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

© GEORGE DROUCAS 1977
MASTER OF SCIENCE (1976) 
(Computation) 
McMaster University 
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  
   1.1 Operating Systems Terminology Used in this Report  

2. DESCRIPTION OF THE OPERATING SYSTEM  

3. HARDWARE COMPONENTS OF THE SYSTEM  
   3.1 Card Readers  
   3.2 Input Drum  
   3.3 Input Disk and Output Disk  
   3.4 Line Printers  
   3.5 Channels  

4. DESCRIPTION OF THE SOFTWARE MODULES OF THE SYSTEM  
   4.1 Input Spooler  
      4.1.1 Algorithm of the Input Spooler  
   4.2 Memory Management  
      4.2.1 Job Classification  
   4.3 Job Scheduling  
      4.3.1 Job Scheduler for Regular Jobs  
      4.3.1.1 Algorithm of the Job Scheduler for Regular Jobs  
      4.3.2 Job Scheduler for Non-Standard Jobs  
   4.4 Compiler Management and Compilers  
      4.4.1 Compilers  
      4.4.2 Compiler Management  
      4.4.2.1 Algorithm of the Compiler Manager  
   4.5 Relocatable Loader  
   4.6 Internal Priority  
   4.7 CPU Scheduler  
   4.8 I/O Device Management  
   4.9 Output Spooler
5. RESULTS AND CONCLUSIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Equipment Statistics</td>
<td>69</td>
</tr>
<tr>
<td>5.2 Main Memory Statistics and Fragmentation</td>
<td>70</td>
</tr>
<tr>
<td>5.3 Job Statistics</td>
<td>78</td>
</tr>
<tr>
<td>5.4 Summary</td>
<td>86</td>
</tr>
</tbody>
</table>

APPENDIX A: A Listing of the GPSS Program SCOS 89

APPENDIX B: GPSS Statements Required for one Additional Input Station 165

REFERENCES 167
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 transferred 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 2.2. 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 characteristics 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:

- \( \text{rotation speed} = 20 \text{ ms} \)
- \( \text{maximum access} = 0.75 \text{ ms} \)
- \( \text{maximum arm movement} = 55 \text{ ms} \)
- \( \text{average access} = 35 \text{ ms} \)
- \( \text{serial access (depending on length of record)} \approx 1 \text{ ms} \).
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 throughput. 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 available for use. If it is being used to hold an input deck, it may be in either input, hold, or run
<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALFA</td>
<td>RUN</td>
<td></td>
</tr>
<tr>
<td>BETA</td>
<td>HOLD</td>
<td></td>
</tr>
<tr>
<td>GAMA</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AVAILABLE</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 1**

<table>
<thead>
<tr>
<th>CLASS</th>
<th>NAME</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 2**

Figure 4-1. Tables used by the input spooler
status:

**input**: The input deck is still being read.

**hold**: The input deck has been completely copied onto the input disk but the corresponding job has not been started yet.

**run**: 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 transferred 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. Algorithm of the Input Spooler

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

- $\text{CBUF}(i)$ is the buffer of card reader $i$.
- $\text{SBUF}(i)$ is the buffer of the input spooler that corresponds to card reader $i$. 
I/O complete interrupt from card-reader i

- **NO**
  - is FLAG(i) = 1 ?
  - Wait until FLAG(i) = 0

- **YES**
  - 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

- **NO**
  - A
  - **NO**
    - Examine the characteristics of the job
  - **YES**
    - B

- **YES**
  - is it a job card ?
  - **YES**
    - is the input disk full ?
      - **YES**
        - Signal the card readers to stop reading any additional jobs
      - **NO**
        - signals to card readers
A

Is buffer SBUF(i) full?

YES

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

NO

Is this the last card of the Job?

YES

FLAG(i) = 1

Send a request to I/O device management

NO

Signal card reader i to read the next card

Signal from channel that transfer is over

FLAG(i) = 0

Return
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

\[ \text{FLAG}(i) = 1 \]

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

\[ \text{FLAG}(i) = 0 \]

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
<table>
<thead>
<tr>
<th>Partition</th>
<th>Partition Size</th>
<th>Job Size</th>
<th>Wasted Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8K</td>
<td>1K</td>
<td>7K</td>
</tr>
<tr>
<td>2</td>
<td>32K</td>
<td>9K</td>
<td>23K</td>
</tr>
<tr>
<td>3</td>
<td>120K</td>
<td>35K</td>
<td>85K</td>
</tr>
<tr>
<td>4</td>
<td>120K</td>
<td>39K</td>
<td>81K</td>
</tr>
<tr>
<td>5</td>
<td>520K</td>
<td>125K</td>
<td>395K</td>
</tr>
<tr>
<td></td>
<td>800K</td>
<td>209K</td>
<td>591K</td>
</tr>
</tbody>
</table>

*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.
<table>
<thead>
<tr>
<th>ROW</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>.30</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>B</td>
<td>E</td>
<td>G</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 A3 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 A3 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.

\( \phi_1 \) to \( \phi_{12} \) are integers which represent the number of output lines of a user job, without including the job listing and system messages. For example \( \phi_5 = 40 \) output lines, \( \phi_7 = 100 \) 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 \( I_3 = 50\% \), \( I_4 = 90\% \).
<table>
<thead>
<tr>
<th>CLASS</th>
<th>CPU</th>
<th>MEMORY</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
<td>S</td>
<td>S1</td>
</tr>
<tr>
<td>B</td>
<td>M</td>
<td>S</td>
<td>L2</td>
</tr>
<tr>
<td>C</td>
<td>M</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>D</td>
<td>S2</td>
<td>M</td>
<td>L1</td>
</tr>
<tr>
<td>E</td>
<td>M</td>
<td>M</td>
<td>L2</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
<td>L</td>
<td>L3</td>
</tr>
<tr>
<td>G</td>
<td>L</td>
<td>M</td>
<td>S2</td>
</tr>
</tbody>
</table>

Figure 4-5. Job classification
<table>
<thead>
<tr>
<th>MEMORY</th>
<th>CPU</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 &lt; M2</td>
<td>T1 &lt; T2</td>
<td>L1</td>
</tr>
<tr>
<td>M2 &lt; M3</td>
<td>T2 &lt; T3</td>
<td>L2</td>
</tr>
<tr>
<td>M3 &lt; M4</td>
<td>T3 &lt; T4</td>
<td>L3</td>
</tr>
<tr>
<td>I1 &lt; I2</td>
<td>S1</td>
<td></td>
</tr>
<tr>
<td>I2 &lt; I3</td>
<td>S2</td>
<td></td>
</tr>
<tr>
<td>I3 &lt; I4</td>
<td>S3</td>
<td></td>
</tr>
</tbody>
</table>

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 characteristics 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.

1. \( I = 1 \)
2. \( J = 1 \)
3. \( I = I + 1 \)
4. \( E \)
5. \( B \)
6. \( A \)
7. \( J = J + 1 \)
8. \( RETURN \)

 Activate the job scheduler either when a new job comes or when a partition becomes free.
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 greater than the size of the partition I?

NO  Assign to the job the partition I

YES Are there more jobs in the table?

NO  B

YES 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 priority 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 transferred 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.

