127    BBB2   LOGIC R    FN31
        *
        *
        *
        *  DESTROY THE PROCESS.
        *
 128           TERMINATE
        *
        *
        *
        *  TRANSFER TO REGEC WHENEVER THE JOB IS REJECTED BY THE SYSTEM.
        *
 129    REGEC  ASSIGN     45,1,PF
        *
        *
        *
 130           SAVEVALUE  25-,1,XF
        *
        *
        *
 131           HELPA      REPOPT,PF45,XF25,PF7,XF20,PF2,PF3
        *
        *
        *
 132           TERMINATE
        *
        *
        *
        *  ASSIGN ALL THE I/O INTEREQUEST INTERVALS EQUAL TO ONE.
        *
 133    BBB    ASSIGN     16,1,PF
        *
        *
        *
 134           TRANSFER   ,CCC
        *
        *
        *
 135    EEE    ASSIGN     10,0,PF
        *
        *
        *
 136           TRANSFER   ,DDD
        *
        *
        *
        *-----------------------------------------------------------------
        *  TRANSFER TO THIS POINT WHENEVER THE DRUM IS REQUESTED.
        *-----------------------------------------------------------------
        *
```

```
BLOCK
NUMBER *LOC  OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  DIRECT I/O DEVICE MANAGEMENT TO SAMPLE FROM FUNCTION 24 IN ORDER
        *  TO OBTAIN THE TIME TAKEN BY THE ARM TO MOVE TO THE REQUIRED TRACK.
        *
  137   DRUM  ASSIGN     21,24,PF
        *
        *
        *
        *  DIRECT I/O DEVICE MANAGEMENT TO SAMPLE FROM FUNCTION 23 IN ORDER
        *  TO OBTAIN THE TIME REQUIRED FOR THE BEGINNING OF THE TRACK TO COME
        *  UNDER THE READ HEAD.
        *
  138         ASSIGN     22,23,PF
        *
        *
        *
        *  THE ROTATION SPEED OF THE DRUM IS 10 MS.
        *
  139         ASSIGN     23,10,PF
        *
        *
        *
        *  GOTO THE I/O DEVICE MANAGEMENT.
        *
  140         TRANSFER   ,DMAN
        *
        *
        *
        *-------------------------------------------------------------------
        *  TRANSFER TO THIS POINT WHENEVER ONE OF THE DISKS IS REQUESTED.
        *-------------------------------------------------------------------
        *
        *  DIRECT I/O DEVICE MANAGEMENT TO SAMPLE FROM FUNCTION 25 IN ORDER
        *  TO OBTAIN THE TIME TAKEN BY THE ARM TO MOVE TO THE REQUIRED TRACK.
        *
  141   DISK  ASSIGN     21,25,PF
        *
        *
        *
        *  DIRECT I/O DEVICE MANAGEMENT TO SAMPLE FROM FUNCTION 26 IN ORDER
        *  TO OBTAIN THE TIME REQUIRED FOR THE BEGINNING OF THE TRACK TO COME
        *  UNDER THE READ HEAD.
        *
  142         ASSIGN     22,26,PF
        *
        *
        *
        *  THE ROTATION SPEED OF THE DISK IS 20 MS.
        *
  143         ASSIGN     23,20,PF
        *
        *
        *
        *  GOTO THE I/O DEVICE MANAGEMENT.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
 144    *      TRANSFER   ,OMAN
        *
        *
        *
        ****************************************
        *      I/O DEVICE MANAGEMENT.
        ****************************************
        *
        *
        *  SAVE THE PRIORITY OF THE PROCESS.
        *
 145    OMAN   ASSIGN     25,PR,PF
        *
        *
        *  DEVICE MANAGEMENT HAS PRIORITY 60.
        *
 146           PRIORITY   60
        *
        *
        *
        *  REQUEST THE CPU TO EXAMINE THE MESSAGE.
        *
        *
        *  SAVE RETURN ADDRESS.
        *
 147           ASSIGN     17,2,PF
        *
        *
        *
        *  TIME REQUIRED TO EXAMINE THE MESSAGE.
        *
 148           ASSIGN     18,1,PF
        *
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
 149           TRANSFER   ,PROSD
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER.
        *
        *  RESTORE THE PRIORITY OF THE PROCESS.
        *
 150    AAA2   PRIORITY   PF25
        *
        *
        *
        *  IF THE TASK IS TO TRANSFER A RECORD FROM THE BUFFER OF THE CARD
        *  READER TO THE MEMORY OR FROM THE MEMORY TO THE BUFFER OF THE LINE
        *  PRINTER GO TO CARD.
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J          COMMENTS
          *
  151           TEST E     BV1,0,CARD
          *
          *
          *
          *  WAIT FOR THE DIRECT ACCESS STORAGE DEVICE (DASD).
          *
  152           QUEUE      PF4
          *
          *
          *
          *  OBTAIN THE DASD.
          *
  153           SEIZE      PF4
          *
          *
          *
          *  LEAVE THE QUEUE.
          *
  154           DEPART     PF4
          *
          *
          *
          *  WAIT FOR THE CHANNEL.
          *
  155           QUEUE      CHANQ
          *
          *
          *
          *  OBTAIN THE CHANNEL
          *
  156           SEIZE      CHAN
          *
          *
          *
          *  LEAVE THE QUEUE.
          *
  157           DEPART     CHANQ
          *
          *
          *
  158           PRIORITY   60
          *
          *
          *
          *  REQUEST THE CPU TO ISSUE THE START I/O COMMAND.
          *
          *  SAVE RETURN ADDRESS.
          *
  159           ASSIGN     17,3,PF
          *
          *
          *
  160           ASSIGN     18,3,PF
          *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
  161          TRANSFER   ,PROSC
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER.
        *
        *  RETURN THE CHANNEL
        *
  162   AAA3   RELEASE    CHAN
        *
        *
        *
        *  RESTORE THE PRIORITY OF THE PROCESS.
        *
  163          PRIORITY   PF25
        *
        *
        *
        *  IF THE PROCESS HAS REQUESTED A DISK THEN THIS REPRESENTS THE TIME
        *  TAKEN BY THE ARM TO MOVE TO THE REQUIRED TRACK.
        *
  164          ADVANCE    FN*PF21
        *
        *
        *
        *  THIS REPRESENTS THE TIME THAT THE BEGINNING OF THE TRACK MUST TAKE
        *  COME UNDER THE HEAD.
        *
  165          ADVANCE    FN*PF22
        *
        *
        *
        *  IF THE CHANNEL IS BUSY ,SEND THE TRANSACTION TO THE ADVANCE BLOCK
        *  AT LOCATION BUSY FOR ONE FULL REVOLUTION OF THE DRUM.
        *
  166   TRAG   TRANSFER   BOTH,ALIK,BUSY
        *
        *
        *
        *  OBTAIN THE CHANNEL
        *
  167   ALIK   SEIZE      CHAN
        *
        *
        *
  168          PRIORITY   80
        *
        *
        *
        *     REQUEST THE CPU TO TRANSFER THE BLOCK.
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *  SAVE RETURN ADDRESS.
        *
 169          ASSIGN     17,4,PF
        *
        *
        *  THIS IS THE TIME REQUIRED TO TRANSFER THE BLOCK.  THIS TIME DEPENDS
        *  UPON THE LENGTH OF THE BLOCK.
        *
 170          ASSIGN     18,PF24,PF
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
 171          TRANSFER   ,PROSC
        *
        *
        *
        *  THIS IS THE TIME TAKEN BY THE DIRECT ACCESS STORAGE DEVICE TO
        *  MAKE A FULL REVOLUTION.
        *
 172    BUSY   ADVANCE    PF23
        *
        *
        *
        *  RETURN AND ATTEMPT TO RECAPTURE THE CHANNEL.
        *
 173          TRANSFER   ,TRAG
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER.
        *
        *  RETURN THE DASD.
        *
 174    AAA4   RELEASE    PF4
        *
        *
        *
        *  RETURN THE CHANNEL.
        *
 175    AAA5   RELEASE    CHAN
        *
        *
        *
        *  RESTORE PRIORITY
        *
 176          PRIORITY   PF25
        *
        *
        *
        *  RETURN TO THE POINT WHERE THE REQUEST WAS MADE.