<table>
<thead>
<tr>
<th>compiler no.</th>
<th>residence</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Memory</td>
<td>Busy</td>
</tr>
<tr>
<td>2</td>
<td>Disk</td>
<td>Free</td>
</tr>
<tr>
<td>3</td>
<td>Load</td>
<td>Free</td>
</tr>
<tr>
<td>4</td>
<td>Disk</td>
<td>Free</td>
</tr>
<tr>
<td>5</td>
<td>Disk</td>
<td>Free</td>
</tr>
</tbody>
</table>

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 memroy 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.
<table>
<thead>
<tr>
<th>compiler no.</th>
<th>residence</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Memory</td>
<td>Free</td>
</tr>
<tr>
<td>2</td>
<td>Disk</td>
<td>Free</td>
</tr>
<tr>
<td>3</td>
<td>Memory</td>
<td>Busy</td>
</tr>
<tr>
<td>4</td>
<td>Disk</td>
<td>Free</td>
</tr>
<tr>
<td>5</td>
<td>Disk</td>
<td>Free</td>
</tr>
</tbody>
</table>

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 Algorithm of the Compiler Manager

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

loader
Signal from loader that the compiler I is in main memory

\[ T(I,1) = \text{Memory} \]

\[ T(I,2) = \text{Busy} \]

Assign to the compiler I the process which is at the front of the queue \( Q(I) \)

RETURN

A

\[ T(K,2) = \text{Busy} \]

? YES

\[ T(K,1) = \text{Disk} \]

\[ T(M,2) = \text{Busy} \]

? YES

\[ T(M,1) = \text{Disk} \]

Enter the process \( j \) into the queue \( Q(\text{SPES}) \)

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.
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 etc.) 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](image-url)
4.7 CPU Scheduler

The CPU scheduler is a software module in the operating system. Its 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 Highest-Static-priority-First-Served (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.

Hold state: 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 SPOODED onto disk).

Ready state: 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. 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 throughput, 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 throughput 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 transferred, 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.
Return from CPU scheduler
Return from CPU scheduler

(AAA3)

RELEASE  CHAN

PF 25

PRIORITY

ADVANCE FN * PF 21

B
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 throughput. 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 set aside 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,

\[ T_1 (3 \text{ ms}) \]
represents the time required by the CPU to issue a start input-output command to the channel,

\[ T_2 (25 \text{ ms}) \]
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.
<table>
<thead>
<tr>
<th>GPSS Entities</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>482</td>
</tr>
<tr>
<td>Facilities</td>
<td>22</td>
</tr>
<tr>
<td>Storage</td>
<td>1</td>
</tr>
<tr>
<td>Queues</td>
<td>11</td>
</tr>
<tr>
<td>Logics</td>
<td>30</td>
</tr>
<tr>
<td>Tables</td>
<td>14</td>
</tr>
<tr>
<td>Functions</td>
<td>45</td>
</tr>
<tr>
<td>Variables</td>
<td>17</td>
</tr>
<tr>
<td>Fullword saves values</td>
<td>29</td>
</tr>
<tr>
<td>Halfword saves values</td>
<td>7</td>
</tr>
<tr>
<td>Fullword Matrixes</td>
<td>2</td>
</tr>
<tr>
<td>Chains</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 5-1. GPSS entities used by the simulated model
<table>
<thead>
<tr>
<th>JOB CLASSES</th>
<th>CARD READER 1</th>
<th>CARD READER 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>35%</td>
<td>8%</td>
</tr>
<tr>
<td>B</td>
<td>24%</td>
<td>5%</td>
</tr>
<tr>
<td>C</td>
<td>22%</td>
<td>6%</td>
</tr>
<tr>
<td>D</td>
<td>8%</td>
<td>11%</td>
</tr>
<tr>
<td>E</td>
<td>7%</td>
<td>16%</td>
</tr>
<tr>
<td>F</td>
<td>2%</td>
<td>25%</td>
</tr>
<tr>
<td>G</td>
<td>2%</td>
<td>29%</td>
</tr>
</tbody>
</table>

Figure 5-2. Job class distribution among card readers one and two.
<table>
<thead>
<tr>
<th>PARTITION</th>
<th>JOB CLASSES</th>
<th>SIZE (in K bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A, B, C</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>A, B, C</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>C, B, A</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>C, D, E, G</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>D, C, E, G</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>F, G, D, C</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>F, G</td>
<td>40</td>
</tr>
</tbody>
</table>

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).
<table>
<thead>
<tr>
<th>DEVICE</th>
<th>CPU</th>
<th>DEVICE</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input drum</td>
<td>2.99</td>
<td>Output disk</td>
<td>9</td>
</tr>
<tr>
<td>Card reader 1</td>
<td>17</td>
<td>Card reader 2</td>
<td>61</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>1410720</td>
<td>UTILIZATION</td>
<td>99</td>
</tr>
<tr>
<td>NUMBER</td>
<td>5253</td>
<td>ENTRIES</td>
<td>382008</td>
</tr>
<tr>
<td>TIME/TRAN</td>
<td>19494</td>
<td>(in parts per hundred)</td>
<td>42992</td>
</tr>
<tr>
<td>(in milliseconds)</td>
<td>51.290</td>
<td>907</td>
<td>19543.979</td>
</tr>
<tr>
<td>8.161</td>
<td>52.671</td>
<td>15051.731</td>
<td>8578.977</td>
</tr>
</tbody>
</table>

Figure 5-4. Statistics of the devices
<table>
<thead>
<tr>
<th>Device</th>
<th>CPU</th>
<th>Maximum Entries</th>
<th>Total Entries</th>
<th>Zero Entries</th>
<th>Average Time/Tran (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input drum</td>
<td>6</td>
<td>1410721</td>
<td>1306700</td>
<td>5202</td>
<td>1.57</td>
</tr>
<tr>
<td>Output disk</td>
<td>2</td>
<td>19494</td>
<td>18697</td>
<td>5253</td>
<td>10.82</td>
</tr>
<tr>
<td>Input disk</td>
<td>3</td>
<td>42992</td>
<td>37892</td>
<td>11.11</td>
<td>27.15</td>
</tr>
<tr>
<td>Channel</td>
<td>3</td>
<td>314296</td>
<td>310431</td>
<td>4.39</td>
<td>4.05</td>
</tr>
<tr>
<td>Line printer</td>
<td>7</td>
<td>907</td>
<td>424</td>
<td>424</td>
<td>7001.97</td>
</tr>
<tr>
<td>Card reader 1</td>
<td>2</td>
<td>97</td>
<td>72</td>
<td>318</td>
<td>7001.97</td>
</tr>
<tr>
<td>Card reader 2</td>
<td>6</td>
<td>811</td>
<td>72</td>
<td>318</td>
<td>7001.97</td>
</tr>
</tbody>
</table>

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 percent 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].
<table>
<thead>
<tr>
<th>MEMORY FRAGMENTATION (in parts per hundred)</th>
<th>TIMES OF OCCURRENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>1.3</td>
</tr>
<tr>
<td>17</td>
<td>2.4</td>
</tr>
<tr>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>19</td>
<td>2.2</td>
</tr>
<tr>
<td>20</td>
<td>3.4</td>
</tr>
<tr>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>22</td>
<td>3.9</td>
</tr>
<tr>
<td>23</td>
<td>2.6</td>
</tr>
<tr>
<td>24</td>
<td>4.0</td>
</tr>
<tr>
<td>25</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Figure 5-6. Memory fragmentation.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.995</td>
<td>0.822</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>5.244</td>
<td>3.024</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>4.545</td>
<td>2.975</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>1.988</td>
<td>1.475</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>6.250</td>
<td>2.716</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>8.328</td>
<td>5.638</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>2.983</td>
<td>0.446</td>
</tr>
</tbody>
</table>

Figure 5.7. Fragmentation for each PP partition (weighted)
<table>
<thead>
<tr>
<th>N.</th>
<th>SIZE (in K bytes)</th>
<th>MEAN ARGUMENT (in K bytes)</th>
<th>STANDARD DEVIATION (in K bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1.000</td>
<td>0.814</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>5.134</td>
<td>2.921</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>4.545</td>
<td>3.032</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>1.679</td>
<td>1.528</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>2.365</td>
<td>1.805</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>1.659</td>
<td>2.068</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>2.211</td>
<td>1.512</td>
</tr>
</tbody>
</table>

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,

$$\frac{\sum_{j=1}^{N_c} (T_j^{out} - T_j^{in})}{N_c}$$

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
<table>
<thead>
<tr>
<th>CLASS</th>
<th>NUMBER OF JOBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>309</td>
</tr>
<tr>
<td>B</td>
<td>203</td>
</tr>
<tr>
<td>C</td>
<td>171</td>
</tr>
<tr>
<td>D</td>
<td>81</td>
</tr>
<tr>
<td>E</td>
<td>66</td>
</tr>
<tr>
<td>F</td>
<td>35</td>
</tr>
<tr>
<td>G</td>
<td>42</td>
</tr>
</tbody>
</table>

Figure 5-9. Number of jobs in various classes
<table>
<thead>
<tr>
<th>JOB INPUT QUEUE FOR CLASS</th>
<th>AVERAGE CONTENTS (N. of jobs waiting)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.48</td>
</tr>
<tr>
<td>B</td>
<td>3.30</td>
</tr>
<tr>
<td>C</td>
<td>35.60</td>
</tr>
<tr>
<td>D</td>
<td>30.09</td>
</tr>
<tr>
<td>E</td>
<td>28.84</td>
</tr>
<tr>
<td>F</td>
<td>21.53</td>
</tr>
<tr>
<td>G</td>
<td>26.41</td>
</tr>
</tbody>
</table>

Figure 5-10. Average number of jobs waiting in the input queue for various classes.
The times taken by the system to complete the job is 1.367 ms.

The time taken at the service level to print out the job is 0.032 ms.

The job did not have to wait in the output queue for the job to complete.

The input service level for the job is 0.011 ms.

The job completed at the output station at the time 22495 ms.

The CPU time used by the job is 22221 ms.

The main memory used by the job is 16 k bytes.

This job had requested the completion of the integral processing associated by the system at 26.

The external processing associated by the use of the class of the job is 0.

This is a report for the 103 job.
The time taken by the CPU to process the job is 2.57 ms.

The time taken by the compiler to compile the job is 57.59 ms.

The time taken by the card reader to read the job is 33.47 ms.

The job entered at input station 4 at the time 06:42:56.

The CPU time used by the job is 2.57 ms.

The main memory used by the job is 19 K bytes.

The internal priority assigned by the system is 3.5.

The external priority assigned by the system is 4.

The class of the job is 4.

This is a report for the job no.
Figure 6-13. Statistical for Job No. 6

The turnaround time of the job is 7.52 minutes. The time taken by the line printer to print out the job is 1.246 minutes. The job had to wait in the input queue for 3.276 minutes. The time taken by the computer to execute the job is 3.905 minutes. The job card was the card selected at the time 3612.45 seconds after the start of the job. The job arrived at the input station 177.24 minutes after the start of the job. The CPU time used by the job is 47.25 minutes. The main memory used by the job is 15.64 bytes. This job had requested the CPU time 12.37 minutes before the job was assigned. The effective priority assigned by the system is 26. The class of the job is 5. This is a pre-empt for the job.
The class of the job is C
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 14 K bytes
The CPU time used by the job is 254.23 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 124.27 ms
The job had to wait in the input queue for 49 ms
The time taken by the compiler to compile the job is 6.75 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 146.93 ms
The turnaround time of the job is 203.17 ms
This is a report 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 simulation 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 multi-programming. 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
IF(ING) 110, I(2), I(3), I(4), I(5), I(6)
2
FORMAT (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), K
10
RETURN
10
PRINT 110, I(2), I(3), I(4), I(5), I(6)
110
FORMAT (*1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), K
RETURN
120
PRINT 120, I(2), I(3), I(4), I(5), I(6)
120
RETURN
130
PRINT 130, I(2), I(3), I(4), I(5), I(6)
130
RETURN
140
PRINT 140, I(2), I(3), I(4), I(5), I(6)
140
RETURN
150
PRINT 150, I(2), I(3), I(4), I(5), I(6)
150
RETURN
160
PRINT 160, I(2), I(3), I(4), I(5), I(6)
160
RETURN
170
PRINT 170, I(2), I(3), I(4), I(5), I(6)
170
RETURN
180
PRINT 180, I(2), I(3), I(4), I(5), I(6)
180
RETURN
190
PRINT 190, I(2), I(3), I(4), I(5), I(6)
190
RETURN
200
PRINT 200, I(2), I(3), I(4), I(5), I(6)
200
RETURN
210
PRINT 210, I(2), I(3), I(4), I(5), I(6)
210
RETURN
220
PRINT 220, I(2), I(3), I(4), I(5), I(6)
220
RETURN
230
PRINT 230, I(2), I(3), I(4), I(5), I(6)
230
RETURN
240
PRINT 240, I(2), I(3), I(4), I(5), I(6)
240
RETURN
250
PRINT 250, I(2), I(3), I(4), I(5), I(6)
250
RETURN
260
PRINT 260, I(2), I(3), I(4), I(5), I(6)
260
RETURN
270
PRINT 270, I(2), I(3), I(4), I(5), I(6)
270
RETURN
280
PRINT 280, I(2), I(3), I(4), I(5), I(6)
280
RETURN
290
PRINT 290, I(2), I(3), I(4), I(5), I(6)
290
RETURN
300
PRINT 300, I(2), I(3), I(4), I(5), I(6)
300
RETURN
310
PRINT 310, I(2), I(3), I(4), I(5), I(6)
310
RETURN
320
PRINT 320, I(2), I(3), I(4), I(5), I(6)
320
RETURN
330
PRINT 330, I(2), I(3), I(4), I(5), I(6)
330
RETURN
340
PRINT 340, I(2), I(3), I(4), I(5), I(6)
340
RETURN
350
PRINT 350, I(2), I(3), I(4), I(5), I(6)
350
RETURN
360
PRINT 360, I(2), I(3), I(4), I(5), I(6)
360
RETURN
370
PRINT 370, I(2), I(3), I(4), I(5), I(6)
370
RETURN
380
PRINT 380, I(2), I(3), I(4), I(5), I(6)
380
RETURN
390
PRINT 390, I(2), I(3), I(4), I(5), I(6)
390
RETURN
400
PRINT 400, I(2), I(3), I(4), I(5), I(6)
400
RETURN
PRINT 146, I(2), I(3), I(5)
FORMAT(*E+4.3X, * THE TIME TAKEN BY THE CARD READER TO READ THE JOB* 11S, I8, * MS*)
PRINT 115, I(8), * THE JOB HAD TO WAIT IN THE INPUT QUEUE FOR* I(11S, I8, * MS*)
RETURN
GO TO 102
FORMAT(*C*, 4.3X, * THE JOB DID NOT HAVE TO WAIT IN THE OUTPUT QUEUE*)
GO TO 103
PRINT 104, I(4), *THE JOB HAD TO WAIT IN THE OUTPUT QUEUE FOR* I(11S, I8, * MS*)
PRINT 105, I(6), * THE TIME TAKEN BY THE LINE PRINTER TO PRINT OUT THE* RETURN
1F JOB IS*, I(18, * MS*)
PRINT 106, I(2) / 36000000.
RETURN
PRINT 107
FORMAT(*1X, 50X, 45 (-*), */0, 53X, * THIS IS A NON STANDARD JOB* 1E3, X, 45 (+-))
RETURN
Y1 = FLOAT(I(2)) / (FLOAT(I(2)) * 36000000.)
FORMAT(*1X, 40X, X2 (++), */0, 54X, * THE AVERAGE TURNAROUND TIME* 1F3, X, 52 (+-))
RETURN
PRINT 520, I(2), I(3), I(4), I(5), I(6)
FORMAT(*5X, 50X, I(13, 7X))
RETURN
L = 5
GO 999 K = 2, 6
IF(I(4), I(11S) 992
L = M = 1
993 CONTINUE
992 IF(I(2) = 1) RETURN
PRINT 521, I(K), K = 2, 5
521 PRINT (*4X, 50X, I(13, 7X))
RETURN
PRINT 661
661 FORMAT(*1X, 7X, * THE SIZE REQUESTED EXCEEDS THE MAIN MEMORY AVAILABLE TO THE* 1F5, X, 7X)
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 HOLOS 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.
**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.

P13  IS THE TOTAL NUMBER OF INPUT STATEMENTS OF A JOB.

P14  IS THE TOTAL NUMBER OF INPUT DATA BLOCKS FOR A JOB.