        *
 177          TRANSFER   FN,22
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *
        *
        *
        *  WAIT FOR THE CHANNEL.
        *
 178    CARD   QUEUE      CHANQ
        *
        *
        *
        *  OBTAIN CHANNEL.
        *
 179           SEIZE      CHAN
        *
        *
        *
        *  LEAVE QUEUE
        *
 180           DEPART     CHANQ
        *
        *
        *
 181           PRIORITY   8C
        *
        *
        *
        *
        *  REQUEST THE CPU TO ISSUE THE START I/O COMMAND AND TRANSFER
        *  THE BLOCK.
        *
        *  SAVE RETURN ADDRESS
        *
 182           ASSIGN     17,5,PF
        *
        *
        *
        *  TIME REQUIRED TO TRANSFER THE BLOCK.
        *
 183           ASSIGN     18,6,PF
        *
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
 184           TRANSFER   ,PROSD
        *
        *
        *
        *
        ************************************************
        *                CPU SCHEDULER
        ************************************************
        *
        *
        *
        *  IF THE PROCESS HAS PRIORITY 8C DO NOT INSERT IT INTO THE QUEUE.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
  185   PROSC  TEST NE    PR,80,ERGEN
        *
        *
        *
        *  SAVE THE PRIORITY OF THE PROCESS.
        *
  186          ASSIGN     19,PR,PF
        *
        *
        *
        *  THE PROCESS SCHEDULER HAS PRIORITY 70.
        *
  187          PRIORITY   70
        *
        *
        *
        *  WAIT FOR THE CPU.
        *
  188          QUEUE      READY
        *
        *
        *
        *  OBTAIN THE CPU.
        *
  189          PREEMPT    CPU,PR
        *
        *
        *
        *  LEAVE THE QUEUE.
        *
  190          DEPART     READY
        *
        *
        *
        *  THIS IS THE TIME TAKEN BY THE CPU SCHEDULER TO INSERT THE PROCESS
        *  INTO THE READY QUEUE.
        *
  191          ADVANCE    1
        *
        *
        *
        *  DONE WITH THE CPU.
        *
  192          RETURN     CPU
        *
        *
        *
        *  RESTORE PRIORITY.
        *
  193          PRIORITY   PF19
        *
        *
        *
        *  INSERT THE PROCESS INTO THE READY QUEUE.
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J          COMMENTS
        *
  194   EPGEN   QUEUE       READY
        *
        *
        *  ALLOCATE THE CPU TO THE PROCESS.
        *
  195           PREEMPT     CPU,PR
        *
        *
        *
        *  LEAVE THE QUEUE.
        *
  196           DEPART      READY
        *
        *
        *
        *  EXECUTE THE PROCESS.
        *
  197           ADVANCE     PF18
        *
        *
        *
        *  DONE WITH THE CPU.
        *
  198           RETURN      CPU
        *
        *
        *
        *  GO BACK TO THE POINT WHERE THE REQUEST WAS MADE.
        *
  199           TRANSFER    FN,21
        *
        *
        *
        ***********************************
        *        JOB SCHEDULER.
        ***********************************
        *
        *
        *
        *  GENERATE ONE TRANSACTION WHICH ACTIVATES THE SCHEDULER.
        *
  200           GENERATE    ,,,1,65,52PF
        *
        *
        *
        *  SEE IF THERE ARE JOBS WAITING FOR SCHEDULING.
        *
  201   BEGIN   TEST G      X2,C
        *
        *
        *
        *  SEE IF THE MEMORY IS FULL.
        *
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
   202          GATE SNF   MEMOR
                *
                *
                *
                *  IF MEMORY IS FULL OR IF THERE ARE NO JOBS WAITING FOR SCHEDULING,
                *  WAIT.
   203          TRANSFER   SIM,CAPT,BEGIN
                *
                *
                *
                *  REQUEST THE CPU TO EXECUTE THE JOB SCHEDULER.
                *  SAVE RETURN ADDRESS.
                *
   204   CAPT   ASSIGN     17,6,PF
                *
                *
                *
                *  PF16 IS THE TIME REQUIRED TO  EXECUTE THE JOB SCHEDULER.
                *
   205          ASSIGN     18,1,PF
                *
                *
                *
                *  TRANSFER TO THE CPU SCHEDULER.
                *
   206          TRANSFER   ,PROSC
                *
                *
                *
                *  RETURN FROM CPU SCHEDULER.
                *
                *  INITIALIZE X9.
                *
                * X9 IS THE NUMBER OF THE PARTITION UNDER EXAMINATION.
                *
   207   AAA6   SAVEVALUE  9,0,X
                *
                *
                *
   208   OUTER  SAVEVALUE  9+,1,X
                *
                *
                *
                *  IF ALL PARTITIONS  ARE EXAMINED GOTO OUT.
                *
   209          TEST LE    X9,X8,OUT
                *
                *
                *
                *  IF THIS PARTITION IS NOT FREE EXAMINE THE NEXT  PARTITION.
                *
   210          GATE LR    FN$LOGIC,OUTER
                *
                *
                *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *   INITIALIZE X10.
        *   X10 IS THE COLUMN OF THE MATRIX 1.
        *
  211            SAVEVALUE  10,0,X
        *
        *
        *
        *   ASSIGN TO X11 THE SIZE OF THE PARTITI3N X9.
        *
  212            SAVEVALUE  11,MX1(X9,5),X
        *
        *
        *
  213   INER     SAVEVALUE  10+,1,X
        *
        *
        *
        *   IF WE HAVE EXAMINED ALL THE CLASSES THAT THIS PARTITION CAN
        *   SERVE TRANSFER TO OUTER TO EXAMINE THE NEXT PARTITION.
        *
  214            TEST LE    X10,4,OUTER
        *
        *
        *
        *   X12 IS THE CLASS OF THE JOB.
        *
  215            SAVEVALUE  12,MX1(X9,X10),X
        *
        *
        *
        *   IF THIS PARTITION CANNOT SERVE THIS CLASS GOTO OUTER TO EXAMINE THE
        *   NEXT PARTITION.
        *
  216            TEST NE    X12,0,OUTER
        *
        *
        *
        *   SEE IF THE JOB QUEUE TABLE FOR THE CLASS X12 IS EMPTY.
        *
  217            TEST NE    CH*XF12,0,INER
        *
        *
        *
        *   ISSUE AN I/O PROCESS TO THE I/O DEVICE MANAGEMENT .
        *   ITS TASK IS TO MOVE THE TABLE FOR THE CLASS X12 FROM THE INPUT
        *   DRUM TO THE JOB SCHEDULER WORK AREA IN THE MAIN MEMORY WHERE THE
        *   OF THE TABLE ARE EXAMINED.
        *
        *   SAVE THE RETURN ADDRESS.
        *
  218            ASSIGN     20,3,PF
        *
        *
        *
        *   INPUT DRUM IS NEEDED.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *
 219            ASSIGN     4,1,PF
         *
         *
         *
         *  PF24 IS THE TIME REQUIRED TO MOVE A TABLE FROM THE INPUT DRUM TO THE
         *  MAIN MEMORY.
         *
 220            ASSIGN     24,4,PF
         *
         *
         *
 221            TRANSFER   ,DRUM
         *
         *
         *
         *  RETURN FROM PROCESS SCHEDULER.
         *  NOW REQUEST THE CPU IN ORDER TO EXAMINE THE ENTRIES OF THE TABLE.
         *
         *  SAVE THE RETURN ADDRESS.
         *
 222    BBB3    ASSIGN     17,7,PF
         *
         *
         *
         *  PF18 IS THE CPU TIME REQUIRED TO EXAMINE THE ENTRIES OF THE TABLE.
         *
 223            ASSIGN     18,2,PF
         *
         *
         *
 224            TRANSFER   ,PROSC
         *
         *
         *
         *  RETURN FROM CPU SCHEDULER.
         *
         *  SELECT A JOB FROM THE CLASS X12 THAT FITS INTO THE PARTITION
         *  UNDER EXAMINATION.
         *
 225    AAA7    UNLINK LE  X12,PREP,1,7PF,X11,INER
         *
         *
         *
 226            LOGIC S    2
         *
         *
         *
 227            GATE LR    2
         *
         *
         *
 226            TRANSFER   ,BEGIN
         *
         *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
 229    OUT    LOGIC S    15
        *
        *
        *
        *  WAIT UNTIL A NEW JOB COMES OR A PARTITION BECOMES FREE.
        *
 230           GATE LR    15
        *
        *
        *
 231           TRANSFER   ,BEGIN
        *
        *
        *
        *
        *****************************************************
        *  COMPILE AND LOAD THE JOB INTO ITS PARTITION.
        *****************************************************
        *
        *
        *  OPEN THE GATE FOR THE JOB SCHEDULER TRANSACTION.
        *
 232    PREP   LOGIC R    2
        *
        *
        *
 233           ASSIGN     39,MP38PF,PF
        *
        *
        *
        *  ALLOCATE MAIN MEMORY TO THE JOB.
        *
 234    COMLD  ENTER      MEMOR
        *
        *
        *
        * MARK THE TIME AT WHICH A PARTITION IS ALLOCATED TO THE JOB
        *
 235           MARK       40PF
        *
        *
        *
        *
        *  SAVE THE NUMBER OF THE PARTITION WHICH HAS BEEN ALOCATED TO THE JOB.
        *
 236           ASSIGN     41,XF9,PF
        *
        *
        *
 237           ASSIGN     28,FN$LOGIC,PF
        *
        *
        *
        *  THE PARTITION PF28 IS NOW OCCUPIED.
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J          COMMENTS
  238           LOGIC S    PF28
        *
        *
        *
        *  PF2 IS THE INTERNAL PRIORITY OF THE JOB.
        *
  239           ASSIGN     2,FN28,PF
        *
        *
        *
  240           SAVEVALUE  2-,1,X
        *
        *
        *
  241           ASSIGN     1,FN27,PF
        *
        *
        *
        *  SAVE THE MEMORY WHICH IS WASTED.
        *
  242           SAVEVALUE  PF1,V8,X
        *
        *
        *
  243           ASSIGN     42,XF*PF1,PF
        *
        *
        *
        *  IF MEMORY IS FULL COMPUTE THE FRAGMENTATION.
        *
  244           GATE SNF   MEMOR,FRAG
        *
        *
        *
        *
        *
        *
        *
        *  NOW WE ARE READY TO COMPILE AND LOAD THE PROGRAM.
        *
        *  TRANSFER TO THIS POINT WHENEVER A COMPILER IS REQUESTED.
        *
  245   FINIS   PRIORITY   60
        *
        *
        *
        *  SAVE RETURN ADDRESS.
        *
  246           ASSIGN     17,13,PF
        *
        *
        *
  247           ASSIGN     18,2,PF
        *
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *  TRANSFER TO THE CPU SCHEDULER.
        *
 248           TRANSFER   ,PPOSC
        *
        *
        *
        *  RETURN FROM PROCESS SCHEDULER.
        *  SEE IF AT THIS TIME WE ARE LOADING THE COMPILER.
        *
 249    AAA13  TEST E     XH*PF33,2,NADO
        *
        *
        *
 250           LINK       PF33,41PF
        *
        *
        *
        *  SEE IF THE COMPILER REQUESTED BY THE JOB IS IN THE MAIN MEMORY.
        *
 251    NADO   TEST NE    XH*PF33,1,QUP
        *
        *
        *
        *  IF THE COMPILER IS NOT IN THE MAIN MEMORY, SEE IF IT CAN BE LOADED
        *  (NOTICE THAT ONLY 2 COMPILERS MAY BE IN THE MEMORY AT THE SAME TIME).
        *
 252           SELECT E   1PH,22,23,0,XH,NOSPA
        *
        *
        *
        *  A SEGMENT OF MAIN MEMORY IS FOUND TO LOAD THE COMPILER.
        *
 253           SAVEVALUE  PH1,PF33,XH
        *
        *
        *
        *  REQUEST THE RELOCATABLE LOADER TO LOAD THE COMPILER.
        *
        *  INDICATE THAT THE COMPILER IS LOADING.
        *
 254    XXX1   SAVEVALUE  PF33,2,XH
        *
        *
        *
 255           ASSIGN     34,FN$BLOCK,PF
        *
        *
        *
        *  SAVE RETURN ADDRESS.
        *
 256           ASSIGN     35,1,PF
        *
        *
        *
 257           SPLIT      1,LOAD
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *
 258           LINK       PF33,41PF
        *
        *
        *
        *  X21 IS THE MEMORY FRAGMENTATION EXPRESSED IN PARTS PER THOUSAND.
        *
 259    FRAG   SAVEVALUE  21,V9,X
        *
        *
        *
        *  SAVE THE FRAGMENTATION.
        *
 260           MSAVEVALUE 2,XF30,XF31,XF21,MX
        *
        *
        *
 261           SAVEVALUE  31+,1,XF
        *
        *
        *
        *  SEE IF THE ROW OF MATRIX 2 IS FULL.
        *
 262           TEST E     XF31,6,RCOL
        *
        *
        *
 263           SAVEVALUE  31,1,XF
        *
        *
        *
 264           SAVEVALUE  30+,1,XF
        *
        *
        *
        *  SEE IF MATRIX 2 IS FULL.
        *
 265           TEST E     XF30,21,RCOL
        *
        *
        *
        *  PRINT OUT MATRIX 2.
        *
 266           ASSIGN     45,2,PF
        *
        *
        *
 267           ASSIGN     4,1,PH
        *
        *
        *
 268    TRP    ASSIGN     10,MX2(PH4,5),PF
        *
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *
         *
   269   *HELPA REPORT,PF45,MX2(PH4,1),MX2(PH4,2),MX2(PH4,3),MX2(PH4,4),PF10
         *
         *
         *
   270          ASSIGN     45,9,PF
         *
         *
         *
   271          ASSIGN     4+,1,PH
         *
         *
         *
   272          TEST E     PH4,21,TRP
         *
         *
         *
   273          SAVEVALUE  30,1,XF
         *
         *
         *
   274          MSAVEVALUE 2,1-20,1-5,0,MX
         *
         *
         *
   275   FCOL   TRANSFER   ,FINIS
         *
         *
         *
         *  TRANSFER TO THIS POINT WHENEVER THERE IS NO SPACE TO LOAD THE
         *  COMPILER.
         *  TEST TO SEE IF THE COMPILERS THAT ARE NOW IN THE MEMORY ARE BUSY.
         *
   276   NOSPA  TEST E     XH*XH22,1,MMM1
         *
         *
         *
   277          GATE NU    XH22,MMM1
         *
         *
         *
         *  ONE OF THE COMPILERS IN THE MAIN MEMORY IS FREE.
         *
   278          SAVEVALUE  XH22,J,XH
         *
         *
         *
   279          SAVEVALUE  22,PF33,XH
         *
         *
         *
         *  REQUEST THE LOADER TO LOAD THE COMPILER.
         *
   280          TRANSFER   ,XXX1
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *
 281    MMM1   TEST E     XH*XH23,1,MMM2
        *
        *
        *
 282           GATE NU    XH23,MMM2
        *
        *
        *
        *  ONE OF THE COMPILERS IN THE MAIN MEMORY IS FREE.
        *
 283           SAVEVALUE  XH23,J,XH
        *
        *
        *
 284           SAVEVALUE  23,PF33,XH
        *
        *
        *
        *  REQUEST THE LOADER TO LOAD THE COMPILER.
        *
 285           TRANSFER   ,XXX1
        *
        *
        *
        *  BOTH COMPILERS IN THE MAIN MEMORY ARE BUSY.
        *
 286    MMM2   LINK       SPES,41PF
        *
        *
        *
        *  RETURN FROM LOADER.
        *  INDICATE THAT THE COMPILER IS IN MEMORY.
        *
 287    NEXBL  SAVEVALUE  PF33,1,XH
        *
        *
        *
 288           UNLINK     PF33,MMM5,1
        *
        *
        *
 289           TERMINATE
        *
        *
        *
 290    PATR   UNLINK     SPES,FINIS,1
        *
        *
        *
 291           TRANSFER   ,ASSIN
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *
        *  TRANSFER TO THIS POINT WHENEVER THE COMPILER IS IN THE MEMORY.