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.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P24</td>
<td>IS</td>
<td>THE TIME</td>
<td>TAKEN BY</td>
</tr>
<tr>
<td>P25</td>
<td>SAVES</td>
<td>THE PRIORITY OF A PROCESS WHICH ENTERS THE I/O DEVICE MANAGEMENT.</td>
<td></td>
</tr>
<tr>
<td>P31</td>
<td>IDENTIFIES</td>
<td>THE LINE PRINTER REQUESTED BY THE INPUT STATION. IT CAN TAKE ONE OF THE VALUES 10, 11, 12, 13, 14.</td>
<td></td>
</tr>
<tr>
<td>P32</td>
<td>TAKES</td>
<td>THE VALUE</td>
<td>9 IF</td>
</tr>
<tr>
<td>P33</td>
<td>IDENTIFIES</td>
<td>THE COMPI</td>
<td>LER REQUESTED BY THE JOB. IT CAN TAKE ONE OF THE VALUES 17, 18, 19, 20, 21.</td>
</tr>
<tr>
<td>P35</td>
<td>SAVES</td>
<td>THE ADDRESS</td>
<td>OF</td>
</tr>
<tr>
<td>P36</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P37</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P38</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P39</td>
<td>IS</td>
<td>THE TIME</td>
<td>THAT</td>
</tr>
<tr>
<td>P40</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P41</td>
<td>IS</td>
<td>THE NUMBER</td>
<td>OF</td>
</tr>
<tr>
<td>P46</td>
<td>IS</td>
<td>THE TUR</td>
<td>NAROUND TIME</td>
</tr>
<tr>
<td>P47</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P48</td>
<td>IS</td>
<td>THE TIME</td>
<td>AT WHICH</td>
</tr>
<tr>
<td>P49</td>
<td>IS</td>
<td>THE TIME</td>
<td>TAKEN</td>
</tr>
</tbody>
</table>
**BLOCK NUMBER**

**LOC** OPERATION A, B, C, D, E, F, G, H, I, J

**COMMENTS**

P50 IS THE TIME AT WHICH THE JOB CAPTURES THE LINE PRINTER.

P51 IS THE TIME TAKEN BY THE LINE PRINTER TO PRINT THE JOB.

P52 IDENTIFIES THE NUMBER OF THE JOB IN THE AUXILIARY REPORT GENERATION.

P53 IS THE TIME AT WHICH THE JOB ENTERS THE OUTPUT QUEUE.

P54 IS THE TIME THAT A JOB HAS TO WAIT IN THE OUTPUT QUEUE.

**EQUIVALENT STATEMENTS**

**THESE ARE DECLARED EXPLICITLY TO AVOID INTERFERENCE WITH THE BLOCKS**

CHANG

CHAN

WAIT1

READY

CPU

AVG1

AVG2

TIME

LOAD0

LOAD1

SPEC

**SAVEVALUE**

**INITIALIZATIONS**

AVERAGE JOBS ARRIVAL TIME AT INPUT STATION 1.

INITIAL XF$AVG1, 15000
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A,B,C,D,E,F,G,H,I,J</td>
<td></td>
</tr>
</tbody>
</table>

AVERAGE JOBS ARRIVAL TIME AT INPUT STATION 2.
INITIAL XF$AVG2,120000

THE TIMER TRANSACTION ARRIVES AT TIME XF$TIME.
INITIAL XF$TIME,11551400

JOBS WHICH REQUEST CPU TIME OF MORE THAN 1000000MS 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 XF20,140

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
   - MATRIX MX, 7, 5

2. α
   - MATRIX MX, 20, 5
**MATRIX SAVE VALUE 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),21
INITIAL MX1(5,1),7
INITIAL MX1(5,2),3/MX1(5,3),5/MX1(5,5),33
INITIAL MX1(5,4),7
INITIAL MX1(5,1),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** SBMEMOR,7

**VARIABLE DEFINITIONS**

**V1 IS ASSIGNED THE NUMBER OF INPUT DATA CARDS.**

```
VARIABLE (PF1*FN*PF10)/100
```

**V2 IS ASSIGNED THE NUMBER OF OUTPUT BLOCKS.**

```
VARIABLE PF9/10+1
```

**V4 IS ASSIGNED THE NUMBER OF INPUT STATEMENTS.**

```
VARIABLE PF1-PF11
```
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>OPERATION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.b</td>
<td>LOC</td>
<td>VE IS ASSIGNED THE NUMBER OF INPUT BLOCKS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE PF11/10+1</td>
</tr>
<tr>
<td>6.b</td>
<td></td>
<td>VE IS ASSIGNED THE NUMBER OF OUTPUT LINES WHICH INCLUDES THE SOURCE LISTING OF THE PROGRAM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE PF9+PF13+5G</td>
</tr>
<tr>
<td>7.b</td>
<td></td>
<td>V7 IS THE MEAN OF I/O INTERREQUEST INTERVALS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE PF8/(PF12+PF14)</td>
</tr>
<tr>
<td>8.b</td>
<td></td>
<td>VE IS THE MAIN MEMORY THAT IS NOT USED IN THE PARTITION XF9.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE XF11-PF7</td>
</tr>
<tr>
<td>9.b</td>
<td></td>
<td>V9 IS THE FRAGMENTATION OF THE MAIN MEMORY EXPRESSED IN PARTS PER THOUSAND.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE (XF13+XF14+XF15+XF17+XF18+XF19)*1000/XF20</td>
</tr>
<tr>
<td>10.b</td>
<td></td>
<td>V10 IS ASSIGNED THE NUMBER OF INPUT BLOCK TRANSFERS THAT THE COMPILER NEEDS IN ORDER TO COMPILE THE PROGRAM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE PF13/50+1</td>
</tr>
<tr>
<td>12.b</td>
<td></td>
<td>V12 IS THE NUMBER OF STANDARD JOBS WHICH HAVE BEEN ENTERED INTO THE SYSTEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE NSWAIT=CH3WAIT1</td>
</tr>
<tr>
<td>14.b</td>
<td></td>
<td>V13 IS THE TOTAL CORE SIZE AVAILABLE TO THE USERS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE VX1(1,5)+MX1(2,5)+MX1(3,5)+MX1(4,5)+MX1(5,5)+MX1(6,5)</td>
</tr>
<tr>
<td>13.b</td>
<td></td>
<td>VARIABLE V14+MX1(7,5)</td>
</tr>
</tbody>
</table>
BOOLEAN VARIABLE DEFINITIONS

1  BVARIALE  PF2J#E#1+PF26#E#11
2  BVARIALE  XH*PF3#E#16*PF1#E#1
3  BVARIALE  CH9#LE#5*PF32#E#9
4  BVARIALE  W3HAT#E#E#F3#E#0*F4#E#0
5  BVARIALE  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  TABLE  PF42, 6, 2, M42

TABLE 2 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 2 WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
2  TABLE  PF42, 5, 2, M42

TABLE 3 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 3 WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
3  TABLE  PF42, 3, 2, M42

TABLE 4 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 4 WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
4  TABLE  PF42, 0, 2, M42

TABLE 5 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 5 WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS PARTITION.
TABLE 5 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 6 WEIGHTED BY THE TIME A JOB OCCUPIES THIS PARTITION.

TABLE 6 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 7 WEIGHTED BY THE TIME A JOB OCCUPIES THIS PARTITION.

TABLE 7 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 8 (IT IS NOT WEIGHTED).

TABLE 8 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 9 (IT IS NOT WEIGHTED).

TABLE 9 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 10 (IT IS NOT WEIGHTED).

TABLE 10 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 11 (IT IS NOT WEIGHTED).

TABLE 11 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 12 (IT IS NOT WEIGHTED).

TABLE 12 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 13 (IT IS NOT WEIGHTED).