        *  CAPTURE THE COMPILER IF POSSIBLE,OTHERWISE WAIT FOR THE COMPILER
        *  BY ENTERING A QUEUE.
        *
 292    QUP    LINK       PF33,41PF,MMM5
        *
        *
        *
        *  OBTAIN THE COMPILER.
        *
 293    MMM5   SEIZE      PF33
        *
        *
        *
 294           PRIORITY   PF2
        *
        *
        *
 295           MARK       48PF
        *
        *
        *
        *  PF1 IS THE NUMBER OF I/O REQUESTS.
        *
        *
 296           ASSIGN     1,V10,PF
        *
        *
        *
        *  REQUEST THE CANNEL TO TRANSFER A BLOCK OF SOURCE CODE FROM THE INPUT
        *  DISK INTO THE SOURCE CODE BUFFER OF THE COMPILER.
        *
        *  PF24 IS THE TIME REQUIRED TO TRANSFER THE BLOCK.
        *
 297           ASSIGN     24,6,PF
        *
        *
        *
        *  SAVE RETURN ADDRESS.
        *
 298           ASSIGN     20,4,PF
        *
        *
        *
        *  REQUEST INPUT DISK.
        *
 299           ASSIGN     4,2,PF
        *
        *
        *
 300           TRANSFER   ,DISK
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
```

```
BLOCK
NUMBER   *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
         *
         *  REQUEST THE CPU TO COMPILE THE CODE WHICH IS NOW IN THE SOURCE CODE
         *  BUFFER OF THE COMPILER AND AT THE SAME TIME TRANSFER THIS SOURCE
         *  CODE INTO OUTPUT DISK.
         *
  301    BBB4   LOGIC S    FN$CORSP
         *
         *
         *
  302           SPLIT      1,LIST
         *
         *
         *
         *  PF18 IS THE TIME REQUIRED TO COMPILE THE BLOCK OF SOURCE CODE.
         *
  303           ASSIGN     18,FN30,PF
         *
         *
         *
         *  SAVE RETURN ADDRESS.IN THIS ADDRESS WE WILL RETURN FROM PROCESS
         *  SCEDULER .
         *
  304           ASSIGN     17,8,PF
         *
         *
         *
         *  GO TO THE PROCESS SCHEDULER.
         *
  305           TRANSFER   ,PROSC
         *
         *
         *
         *  SAVE RETURN ADDRESS.
         *
  306    LIST   ASSIGN     20,7,PF
         *
         *
         *
         *  REQUEST OUTPUT DISK.
         *
  307           ASSIGN     4,8,PF
         *
         *
         *
         *  GO TO I/O DEVICE MANAGEMENT.
         *
  308           TRANSFER   ,DMAN
         *
         *
         *
         *  RETURN FROM I/O DEVICE MANAGEMENT.
         *
  309    BBB7   GATE LR    FN$CORSP
         *
         *
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
         *
  310           TRANSFER   ,ASSEB
         *
         *
         *
         *  RETURN FROM PROCESS SCHEDULER.
         *
  311    AAA8   LOGIC R    FN$CORSP
         *
         *
         *
         *  WAIT UNTIL BOTH PROCESSES ARRIVE.
         *
  312    ASSEB  ASSEMBLE   2
         *
         *
         *
  313           ASSIGN     1-,1,PF
         *
         *
         *
         *  FIND THE NUMBER OF STATEMENTS IN THE SOURCE CODE.
         *
  314           ASSIGN     34+,FN$SOURC,PF
         *
         *
         *
         *  SEE IF MORE SOURCE CODE REMAINS.
         *
  315           TEST NE    PF1,0,TELOS
         *
         *
         *
         *  REQUEST THE CHANNEL TO TRANSFER THE OBJECT CODE PRODUCED BY THE
         *  COMPILER FROM THE OBJECT CODE BUFFER TO THE INPUT DISK, AND AT THE
         *  SAME TIME TRANSFER THE NEXT BLOCK OF THE SOURCE CODE FROM THE
         *  INPUT DISK TO THE SOURCE CODE BUFFER.
         *
         *  SAVE RETURN ADDRESS.
         *
  316           ASSIGN     23,4,PF
         *
         *
         *
  317           ASSIGN     24,6,PF
         *
         *
         *
         *  REQUEST INPUT DISK.
         *
  318    MANA   ASSIGN     4,2,PF
         *
         *
         *
         *  GO TO DEVICE MANAGEMENT.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
  319           TRANSFER    ,DMAN
        *
        *
        *   NO SOURCE CODE REMAINS.
        *
        *   REQUEST THE CHANNEL TO TRANSFER THE OBJECT CODE FROM THE OBJECT CODE
        *   BUFFER OF THE COMPILER INTO THE INPUT DISK.
        *
        *   SAVE RETURN ADDRESS.
        *
  320   TELOS  ASSIGN     20,13,PF
        *
        *
        *
  321           ASSIGN     24,6,PF
        *
        *
        *
  322           TRANSFER    ,MANA
        *
        *
        *
        *   RETURN FROM I/O DEVICE MANAGEMENT.
        *   RETURN THE COMPILER.
        *
  323   BB913  RELEASE    PF33
        *
        *
        *
        *   ASSIGN TO PF49 THE COMPILATION TIME OF THE JOB.
        *
  324           ASSIGN     49,MP48PF,PF
        *
        *
        *
  325           UNLINK     PF33,MMM5,1,,,PATR
        *
        *
        *
        *   REQUEST THE RELOCATABLE LOADER TO RELOCATE AND LOAD THE OBJECT
        *   CODE PRODUCED BY THE COMPILER INTO THE PARTITION OF THE MEMORY
        *   ASSIGNED TO THE JOB BY THE JOB SCHEDULER.
        *
        *   SAVE RETURN ADDRESS.
        *
  326   ASSIN  ASSIGN     35,2,PF
        *
        *
        *
        *
        ****************************************************
        *   THE RELOCATABLE LOADER.
        ****************************************************
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *  WAIT FOR THE LOADER.
        *
  327   LOAD   QUEUE      LOADQ
        *
        *
        *
        *  OBTAIN THE LOADER.
        *
  328          SEIZE      LOADR
        *
        *
        *
        *  LEAVE THE QUEUE.
        *
  329          DEPART     LOADQ
        *
        *
        *
        *  REQUEST THE CHANNEL TO TRANSFER A BLOCK OF CODE FROM THE INPUT DISK
        *  TO THE BUFFER OF THE LOADER.
        *
        *  SAVE RETURN ADDRESS.
        *
  330   AGAN   ASSIGN     20,14,PF
        *
        *
        *
        *  INPUT DISK IS REQUESTED.
        *
  331          ASSIGN     4,2,PF
        *
        *
        *
  332          ASSIGN     24,6,PF
        *
        *
        *
  333          TRANSFER   ,DISK
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *  REQUEST THE CPU TO PRODUCE ABSOLUTE CODE. THIS CODE IS LOADED INTO
        *  A PARTITION IN THE MEMORY.
        *
        *  SAVE RETURN ADDRESS.
        *
  334   BBB14  ASSIGN     17,14,PF
        *
        *
        *
  335          ASSIGN     18,3,PF
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
 336           TRANSFER   ,PROSC
        *
        *
        *
        *  RETURN FROM CPU SCHEDULER.
        *
 337    AAA14  LOOP       34PF,AGAN
        *
        *
        *
        *  LEAVE THE LOADER.
        *
 338           RELEASE    LOADR
        *
        *
        *
 339           TRANSFER   FN,ADDRS
        *
        *
        *
        *  DECREASE THE NUMBER OF ENTRIES IN THE INPUT DRUM BY ONE.
        *
 340    XXX2   SAVEVALUE  1-,1,X
        *
        *
        *
        *  TEST TO SEE IF THE INPUT DRUM IS FULL.
        *
 341           GATE LS    FN3SHCRS,PILO
        *
        *
        *
        *  OPEN THE GATE.
        *
 342           LOGIC R    FN3SHCRS
        *
        *
        *
        *  ASSIGN  THE INTERNAL PRIORITY OF THE JOB.
        *
 343    PILO   PRIORITY   PF2
        *
        *
        *
        *  THE JOB IS READY TO BE EXECUTED.