TABLE 13 IS USED TO TABULATE THE FRAGMENTATION OF PARTITION 14 (IT IS NOT WEIGHTED).
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>FUNCTION DEFINITIONS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FUNCTION 1 RETURNS THE CLASS FOR A JOB ARRIVING AT INPUT STATION 1.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>FUNCTION 1 = 77</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>75, 1, 59, 2, 82, 3, 69, 2, 96, 5, 98, 5, 1, 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FUNCTION 2 RETURNS THE CLASS FOR A JOB ARRIVING AT INPUT STATION 2.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>FUNCTION 2 = 77</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>79, 7, 5, 7, 3, 1, 5, 61, 5, 97, 3, 92, 7, 1, 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXTERNAL PRIORITY OF THE JOB ASSIGNED BY THE USER.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td>FUNCTION 3 = 75</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40, 7, 5, 7, 5, 2, 9, 1, 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAIN MEMORY USED BY THE JOB (S).</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>FUNCTION 4 = 76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5, 7, 4, 1, 7, 9, 1, 2, 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAIN MEMORY USED BY THE JOB (M).</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td>FUNCTION 5 = 76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0, 22, 55, 26, 1, 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAIN MEMORY USED BY THE JOB (L).</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td>FUNCTION 6 = 73</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0, 22, 7, 4, 1, 32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CPU TIME USED BY THE JOB (S).</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>FUNCTION 7 = 71</td>
<td></td>
</tr>
<tr>
<td>BLOCK NUMBER</td>
<td>LOC</td>
<td>OPERATION</td>
<td>A, B, C, D, E, F, G, H, I, J</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>-----------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMBER OF OUTPUT LINES (L2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>FUNCTION</td>
<td>PN3, 09</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>NUMBER OF OUTPUT LINES (L7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>FUNCTION</td>
<td>PN2, 09</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>NUMBER OF DATA CARDS GIVEN AS THE PERCENTAGE OF INPUT CARDS (S1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>FUNCTION</td>
<td>PN1, 06</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>NUMBER OF DATA CARDS GIVEN AS THE PERCENTAGE OF INPUT CARDS (L)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>FUNCTION</td>
<td>PN1, 06</td>
<td></td>
</tr>
</tbody>
</table>

FUNCTION 20 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 5 AND IS USED BY THE CPU SCHEDULE.

FUNCTION 21 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 17 AND IS USED BY THE PROCESS SCHEDULE.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FUNCTION 22 SELECTS A LOCATION BASED UPON THE VALUE OF PARAMETER 20 AND IS USED BY THE I/O DEVICE MANAGEMENT.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>FUNCTION PF24, 4</td>
<td>1/132F1/2, 9DPE/3, 03/4/5, 9B34/5, 9BB6/6, 9BR6/7, 9BB7/8, 9BB8/9, 9BB9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION 23 RETURNS THE ACCESS TIME FOR THE DRUM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>FUNCTION RN2, 12</td>
<td>0, 1, 0/1, 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DUMMY FUNCTION WHICH RETURNS THE VALUE ZERO.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>FUNCTION 1, 02</td>
<td>1/0/2, 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION 25 RETURNS THE TIME FOR THE HEAD OF THE DISK TO MOVE INTO THE REQUIRED POSITION.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>FUNCTION RN3, 02</td>
<td>0, 5/1, 95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION 26 RETURNS THE TIME REQUIRED FOR THE BEGINNING OF THE TRACK WILL TAKE TO COME UNDER THE HEAD OF THE DISK.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>FUNCTION RN4, 02</td>
<td>0, 0/1, 921</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION XF9, 17</td>
<td>1, 13/2, 14/3, 15/4, 16/5, 17/6, 18/7, 19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION 29 IS USED TO CHANGE THE LEVEL OF PRIORITIES.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FUNCTION 32 RETURNS THE TIME REQUIRED BY THE COMPILER TO COMPILE A
BLOCK OF SOURCE CODE.

FUNCTION 31 IS USED BY THE INPUT SPOOLER.

FUNCTION SHCRS PROVIDES A CORRESPONDENCE BETWEEN THE CARD READERS
AND THEIR RESPECTIVE CONTROL SWITCHES.

FUNCTION LOGIC IS USED BY THE JOB SCHEDULER TO RETURN VALUES FOR THE
LOGIC SWITCHES.

FUNCTION NCA1 RETURNS THE NUMBER OF INPUT CARDS FOR JOBS THAT
ARRIVE AT INPUT STATION 1.

FUNCTION NCA2 RETURNS THE NUMBER OF INPUT CARDS FOR JOBS THAT
ARRIVE AT INPUT STATION 2.
* BLOCK NUMBER    * LOC   OPERATION  A, B, C, D, E, F, G, H, I, J   COMMENTS
*               *       *        *         *                              *  
* FUNCTION MORSH IS USED BY THE JOBS IN CLASS G.  *  
36 MORSF FUNCTION RN1,02  
     56/30,51.1,9

* FUNCTION LORMM IS USED BY JOBS IN CLASS G.  *  
37 LO=MM FUNCTION RN2,02  
     56/35,61,1,1,9

* FUNCTION SOURC IS USED BY THE COMPILER.  *  
38 SOURC FUNCTION RN1,06  
     05,2/2,4/3,6/5,55,8/85,10/1,1,12

* PLOCK FUNCTION PF33,05  
39 17,20/16,10/9,5/20,15/21,100

* FUNCTION ADDPS RETURNS A LOCATION BASED UPON THE VALUE OF PARAMETER 3F AND IS USED BY THE LOADER.  *  
40 ADDPS FUNCTION PF35,12  
     1, NEXBL/2, XXX2

* FUNCTION COMP RETURNS THE NAME OF THE COMPILER REQUESTED BY A JOB.  *  
41 COMP FUNCTION RN5,05  
     50,17/52,16/92,19/37,20/1,1,21

* FUNCTION TABL IS USED TO GIVE THE NAMES TO THE TABLES 8, 9, 10, 11, 12, 13, 14.  *  
42 TABL FUNCTION PF41,17  
     1/3, 9/3, 14/74, 11/5, 12/6, 13/7, 14

* FUNCTION CORSP PROVIDES A CORRESPONDENCE BETWEEN THE COMPILERS AND THEIR RESPECTIVE CONTROL SWITCHES.  *  
43 CORSP FUNCTION PF33,05  
     17,23/16,24/19,25/20,25/21,27

FUNCTION LINPE PROVIDES A CORRESPONDENCE BETWEEN THE LINE PRINTERS AND THEIR RESPECTIVE CONTROL SWITCHES.

LINPE FUNCTION, PF1, D6.
9, 39/11, 41/12, 41/13, 41/14, 44

* EXPONENTIAL DISTRIBUTION FUNCTION.

XPODIS FUNCTION, 5N8, 024
0.01, 0.1, 0.9, 2.2, 2.8, 3.2, 3.5, 3, 5, 4, 5, 6, 7, 6, 8, 9, 15, 7.1, 2.7, 5, 1.38
6.5, 1.5, 0.1, 0.0, 0.1, 0.2, 0.2, 0.3, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 0.99, 1, 0.98, 0.9, 0.97, 0.96, 0.95, 0.94, 0.93, 0.92, 0.91, 0.9, 0.89, 0.88, 0.87, 0.86, 0.85, 0.84, 0.83, 0.82, 0.81, 0.8, 0.79, 0.78, 0.77, 0.76, 0.75, 0.74, 0.73, 0.72, 0.71, 0.7, 0.69, 0.68, 0.67, 0.66, 0.65, 0.64, 0.63, 0.62, 0.61, 0.6, 0.59, 0.58, 0.57, 0.56, 0.55, 0.54, 0.53, 0.52, 0.51, 0.5, 0.49, 0.48, 0.47, 0.46, 0.45, 0.44, 0.43, 0.42, 0.41, 0.4, 0.39, 0.38, 0.37, 0.36, 0.35, 0.34, 0.33, 0.32, 0.31, 0.3, 0.29, 0.28, 0.27, 0.26, 0.25, 0.24, 0.23, 0.22, 0.21, 0.2, 0.19, 0.18, 0.17, 0.16, 0.15, 0.14, 0.13, 0.12, 0.11, 0.1, 0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01, 0, 0

***************
***************
***************
***************

***************
***************
***************
***************

***************
***************
***************
***************

INPUT STATION 2

JCBS ARRIVE AT INPUT STATION 2.

1

GENERATE \$AVS2, FN$XPCIS, Z, 54PF, 4PH

IF IT IS TIME TO CLOSE THE COMPUTING CENTER, DO NOT ALLOW ANY FURTHER
JCBS 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
ASSIGN 3,2,PF

ASSIGN LINE PRINTER 1 FOR PRINTING.
ASSIGN 31,10,PF

ASSIGN THE AUXILIARY LINE PRINTER IF REQUESTED.
ASSIGN 32,9,PF

GO TO WAIT FOR THE CARD READER.
TRANSFER ,WAIT

INPUT STATION 1:

JOBS ARRIVE AT INPUT STATION 1.
GENEFATE X$AV51,FN$XPOIS,,5,PF,4PH

IF IT IS TIME TO CLOSE THE COMPUTING CENTER, DO NOT ALLOW ANY FURTHER JOBS TO COME INTO THE SYSTEM.

ASSIGN THE NUMBER OF INPUT CARDS.
ASSIGN 1,FN$NCAP1,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
14  ASSIGN LINE PRINTER 1 FOR PRINTING.
14  ASSIGN  31,10,PF
15  ASSIGN THE AUXILIARY LINE PRINTER IF REQUESTED.
15  ASSIGN  32,9,PF

THE CARD READER:
16  WAIT FOR THE CARD READER.
16  WAIT QUEUE PF2
17  SAVEVALUE  25+,1,XF
17  ASSIGN TO PF52 THE ID OF THE JOB.
18  ASSIGN  52,XF25,PF
19  MARK THE TIME AT WHICH THE JOB ARRIVES.
19  MARK  47PF
20  CAPTURE THE CARD READER.
20  SEIZE PF2
20  MARK THE TIME AT WHICH THE JOB CAPTURED THE CARD READER.
<table>
<thead>
<tr>
<th>BLOCK</th>
<th>LOG</th>
<th>OPERATION</th>
<th>A,B,C,D,E,F,G,H,I,J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MARK</td>
<td>36PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVE THE NUMBER OF INPUT CARDS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN</td>
<td>26,PF1,PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INDICATES THAT WE ARE READING THE JOB CARD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOGIC S</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAVE THE QUEUE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEPART</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>THE CARD READER HAS PRIORITY 30.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AGAIN PRIORITY 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIME TAKEN BY THE CARD READER TO READ A CARD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADVANCE</td>
<td>60,1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IF LOGIC SWITCH IS SET, THEN WAIT SINCE THE INPUT DRUM IS FULL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACK GATE LR</td>
<td>FN1SWCRS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GOTO INPUT SPOOLER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSFER</td>
<td>INPSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETURN FROM INPUT SPOOLER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IF THERE ARE MORE CARDS LEFT GO BACK TO READ THE NEXT CARD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CARDF LOOP</td>
<td>1PF,AGAIN</td>
<td></td>
</tr>
</tbody>
</table>
LOC  OPERATION A,B,C,D,E,F,G,H,I,J

LEAVE THE CARD READER.

RELEASE PF2

ASSIGN TO PF 37 THE TIME TAKEN BY THE CARD READER TO READ A JOB.

ASSIGN 37,HP36PF,PF

MARK THE TIME AT WHICH THE JOB ENTERS THE INPUT QUEUE.

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.

TEST NE PF10, J,FFF

OPEN THE GATE FOR THE JOB SCHEDULER TRANSACTION.

CONTEXT 15

XF2 IS A COUNTER THAT COUNTS THE NUMBER OF JOBS IN THE CHAINS 1,2,3,4
5,6,7.

SAVEVALUE 2+,1,X

INSERT THE JOB INTO THE CHAIN PF5.

LINK PF5,5PF

THIS CHAIN IS FOR NON STANDARD JOBS.

FFF LINK WAIT1,6PF

INPUT SPOOLER.
THE INPUT SPOOLER HAS PRIORITY 60.

38 INPS 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.

TRANSFER ,DMAN

REQUEST THE CPU TO EXAMINE THE CONTENTS OF THE CARD.

SAVE RETURN ADDRESS.

41 BBB1 ASSIGN 17,1,PF

PF13 IS THE TIME TAKEN BY THE INPUT SPOOLER TO EXAMINE THE CONTENTS OF THE CARD.

ASSIGN 18,1,PF

TRANSFER TO THE CPU SCHEDULER.

TRANSFER ;PROSC

RETURN FROM CPU SCHEDULER AND I/O DEVICE MANAGEMENT.

Now examine to see if the card read is the job card.

45 AAA1 GATE L0 PF2,F1,SC

Assignment L2,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 ',CAHR

Increase by one the contents of the buffer of the spooler which corresponds to the card reader under 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 8V2,0,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  LOG  OPERATION  A, B, C, D, E, F, G, H, I, J  COMMENTS

THE TRANSFER TIME IS 44S.

ASSIGN  24, 4, PF

TRANSFER ,DISK

RETURN FROM THE I/O DEVICE MANAGEMENT.
OPEN THE GATE SINCE THE TRANSFER IS OVER.

R399  LOGIC 3  FN31

NOW THE BUFFER OF THE INPUT SPOOLER IS EMPTY.
SAVEVALUE  PF3, 4, XH

DESTROY THE PROCESS.
TERMINATE

TRANSFER TO FIRSC WHENEVER THE CARD UNDER EXAMINATION IS THE JOB CARD.

FIRSC  ASSIGN  4, 1, PF

TEST TO SEE IF THE INPUT DISK IS FULL.
TEST G  XF1, 400, XFULL

IF THE INPUT DISK IS FULL CLOSE THE GATE.
LOGIC 5  FN35HCFS

RETURN TO THE CARD READER.
TRANSFER ,BACK
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>SWAP</th>
<th>OPERATION</th>
<th>A,B,C,D,E,F,G,H,I,J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td></td>
<td>INC</td>
<td></td>
<td>INCREASE THE ENTRIES IN THE INPUT DISK BY ONE.</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>SAVEVALUE</td>
<td>1,2,3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PF5</td>
<td></td>
<td>PF5 IS THE CLASS OF THE JOB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN</td>
<td>5, FN3, PF</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>PF6</td>
<td></td>
<td>PF6 IS THE PRIORITY OF THE JOB. THIS IS AN EXTERNAL PRIORITY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN</td>
<td>5, FN3, FF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PF33</td>
<td></td>
<td>PF33 IS THE NAME OF THE COMPILER REQUESTED BY THE JOB.</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td>ASSIGN</td>
<td>33, FN3COMP, PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSFER</td>
<td></td>
<td>TRANSFER TO THE CLASS IN WHICH THE JOB BELONGS.</td>
</tr>
<tr>
<td>67</td>
<td></td>
<td>FN, 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS A.</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>CLASA</td>
<td>7, FN4, PF</td>
<td>PF7 IS THE MAIN MEMORY USED BY THE JOB.</td>
</tr>
<tr>
<td>69</td>
<td></td>
<td>ASSIGN</td>
<td>9, FN7, PF</td>
<td>PF8 IS THE CPU TIME USED BY THE JOB.</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>ASSIGN</td>
<td>9, FN11, PF</td>
<td>PF9 IS THE NUMBER OF OUTPUT RECORDS (OUTPUT LINES).</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td>ASSIGN</td>
<td>10, 17, PF</td>
<td></td>
</tr>
</tbody>
</table>
THIS SECTION IS FOR THE JOBS WHICH BELONG TO CLASS B.

PF7 IS THE MAIN MEMORY USED BY THE JOB.

CLASS ASSIGN 7, F14, PF

PF8 IS THE CPU TIME USED BY THE JOB.

ASSIGN 8, F13, PF

PF9 IS THE NUMBER OF OUTPUT RECORDS (OUTPUT LINES).

ASSIGN 9, F15, PF

ASSIGN 10, 19, PF

TRANSFER , AAA

THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS C.

CLASS ASSIGN 16, FNMORSM, PF

PF7 IS THE MAIN MEMORY USED BY THE JOB.

ASSIGN 7, FN*PF16, PF

PF8 IS THE CPU TIME USED BY THE JOB.