        *
 344           ASSIGN     27,PF8,PF
        *
        *
        *
        *  SEE IF THE MEAN OF THE I/O INTERREQUEST INTERVAL  (PF16) IS LESS
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *  THAN THE CPU TIME REMAINING FOR THE JOB.
        *
  345   GEOR    TEST L     PF16,PF27,INTER
        *
        *
        *
  346           ASSIGN     18,PF16,PF
        *
        *
        *
        *  SAVE THE RETURN ADDRESS FROM CPU SCHEDULER.
        *
  347   MARIA   ASSIGN     17,9,PF
        *
        *
        *  TRANSFER TO THE CPU SCHEDULER.
        *
  348           TRANSFER   ,PROSC
        *
        *
        *
        *  PF18 IS THE I/O INTERREQUEST INTERVAL.
        *
  349   INTER   ASSIGN     18,PF27,PF
        *
        *
        *
  350           TRANSFER   ,MARIA
        *
        *
        *
        *  RETURN FROM GPU SCHEDULER.
        *
        *  REDUCE THE CPU TIME OF THE JOB BY ONE I/O INTERVEL.
        *
  351   AAA9    ASSIGN     27-,PF18,PF
        *
        *
        *  TEST TO SEE IF NO INPUT DATA REMAINS.
        *
  352   INPUT   TEST NE    PF14,0,OUTP
        *
        *
        *
        *  OBTAIN FROM  THE INPUT DISK THE NEXT INPUT BLOCK.
        *
        *  REQUEST THE INPUT DISK.
        *
  353           ASSIGN     4,2,PF
        *
        *
        *
        *  ASSIGN THE TIME REQUIRED TO TRANSFER THE BLOCK.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
  354          ASSIGN     24,4,PF
        *
        *
        *
        *  ASSIGN THE RETURN ADDRESS FROM THE I/O DEVICE MANAGEMENT.
        *
  355          ASSIGN     20,5,PF
        *
        *
        *
        *  GO TO I/O DEVICE MANAGEMENT.
        *
  356          TRANSFER   ,DISK
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *
        *  REDUCE INPUT BLOCKS BY ONE.
        *
  357   BBB5   ASSIGN     14-,1,PF
        *
        *
        *
        *  TEST FOR ZERO CPU TIME.
        *
  358          TEST NE    PF27,0,INPUT
        *
        *
        *
  359          TRANSFER   ,GEOR
        *
        *
        *
        *  TEST TO SEE IF NO OUTPUT BLOCKS REMAIN.
        *
  360   OUTP   TEST NE    PF12,0,VIKY
        *
        *
        *
        *  OBTAIN THE OUTPUT DISK TO PRINT OUT THIS BLOCK.
        *  REQUEST OUTPUT DISK.
        *
  361          ASSIGN     4,6,PF
        *
        *
        *
        *  ASSIGN THE TIME REQUIRED TO TRANSFER THE BLOCK.
        *
  362          ASSIGN     24,4,PF
        *
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *  SAVE THE RETURN ADDRESS FROM I/O DEVICE MANAGEMENT.
        *
 363    *      ASSIGN     20,6,PF
        *
        *
        *
        *  GO TO I/O DEVICE MANAGEMENT.
        *
 364           TRANSFER   ,DISK
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *
        *  REDUCE THE OUTPUT BLOCKS BY ONE.
        *
 365    BBB6   ASSIGN     12-,1,PF
        *
        *
        *
        *  TEST FOR ZERO CPU TIME.
        *
 366           TEST NE    PF27,0,OUTP
        *
        *
        *
 367           TRANSFER   ,GEOR
        *
        *
        *
        *  TEST FOR ZERO CPU TIME.
        *
 368    VIKY   TEST E     PF27,0,GEOR
        *
        *
        *
        *  THE JOB HAS COMPLETED EXECUTION AND IS READY TO BE PRINTED.
        *
        *  OPEN THE GATE FOR THE SCHEDULER TRANSACTION.
        *
 369           LOGIC R    15
        *
        *
        *
        *  LEAVE THE MEMORY.
        *
 370           LEAVE      MEMOR
        *
        *
        *
        *  RETURN THE PARTITION.
        *
 371           LOGIC R    PF28
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
  372   *        TEST NE    PH3,-1,JUMP
        *
        *
        *
        *   TABULATE THE FRAGMENTATION OF THE PARTITION  (THIS IS A WEIGHTED
        *   TABLE).
  373   *        TABULATE   PF41,MP40PF
        *
        *
        *
        *   TABULATE THE FRAGMENTATION OF THE PARTITION  (THIS IS NOT WEIGHTED).
  374   *        TABULATE   FN$TABL
        *
        *
        *
        *
        ************************************************
        *   OUTPUT SPOOLER AND LINE PRINTERS.
        ************************************************
        *
        *
        *   THE OUTPUT SPOOLER HAS PRIORITY 60.
  375   JUMP    PRIORITY    60
        *
        *
        *
        *   REQUEST THE CPU TO EXECUTE THE OUTPUT SPOOLER.
        *   SAVE THE RETURN ADDRESS FROM CPU SCHEDULER.
  376   *        ASSIGN     17,10,PF
        *
        *
        *
        *   PF18 IS THE SPOOLER EXECUTION TIME.
  377   *        ASSIGN     18,2,PF
        *
        *
        *
        *   TRANSFER TO THE CPU SCHEDULER.
  378   *        TRANSFER   ,PPOSC
        *
        *
        *
        *   RETURN FROM CPU SCHEDULER.
  379   AAA10   MARK        53PF
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *
        *  IF THE LINE PRINTER IS NOT FREE INSERT THE JOB INTO THE OUTPUT QUEUE.
        *
  380          TRANSFER   BOTH,LINPR,QLENG
        *
        *
        *
        *  OBTAIN THE LINE PRINTER
        *
  381   LINPR  SEIZE      PF31
        *
        *
        *
  382          MARK       50PF
        *
        *
        *
        *  ASSIGN TO PF54 THE TIME THAT THE JOB HAS TO WAIT IN THE OUTPUT QUEUE.
        *
  383          ASSIGN     54,MP53PF,PF
        *
        *
        *
        *  SAVE THE NUMBER OF OUTPUT LINES.
        *
  384          ASSIGN     29,PF15,PF
        *
        *
        *
        *  REQUEST THE CHANNEL TO TRANSFER A BLOCK OF INFORMATION FROM OUTPUT
        *  DISK TO THE BUFFER OF THE INPUT SPOOLER.
        *
        *  REQUEST OUTPUT DISK.
        *
  385          ASSIGN     4,8,PF
        *
        *
        *
        *  TRANSFER TIME IS 4MS.
        *
  386          ASSIGN     24,4,PF
        *
        *
        *
        *  SAVE RETURN ADDRESS.
        *
  387          ASSIGN     20,10,PF
        *
        *
        *
        *  GO TO I/O DEVICE MANAGEMENT.
        *
  388          TRANSFER   ,DISK
        *
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *  IF MORE THAN 6 JOBS ARE WAITING FOR THE LINE PRINTER GO TO AUXIL.
        *
  389   OLENG  TEST LE    CH*PF31,6,AUXIL
        *
        *
        *
  390   DELTA  LINK       PF31,15PF
        *
        *
        *
        *  TEST TO SEE IF THE AUXILIARY LINE PRINTER IS REQUESTED .
        *
  391   AUXIL  TEST E     BV3,1,DELTA
        *
        *
        *
  392          ASSIGN     31,9,PF
        *
        *
        *
  393          GATE U     9,LINPR
        *
        *
        *
        *  IF THE QUEUE OF THE AUXILIARY LINE PRINTER IS LESS THAN 5 ASSIGN THE
        *  JOB TO AUXILIARY LINE PRINTER.
        *
  394          LINK       9,15PF
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *
        * TEST IF THE TRANSFERED BLOCK HAS 10 RECORDS OR LESS.
        *
  395   BB81C  TEST GE    PF29,10,LESS
        *
        *
        *
        *     THE BLOCK HAS 10 RECORDS.
  396          ASSIGN     30,10,PF
        *
        *
        *
        *  REDUCE THE NUMBER OF OUTPUT LINES BY PF30
        *
  397   THETA  ASSIGN     29-,PF30,PF
        *
        *
        *
        * REQUEST THE CHANNEL TO TRANSFER A RECORD FROM THE BUFFER OF THE OUTPUT
        *  SPOOLER TO THE BUFFER OF THE LINE PRINTER.