ASSIGN 8, F13, PF
LOC OPERATION A, B, C, D, E, F, G, H, I, J

PF9 IS THE NUMBER OF OUTPUT RECORDS (OUTPUT LINES).

ASSIGN 9, FN13, PF

ASSIGN 10, 16, PF

TRANSFER , AAA

THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS 0.

PF7 IS THE MAIN MEMORY USED BY THE JOB.

CLASD ASSIGN 7, FN5, PF

PF8 IS THE CPU TIME USED BY THE JOB.

ASSIGN 8, FN8, PF

PF9 IS THE NUMBER OF OUTPUT RECORDS (OUTPUT LINES).

ASSIGN 9, FN14, PF

ASSIGN 10, 13, PF

TRANSFER , AAA

THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS E.

PF7 IS THE MAIN MEMORY USED BY THE JOB.

CLASE ASSIGN 7, FN5, PF

PF8 IS THE CPU TIME USED BY THE JOB.

ASSIGN 8, FN3, PF
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td></td>
<td>ASSIGN</td>
<td>9, FN15, PF</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td></td>
<td>ASSIGN</td>
<td>10, 19, PF</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td></td>
<td>TRANSFER</td>
<td>, AAA</td>
<td></td>
</tr>
</tbody>
</table>

**THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS F.**

<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>94</td>
<td></td>
<td>CLASSF</td>
<td>ASSIGN 7, FN5, PF</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td></td>
<td>PF8</td>
<td>ASSIGN 8, FN15, PF</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td></td>
<td>PF9</td>
<td>ASSIGN 9, FN16, PF</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td></td>
<td>ASSIGN</td>
<td>10, 13, PF</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td></td>
<td>TRANSFER</td>
<td>, AAA</td>
<td></td>
</tr>
</tbody>
</table>

**THIS SECTION IS FOR JOBS WHICH BELONG TO THE CLASS G.**

<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td></td>
<td>CLASSG</td>
<td>ASSIGN 10, FN3LORHM, PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PF7</td>
<td>IS THE MAIN MEMORY USED BY THE JOB.</td>
<td></td>
</tr>
<tr>
<td>BLOCK</td>
<td>OPERATION</td>
<td>COMMENTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>ASSIGN 7, FN*PF10, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PF8 IS THE CPU TIME USED BY THE JOB.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>ASSIGN 8, FN10, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PF9 IS THE NUMBER OF OUTPUT RECORDS (OUTPUT LINES).</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>ASSIGN 9, FN12, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>ASSIGN 10, 17, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHECK TO SEE IF THE JOB HAS REQUESTED MORE MEMORY THAN AVAILABLE TO THE USERS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>AAA TEST LE PF7, XF20, PEGEC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PF11 IS THE NUMBER OF DATA CARDS.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>ASSIGN 11, VI, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PF12 IS THE NUMBER OF THE OUTPUT BLOCKS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>ASSIGN 12, V2, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PF13 IS THE NUMBER OF INPUT STATEMENTS.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>ASSIGN 13, V4, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PF14 IS THE NUMBER OF INPUT BLOCKS.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>ASSIGN 14, V5, PF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PF15 IS THE NUMBER OF THE OUTPUT LINES WHICH INCLUDES THE SOURCE LISTING OF THE PROGRAM.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ASSIGN THE I/O INTERREQUEST INTERVALS.
Pf16 is the mean of the I/O INTERREQUEST INTERVALS.

ASSIGN 16, V7, PF

IF THE MEAN IS LESS THAN ONE GOTO BBB

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.

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.

TEST LE PF7, X4, EEE

SIGNAL THE CARD READER TO READ THE NEXT CARD.

LOGIC R PF2

AT THE SAME TIME REQUEST THE CHANNEL TO TRANSFER THE INPUT QUEUE

SPLIT 1, GO3

TABLE OF THE CLASS IN WHICH THE JOB BELONGS FROM THE INPUT DRUM TO

TRANSFER CARDP

THE WORK AREA OF THE INPUT SPOOLER.

CLOSE THE GATE UNTIL THE TRANSFER IS OVER.

LOGIC S FN31
REQUEST THE CHANNEL.
SAVE RETURN ADDRESS.

118
ASSIGN 26,12,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
DBB12 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 ,P#OSC

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
THE TRANSFER IS OVER AND OPEN THE GATE.

127 BBB2 LOGIC R FN31

DETERMINE THE PROCESS.

128 TERMINATE

TRANSFER TO REJEC WHENEVER THE JOB IS REJECTED BY THE SYSTEM.

129 REJEC ASSIGN 45,1,PF

130 SAVEVALUE 25-,1,XF

131 HELPA REPORT,PF45,XF25,PF7,XF20,PF2,PF3

132 TERMINATE

ASSIGN ALL THE I/O INTERREQUEST 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.
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 23,21,PF

GOTO THE I/O DEVICE MANAGEMENT.
*LOC  OPERATION  A,B,C,D,E,F,G,H,I,J
*  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.
147  ASSIGN  17,2,PF

TIME REQUIRED TO EXAMINE THE MESSAGE.
148  ASSIGN  18,1,PF

TRANSFER TO THE CPU SCHEDULER.
149  TRANSFER  ,PROCS

RETURN FROM CPU SCHEDULER.
150  AAA2  PRIORITY  PF2F

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.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
<td></td>
<td>TEST E</td>
<td></td>
<td>3V1, J, CAPD</td>
</tr>
<tr>
<td>152</td>
<td></td>
<td>QUEUE</td>
<td></td>
<td>PF4</td>
</tr>
<tr>
<td>153</td>
<td></td>
<td>OBTAIN THE DASD</td>
<td></td>
<td>PF4</td>
</tr>
<tr>
<td>154</td>
<td></td>
<td>SEIZE</td>
<td></td>
<td>PF4</td>
</tr>
<tr>
<td>155</td>
<td></td>
<td>LEAVE THE QUEUE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>156</td>
<td></td>
<td>DEPAFT</td>
<td></td>
<td>PF4</td>
</tr>
<tr>
<td>157</td>
<td></td>
<td>OBTAIN THE CHANNEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>158</td>
<td></td>
<td>SEIZE</td>
<td></td>
<td>CHAN</td>
</tr>
<tr>
<td>159</td>
<td></td>
<td>LEAVE THE QUEUE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td></td>
<td>DEPAFT</td>
<td></td>
<td>CHAN1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRIORITY</td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

REQUEST THE CPU TO ISSUE THE START I/O COMMAND.
SAVE RETURN ADDRESS.
ASSIGN 17,3,PF
ASSIGN 18,3,PF
TRANSFER TO THE CPU SCHEDULER.

TRANSFER, PROSC

RETURN FROM CPU SCHEDULER.

RETURN THE CHANNEL

RELEASE CHAN

PRIORITY OF THE PROCESS.

PRIORITY PF25

IF THE PROCESS HAS REQUESTED A DISK THEN THIS REPRESENTS THE TIME TAKEN BY THE ARM TO MOVE TO THE REQUIRED TRACK.

ADVANCE FN*PF21

THIS REPRESENTS THE TIME THAT THE BEGINNING OF THE TRACK MUST TAKE COME UNDER THE HEAD.

ADVANCE FN*PR22

IF THE CHANNEL IS BUSY, SEND THE TRANSACTION TO THE ADVANCE BLOCK AT LOCATION BUSY FOR ONE FULL REVOLUTION OF THE OPUM.

TRANSFER BOTH, ALIK, BUSY

DETAIN THE CHANNEL

SEIZE CHAN

PRIORITY EQ

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
171    TRANSFER TO THE CPU SCHEDULER.
        TRANSFER    ,PROC
        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
WAIT FOR THE CHANNEL.

178. CARD QUEUE CHAN.

OBTAIN CHANNEL.

179. SEIZE CHAN

LEAVE QUEUE

180. DEPART CHAN

181. PRIORITY 80

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 .,PROC.