        *
        *  SAVE RETURN ADDRESS.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
  396          ASSIGN     20,11,PF
        *
        *
        *
        *  GO TO I/O DEVICE MANAGEMENT.
        *
  399   KAPA   TRANSFER   ,DMAN
        *
        *
        *
        *  THE BLOCK HAS LESS THAN 10 RECORDS.
        *
  400   LESS   ASSIGN     30,PF29,PF
        *
        *
        *
  401          TRANSFER   ,THETA
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *  IF THE BUFFER OF THE OUTPUT SPOOLER IS EMPTY REQUEST THE CHANNEL TO
        *  TRANSFER THE NEXT BLOCK OF INFORMATION.
        *
  402   BBB11  TEST E     BV5,1,PPP1
        *
        *
        *
  403          ASSIGN     2,-30,PH
        *
        *
        *
  404          SPLIT      1,PPP2
        *
        *
        *
        *  TIME TAKEN BY THE LINE PRINTER TO PRINT OUT ONE LINE.
  405   PPP1   ADVANCE    60
        *
        *
        *
  406          TEST E     PH2,-30,PPP3
        *
        *
        *
  407          ASSIGN     2,0,PH
        *
        *
        *
  408          LOGIC R    FNBLINPE
        *
        *
        *
  409   PPP4   ASSEBLE    2
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
        *
 410           TRANSFER   ,BBB10
        *
        *
        *
        *    IF THERE ARE MORE RECORDS LEFT IN THE BUFFER OF THE OUTPUT SPOOLER
        *    RETURN TO PRINT THE NEXT RECORD.
        *
 411    PPP3   LOOP       30PF,KAPA
        *
        *
        *
        *    THE JOB HAS BEEN PRINTED.
        *
        *    RETURN THE LINE PRINTER.
        *
 412           RELEASE    PF31
        *
        *
        *
        *    ASSIGN TO PF51 THE TIME TAKEN BY THE LINE PRINTER TO PRINT
        *    THE JOB.
        *
 413           ASSIGN     51,MP50PF,PF
        *
        *
        *
        *    TAKE THE NEXT JOB FROM THE OUTPUT QUEUE AND ALOCATE IT TO THE
        *    LINE PRINTER .
        *
 414           UNLINK     PF31,LINPR,1
        *
        *
        *
        *    TEST TO SEE IF THIS IS A NON STANDARD JOB.
        *
 415           TEST NE    PH3,-1,NONST
        *
        *
        *
 416           ASSIGN     45,3,PF
        *
        *
        *
        *    ASSIGN TO PF46 THE TURNAROUND TIME OF THE JOB.
        *
 417           ASSIGN     46,M1,PF
        *
        *
        *
        *    THIS SAVEVALUE IS USED TO CALCULATE THE AVERAGE TURNAROUND TIME.
        *
 418           SAVEVALUE  26+,PF46,XF
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J          COMMENTS                N
         *
         *
         *
  419          HELPA      REPORT,PF45,PF52,PF5,PF6,PF2,PF33
         *
         *
         *
  420          ASSIGN     45,4,PF
         *
         *
         *
  421          HELPA      REPORT,PF45,PF7,PF8,PF3,PF47,PF36
         *
         *
         *
  422          ASSIGN     45,5,PF
         *
         *
         *
  423          HELPA      REPORT,PF45,PF37,PF39,PF54,PF49,PF51
         *
         *
         *
  424          ASSIGN     45,6,PF
         *
         *
         *
  425          HELPA      REPORT,PF45,PF46,PF52,PF13,PF14,PF15
         *
         *
         *
  426          TRANSFER   ,OKY
         *
         *
         *
  427   NONST  ASSIGN     45,7,PF
         *
         *
         *
  428          HELPA      REPORT,PF45,PF15,PF16,PF12,PF35,PF19
         *
         *
         *
  429          ASSIGN     45,4,PF
         *
         *
         *
  430          HELPA      REPORT,PF45,PF7,PF8,PF3,PF47,PF36
         *
         *
         *
  431          ASSIGN     45,5,PF
         *
         *
         *
```

```
 BLOCK
 NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
   432          HELPA      REPORT,PF45,PF37,PF39,PF54,PF49,PF51
         *
         *
         *
   433   OKY    TERMINATE
         *
         *
         *
         *  REQUEST OUTPUT DISK.
         *
   434   PPP2   ASSIGN     4,8,PF
         *
         *
         *
         *  TRANSFER TIME IS 4MS.
         *
   435          ASSIGN     24,4,PF
         *
         *
         *
         *  SAVE RETURN ADDRESS.
         *
   436          ASSIGN     20,15,PF
         *
         *
         *
   437          LOGIC S    FN$LINPE
         *
         *
         *
   438          TRANSFER   ,DISK
         *
         *
         *
   439   BB315  GATE LR    FN$LINPE
         *
         *
         *
   440          TRANSFER   ,PPP4
         *
         *
         *
         *
         *
         *
         *
         *  THE TIMER ARRIVES AT TIME X$TIME.
         *
   441          GENERATE   X$TIME,,,1,,54PF,4PH
         *
         *
         *
   442          SPLIT      1,PETR
         *
         *
         *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
  443          ASSIGN     1,-9,PF
        *
        *
        *
  444          TRANSFER   ,PRIOT
        *
        *
        *
        *  DO NOT ALLOW MORE JOBS TO COME INTO THE SYSTEM.
        *
  445   PETR   LOGIC S    22
        *
        *
        *
        *  WAIT UNTIL THE QUEUE FOR THE CARD READERS AND THE CARD READERS
        *  THEMSELVES ARE FREE.
        *
  446          TEST E     BV4,1
        *
        *
        *
        *  WAIT UNTIL ALL THE REGULAR JOBS HAVE BEEN SERVED.
        *
  447          TEST E     V12,N$OKY
        *
        *
        *
        *  PRINT OUT THE FRAGMENTATION MATRIX 2.
        *
  448   PRIOT  ASSIGN     4,1,PH
        *
        *
        *
  449          ASSIGN     45,2,PF
        *
        *
        *
  450   MAKL   ASSIGN     10,MX2(PH4,5),PF
        *
        *
        *
  451   HELPA REPORT,PF45,MX2(PH4,1),MX2(PH4,2),MX2(PH4,3),MX2(PH4,4),PF10
        *
        *
        *
  452          ASSIGN     45,10,PF
        *
        *
        *
  453          ASSIGN     4+,1,PH
        *
        *
        *
  454          TEST G     PH4,XF30,MAKL
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J      COMMENTS
        *
        *
 455           TEST NE    PF1,-9,TERM
        *
        *
        *
        *  SEE IF THERE ARE ANY SPECIAL JOBS WAITING FOR THE SCHEDULER.
        *
 456           TEST NE    CH$WAIT1,C,TERM
        *
        *
        *
        *
        *
        ***********************************************
        *  JOB SCHEDULER FOR NON STANDARD JOBS.
        ***********************************************
        *
        *
        *
        *  ISSUE AN I/O PROCESS TO THE I/O DEVICE MAN. ,ITS TASK IS TOTRANSFER
        *  THE TABLE WHICH CONTAINS THE CHARACTERISTICS OF THE SPECIAL JOBS FROM
        *  THE INPUT DRUM TO THE MAIN MEMORY IN ORDER TO EXAMINE ITS ENTRIES.
        *
        *  SAVE RETURN ADDRESS.
        *
 457           ASSIGN     20,8,PF
        *
        *
        *
        *  INPUT DRUM IS REQUESTED.
        *
 458           ASSIGN     4,1,PF
        *
        *
        *
        *  ASSIGN THE TRANSFER TIME.
        *
 459           ASSIGN     24,4,PF
        *
        *
        *
 460           TRANSFER   ,DRUM
        *
        *
        *
        *  RETURN FROM I/O DEVICE MANAGEMENT.
        *
        *  REQUEST THE CPU TO EXAMINE THE ENTRIES OF THE TABLE.
        *
        *  SAVE RETURN ADDRESS.
        *
 461    BB38   ASSIGN     17,12,PF
        *
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
        *
        *
 462    *      ASSIGN     18,1,PF
        *
        *
        *
 463    *      TRANSFER   ,PROSC
        *
        *
        *
        *   RETURN FROM CPU SCHEDULER.
        *   TAKE THE FIRST JOB FROM THE QUEUE AND SCHEDULE IT FOR EXECUTION.
        *
 464    AAA12  UNLINK     WAIT1,TORA,1,,,UNT
        *
        *
        *
 465    *      LOGIC S    8
        *
        *
        *
        *   WAIT UNTIL THE MEMORY BECOMES FREE.
        *
 466    *      GATE LR    8
        *
        *
        *
 467    *      TRANSFER   ,BBB9
        *
        *
        *
        *   ASSIGN TO THIS JOB THE COMPLETE MEMORY WHICH IS AVAILABLE TO THE
        *   USERS.
        *
 468    TORA   SAVEVALUE  9,1,XF
        *
        *
        *
        *   INDICATE THAT THIS IS A NON STANDARD JOB.
        *
 469    *      ASSIGN     3,-1,PH
        *
        *
        *
 470    *      SAVEVALUE  25-,1,XF
        *
        *
        *
        *   ASSIGN TO PF39 THE TIME THAT THE JOB HAS TO WAIT IN THE INPUT QUEUE
        *
 471    *      ASSIGN     39,MP38PF,PF
        *
        *
        *
        *   TRANSFER TO COMLD WHERE WE COMPILE AND LOAD THE JOB.
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
  472          TRANSFER   ,COMLO
        *
        *
        *
        *  WAIT UNTIL ALL THE NON STANDARD JOBS ARE PRINTED .
        *
  473   UNT    TEST E     N$WAIT,N$OKY
        *
        *
  474   TERM   ASSIGN     45,8,PF
        *
        *
  475          HELPA      REPORT,PFL5,XF25,XF26,PF3,PF1,PF2
        *
        *
        *
  476          TERMINATE  1
        *
        *
        *
        *
        *
        *
        *  THIS SEGMENT IS TO MAKE SURE THAT THE SUM OF THE SIZE OF THE
        *  PARTITIONS DOES NOT SUPPASS THE MEMORY AVAILABLE TO THE USERS.
        *
  477          GENERATE   ,,,1,76
        *
        *
        *
  478          TEST LE    V13,XF20,NOKY
        *
        *
        *
  479          TERMINATE
        *
        *
        *
  480   NOKY   ASSIGN     45,11,PF
        *
        *
        *
  481          HELPA      REPORT,PFL5,PF1,PF2,PF3,PF4,PF5
        *
        *
        *
  482          TERMINATE  2
        *
        *
        *
        *
        *
```

```
BLOCK
NUMBER  *LOC    OPERATION  A,B,C,D,E,F,G,H,I,J        COMMENTS
        *
        *
                REPORT
        CLO     TITLE      ,CLOCK≠S STATISTICS.
                SPACE      3
                SPACE      3
                SPACE      3
        BLO     TITLE      ,BLOCK COUNTS.
                SPACE      3
                SPACE      3
                SPACE      3
        FSV     TITLE      20,THE MEMORY AVAILABLE TO THE USERS EXPRESSED IN  K 0
        BYTES.
                SPACE      3
                SPACE      3
        FSV     TITLE      2,THIS IS THE NUMBER OF PARTITIONS IN THE MAIN MEMORY
                SPACE      3
                SPACE      3
                SPACE      3
        FAC     TITLE      1,STATISTICS FOR THE INPUT DRUM.
                SPACE      3
                SPACE      3
        FAC     TITLE      2,STATISTICS FOR THE INPUT DISK.
                SPACE      3
                SPACE      3
        FAC     TITLE      8,STATISTICS FOR THE OUTPUT DISK.
                SPACE      3
                SPACE      3
        FAC     TITLE      3,STATISTICS FOR THE CARD READER 1.
                SPACE      3
                SPACE      3
        FAC     TITLE      4,STATISTICS FOR THE CARD READER 2
                SPACE      3
                SPACE      3
        FAC     TITLE      10,STATISTICS FOR THE LINE PRINTER 1
                SPACE      3
                SPACE      3
        FAC     TITLE      6,STATISTICS FOR THE AUXILIARY LINE PRINTEF.
                SPACE      3
                SPACE      3
        FAC     TITLE      17,STATISTICS FOR THE COMPILEP 17.
                SPACE      3
                SPACE      3
        FAC     TITLE      16,STATISTICS FOR THE COMPILER 18.
                SPACE      3
                SPACE      3
        FAC     TITLE      19,STATISTICS FOR THE COMPILER 19.
                SPACE      3
                SPACE      3
        FAC     TITLE      20,STATISTICS FOR THE COMPILEP 20.
                SPACE      3
                SPACE      3
        FAC     TITLE      21,STATISTICS FOR THE COMPILEP 21.
                SPACE      3
                SPACE      3
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
         FAC    TITLE      LOADR,STATISTICS FOR THE RELOCATABLE LOADER.
                SPACE      3
                SPACE      3
         FAC    TITLE      CHAN,STATISTICS FOR THE CHANNAL
                SPACE      3
                SPACE      3
         FAC    TITLE      CPU,STATISTICS FOR THE CPU
                SPACE      3
                SPACE      3
                SPACE      3
         STO    TITLE      MEMOR,STATISTICS FOR THE MAIN MEMORY
                SPACE      3
                SPACE      3
                SPACE      3
         QUE    TITLE      1,STATISTICS FOR THE QUEUE OF THE INPUT DRUM
                SPACE      3
                SPACE      3
         QUE    TITLE      2,STATISTICS FOR THE QUEUE OF THE INPUT DISK
                SPACE      3
                SPACE      3
         QUE    TITLE      0,STATISTICS FOR THE QUEUE OF THE OUTPUT DISK
                SPACE      3
                SPACE      3
         QUE    TITLE      3,STATISTICS FOR THE QUEUE OF THE CARD READER 1
                SPACE      3
                SPACE      3
         QUE    TITLE      4,STATISTICS FOR THE QUEUE OF THE CARD READER 2
                SPACE      3
                SPACE      3
         CHA    TITLE      9,STATISTICS FOR THE OUTPUT QUEUE OF THE AUXILIARY  0
        LINE PRINTER.
                SPACE      3
                SPACE      3
         CHA    TITLE      16,STATISTICS FOR THE OUTPUT QUEUE OF THE LINE PRINTERO
        P 1.
                SPACE      3
                SPACE      3
         CHA    TITLE      17,STATISTICS FOR THE QUEUE OF THE COMPILER 17.
                SPACE      3
                SPACE      3
         CHA    TITLE      18,STATISTICS FOR THE QUEUE OF THE COMPILER 18.
                SPACE      3
                SPACE      3
         CHA    TITLE      19,STATISTICS FOR THE QUEUE OF THE COMPILER 19.
                SPACE      3
                SPACE      3
         CHA    TITLE      20,STATISTICS FOR THE QUEUE OF THE COMPILER 20.
                SPACE      3
                SPACE      3
         CHA    TITLE      21,STATISTICS FOR THE QUEUE OF THE COMPILER 21.
                SPACE      3
                SPACE      3
         QUE    TITLE      CHANQ,STATISTICS FOR THE QUEUE OF THE CHANNAL
                SPACE      3
                SPACE      3
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
         QUE    TITLE      LOADQ,STATISTICS FOR THE QUEUE OF THE RELOCATABLE LOAD
        DEF.
                SPACE      3
                SPACE      3
         QUE    TITLE      READY,STATISTICS FOR THE QUEUE OF THE CPU
                SPACE      3
                SPACE      3
                SPACE      3
         CHA    TITLE      1,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS A.
                SPACE      3
                SPACE      3
         CHA    TITLE      2,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS B
                SPACE      3
                SPACE      3
         CHA    TITLE      3,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS C
                SPACE      3
                SPACE      3
         CHA    TITLE      4,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS D
                SPACE      3
                SPACE      3
         CHA    TITLE      5,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 00
        TO CLASS E
                SPACE      3
                SPACE      3
         CHA    TITLE      6,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS F
                SPACE      3
                SPACE      3
         CHA    TITLE      7,STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0
        TO CLASS G
                SPACE      3
                SPACE      3
         CHA    TITLE      WAIT1,STATISTICS FOR THE INPUT QUEUE OF NON-STANDARD J
        JOBS.
                SPACE      3
                SPACE      3
         FMS    TITLE      1,MATRIX 1.
                SPACE      3
                SPACE      3
                SPACE      3
         5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
         5      TEXT       1,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
         5      TEXT       PARTITION.
         TAB    INCLUDE    T1/1,6,7,8,9,10,11,12,13,14,16
                SPACE      3
                SPACE      3
                SPACE      3
         5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
         5      TEXT       2,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
         5      TEXT       PARTITION.
         TAB    INCLUDE    T2/1,6,7,8,9,10,11,12,13,14,16
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
               SPACE      3
               SPACE      3
               SPACE      3
        5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
        5      TEXT       3,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
        5      TEXT       PARTITION.
        TAB    INCLUDE    T3/1,6,7,8,9,10,11,12,13,14,16
               SPACE      3
               SPACE      3
               SPACE      3
        5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
        5      TEXT       4,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
        5      TEXT       PARTITION.
        TAB    INCLUDE    T4/1,6,7,8,9,10,11,12,13,14,16
               SPACE      3
               SPACE      3
               SPACE      3
        5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
        5      TEXT       5,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
        5      TEXT       PARTITION.
        TAB    INCLUDE    T5/1,6,7,8,9,10,11,12,13,14,16
               SPACE      3
               SPACE      3
               SPACE      3
        5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
        5      TEXT       6,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
        5      TEXT       PARTITION.
        TAB    INCLUDE    T6/1,6,7,8,9,10,11,12,13,14,16
               SPACE      3
               SPACE      3
               SPACE      3
        5      TEXT       THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION
        5      TEXT       7,WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS
        5      TEXT       PARTITION.
        TAB    INCLUDE    T7/1,6,7,8,9,10,11,12,13,14,16
               SPACE      3
               SPACE      3
               SPACE      3
        TAB    TITLE      8,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITIO
       ON 1 (IT IS NOT WEIGHTED).
               SPACE      3
               SPACE      3
               SPACE      3
        TAB    TITLE      9,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITIO
       ON 2 (IT IS NOT WEIGHTED).
               SPACE      3
               SPACE      3
               SPACE      3
        TAB    TITLE      10,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITI
       ON 3 (IT IS NOT WEIGHTED).
               SPACE      3
               SPACE      3
               SPACE      3
        TAB    TITLE      11,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITI
       ON 4 (IT IS NOT WEIGHTED).
```

```
BLOCK
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I,J       COMMENTS
               SPACE       3
               SPACE       3
               SPACE       3
         TAB   TITLE       2,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITI
        ON 5 (IT IS NOT WEIGHTED).
               SPACE       3
               SPACE       3
               SPACE       3
         TAB   TITLE       3,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITI
        ON 6 (IT IS NOT WEIGHTED).
               SPACE       3
               SPACE       3
               SPACE       3
         TAB   TITLE       4,THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITI
        ON 7 (IT IS NOT WEIGHTED).
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               SPACE       3
               START       2,,1
               END
```

APPENDIX B

GPSS statements required for one additional input station.

Statement	Description
CAR_J FUNCTION A,B $X_1,Y_1/X_2,Y_2/\ldots/X_n,Y_n$	This Function provides the number of input cards of a job that arrives at an input station and may be any distribution function. J takes the values 1, 2, 3, 4 and 5 for the input stations 1, 2, 3, 4 and 5 respectively.
K FUNCTION A,B $X_1,Y_1/X_2,Y_2/\ldots/X_m,Y_m$	This Function provides the Job-class to which a job, that arrives at an input station, belong. K takes the values 1, 2, 33, 34 and 35 for the input stations 1, 2, 3, 4 and 5 respectively. Function K may be any distribution function.

Add the following segment between blocks 8 and 9 (Appendix A).

GENERATE A,B,,,54PF,4PH	This statement can be thought of as a door through which users enter the room where a card reader is located (a input station). The specification of the interarrival-time distribution of the users, is expressed through the A and B operands of the GENERATE Block.
GATE LR 22	This statement can be thought of as a lock which locks the door of the room in which a card reader is located. The timer transaction, which arrives at GENERATE block no. 441 (Appendix A), turns the logic switch 22 to "set".

ASSIGN 1,FN$CARJ,PF	Assigns to the fullword parameter one, the number of input card of a job.
ASSIGN 2,M,PF	Assigns to the fullword parameter 2, the identification number of a card reader. M takes the values 3, 4, 5, 6 and 7 for the card readers 1, 2, 3, 4 and 5 respectively.
ASSIGN 3,I,PF	Where I represents the previously defined function K.
ASSIGN 31,L,PF	Assigns to the fullword parameter 31 the code of a line printer. L takes the values 10, 11, 12, 13 and 14 for the line printers 1, 2, 3, 4 and 5 respectively.
ASSIGN 32,N,PF	N takes the value 9, if the auxiliary line printer is requested and 0, if the auxiliary line printer is not requested.
TRANSFER ,WAIT	This statement sends a job to wait, if necessary, for the card reader.

REFERENCES

[BRI 73] Brinch Hansen, P., Operating System Principles, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1973.

[BRI 70] Brinch Hansen, P., "The nucleus of a multiprogramming system", Comm. ACM, 13, No. 4 (April 1970), 238-241, 250.

[CDC 71] CDC Corporation, "General Prupose Simulation System-V/600", General Information Manual (Form 84003900), Minneapolis, Minnesota, 1971.

[COM 65] Comfort, W.T., "A computing system design for user service", Proc. AFIPS 1965 Fall Joint Comput. Conf., 27 Spartan Books, New York, 619-628.

[FLO 69] Flores, I., Computer Organization, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1969.

[FOX 67] Fox, D., and Kessler, J.L., "Experiments in software modeling", Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, pp. 429-436.

[GLI 67] Glinka, L.R., Brush, R.M., and Ungan, A.J., "Design thru simulation of a multiple-access information system", Proc. AFIPS 1976 Fall Joint Comput. Conf., Vol. 31, pp. 437-447.

[GOR 75] Gordon, Geoffrey, The Application of GPSSV to Design System Simulation, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

[HEL 75] Hellerman, H., and Conroy, F.T., Computer System Performance, McGraw-Hill, New York, 1975.

[HOL 68] Holland, C.F., and Merikallio, A.R., "Simulation of a multiprocessing system using GPSS", IEEE 1968, Vol. 4, pp. 395-400.

[HUE 67] Huesman, L.R., and Goldberg, R.P., "Evaluating computer system through simulation", Comput. J. 10, 2 (Aug. 1967), pp. 150-156.

[IBM 74] IBM Corporation, "General Purpose Simulation System-V", User's Manual (Form SH20-0851-1), White Plains, N.Y., 1974.

[KAT 66] Katz, J.H., "Simulation of a multiprocessor computer system", Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, pp. 127-139.

[KEC 75] Keech, G.L., "Computer Organization", Course notes for Applied Mathematics 3D3/6D3, September, 1975.

[KUR 75] Kurzban, S., Heines, T.S., and Sayers, A.P., Operating Systems Principles, Petrocelli/charter, New York, 1975, 416-427.

[LAM 76] Lamie, L.E., "Using GPSS to Teach Operating System Concepts", Sigese Bulletin, Vol. 8 (Febr. 1976), pp. 174-178.

[LEH 68] Lehman, M.M., and Rosenfeld, J.L., "Performance of a simulated multiprogramming system", Proc. AFIPS 1968 Fall Joint Comput. Conf., Vol. 33, pp. 1431-1442.

[MAC 67] MacDougall, M.H., "Simulation of an ECS-based operating system", Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, pp. 735-741.

[MAD 74] Madnick, E.M., and Donovan, J.J., Operating Systems, McGraw-Hill, New York, 1974.

[SCH 74] Schriber, T.J., Simulation Using GPSS, Wiley, New York, 1974.

[SHA 74] Shaw, A.C., The Logical Design of Operating Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1974.

[TSI 74] Tsichritzis, C.D., and Berstein, A.P., Operating Systems, Academic Press, New York, 1974.