IF THE PROCESS HAS PRIORITY 80 DO NOT INSERT IT INTO THE QUEUE.
**LOC**  
**OPERATION A, B, S, O, E, F, G, H, I, J**  
**COMMENTS**

185  
PROSC  
TEST NE PR, R3, ERGEN

*SAVE THE PRIORITY OF THE PROCESS.*

186  
ASSIGN  
19, PR, PF

*THE PROCESS SCHEDULER HAS PRIORITY 7G.*

187  
PRIORITY  
7G

*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.*
**LOC**  OPERATION, A, B, O, D, E, F, G, H, I, J  **COMMENTS**

194 **EPGEF**  QUEUE   READY,

   ALLOCATE THE CPU TO THE PROCESS.

195        PREEMPT   CPU, PR.

   LEAVE THE QUEUE.

196        DEPART   READY

   EXECUTE THE PROCESS.

197        ADVANCE   PF13

   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   ,,,1,50,52PF

   SEE IF THERE ARE JOBS WAITING FOR SCHEDULING.

201        BEGIN   TEST G   X2, 0

   SEE IF THE MEMORY IS FULL.
IF MEMORY IS FULL OR IF THERE ARE NO JOBS WAITING FOR SCHEDULING,
WAIT TRANSFER SIM,CAPT,BEGIN
REQUEST THE CPU TO EXECUTE THE JOB SCHEDULER.
SAVE RETURN ADDRESS.
CAPT ASSIGN 17,6,PF
PF16 IS THE TIME EQUIPED TO EXECUTE THE JOB SCHEDULER.
ASSIGN 18,1,PF
TRANSFER TO THE CPU SCHEDULER.
TRANSFER PROSC
RETURN FROM CPU SCHEDULER.
INITIALIZE X9.
X9 IS THE NUMBER OF THE PARTITION UNDER EXAMINATION.
AAA5 SAVEVALUE 9,C,X
OUTER SAVEVALUE 9+,1,X
IF ALL PARTITIONS ARE EXAMINED GOTO OUT.
TEST LE X9,X9,OUT
IF THIS PARTITION IS NOT FREE EXAMINE THE NEXT PARTITION.
GATE LR FM3LOGIC,OUTER
LOC  OPERATION  A,B,C,D,E,F,G,H,I,J  COMMENTS

* INITIALIZE XI1,
* XI1 IS THE COLUMN OF THE MATRIX 1.
211  *  SAVEVALUE  10,0,X

* ASSIGN TO XI1 THE SIZE OF THE PARTITION X9.
212  *  SAVEVALUE  11,MX1(X9,5),X

213  *  INNER  SAVEVALUE  16+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  XI6,4,OUTER

* XI2 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  XI2,0,OUTER

* SEE IF THE JOB QUEUE TABLE FOR THE CLASS XI2 IS EMPTY.
217  *  TEST NE  CH*XI2,0,INNER

* ISSUE AN I/O PROCESS TO THE I/O DEVICE MANAGEMENT.
* ITS TASK IS TO MOVE THE TABLE FOR THE CLASS XI2 FROM THE INPUT
* OF THE TABLE ARE EXAMINED.
* SAVE THE RETURN ADDRESS.
218  *  ASSIGN  26,3,PF

* INPUT DRUM IS NEEDED.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>4, B, C, 0, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>219</td>
<td></td>
<td>ASSIGN</td>
<td>4, 1, PF</td>
<td></td>
</tr>
</tbody>
</table>

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          | B883| ASSIGN    | 17, 7, PF                       |          |

PF19 is the CPU time required to examine the entries of the table.

223          |     | ASSIGN    | 18, 2, PF                       |          |

224          |     | TRANSFER  | , PPOS C                        |          |

RETURN FROM CPU SCHEDULER.
SELECT A JOB FROM THE CLASS X12 THAT FITS INTO THE PARTITION UNDER EXAMINATION.

225          | AAA7| UMLINK    | 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  

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,VE,X  

243  

ASSIGN 42,XF*PF1,PF  

IF MEMORY IS FULL COMPUTE THE FRAGMENTATION.  

244  

GATE SNF MEMR,FRAG  

NOW WE ARE READY TO COMPILE AND LOAD THE PROGRAM.  

TRANSFER TO THIS POINT WHENEVER A COMPILER IS REQUESTED.  

245  

FINIS PRIORITY 60  

SAME RETURN ADDRESS.  

246  

ASSIGN 17,13,PF  

247  

ASSIGN 18,2,PF
*LOC  OPERATION  A,3,C,D,E,F,G,H,I,J  COMMENTS
*   TRANSFER TO THE CPU SCHEDULER.
248  TRANSFER  ,PP05C

*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  :PH,22,23,u,XH,NOSPA

*A SEGMENT OF MAIN MEMORY IS FOUND TO LOAD THE COMPILER.
253  SAVEVALUE  PH1,PF33,XH

*REQUEST THE RELOCATABLE LOADEP TO LOAD THE COMPILER.
*INDICATE THAT THE COMPILER IS LOADING.
254  XXX1  SAVEVALUE  PF33,2,XH

255  ASSIGN  34,FN,8,3,BLOCK,PF

*SAVE RETURN ADDRESS
256  ASSIGN  35,1,PF

257  SPLIT  1,LOAD
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>258</td>
<td></td>
<td>LINK</td>
<td>PF33, 41PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>X21 IS THE MEMORY FRAGMENTATION EXPRESSED IN PARTS PER THOUSAND.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259</td>
<td></td>
<td>FPAG</td>
<td>SAVEVALUE 21, V3, X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SAVE THE FRAGMENTATION.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td></td>
<td>MSAVEVALUE 2, XF30, XF31, XF21, HX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>261</td>
<td></td>
<td>SAVEVALUE 31+1, XF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SEE IF THE ROW OF MATRIX 2 IS FULL.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>262</td>
<td></td>
<td>TEST E</td>
<td>XF31, 6, P, COL</td>
<td></td>
</tr>
<tr>
<td>263</td>
<td></td>
<td>SAVEVALUE 31, 1, XF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>264</td>
<td></td>
<td>SAVEVALUE 30+1, XF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SEE IF MATRIX 2 IS FULL.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>265</td>
<td></td>
<td>TEST E</td>
<td>XF30, 21, P, COL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PRINT OUT MATRIX 2.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td></td>
<td>ASSIGN</td>
<td>45, 2, PF</td>
<td></td>
</tr>
<tr>
<td>267</td>
<td></td>
<td>ASSIGN</td>
<td>4, 1, PH</td>
<td></td>
</tr>
<tr>
<td>268</td>
<td></td>
<td>TRP</td>
<td>ASSIGN 2u, MX2(Ph4, 5), PF</td>
<td></td>
</tr>
<tr>
<td>BLOCK</td>
<td>LOC</td>
<td>OPERATION</td>
<td>A, B, C, D, E, F, G, H, I, J</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>------------------------------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>269</td>
<td></td>
<td>HELPA REPORT, PF45, MX2(PH4,1), MX2(PH4,2), MX2(PH4,3), MX2(PH4,4), PF10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>270</td>
<td></td>
<td>ASSIGN</td>
<td>45, 9, PF</td>
<td></td>
</tr>
<tr>
<td>271</td>
<td></td>
<td>ASSIGN</td>
<td>4*, 1, PH</td>
<td></td>
</tr>
<tr>
<td>272</td>
<td></td>
<td>TEST E</td>
<td>PH4, 21, TRP</td>
<td></td>
</tr>
<tr>
<td>273</td>
<td></td>
<td>SAVEVALUE</td>
<td>30, 1, XF</td>
<td></td>
</tr>
<tr>
<td>274</td>
<td></td>
<td>MSAVEVALUE</td>
<td>2, 1-20, 1-5, 0, MX</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td></td>
<td>PCOL TRANSFER</td>
<td>, FINIS</td>
<td></td>
</tr>
</tbody>
</table>

TRANSFER TO THIS POINT WHENEVER THERE IS NO SPACE TO LOAD THE COMPLILES. TEST TO SEE IF THE COMPILERS THAT ARE NOW IN THE MEMORY ARE BUSY.

276    |     | NOSPA TEST E                | XH*XH22, 1, HMM1             |          |
| 277    |     | GATE MU                     | XH22, XM*41                  |          |

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*X423,1,MMM2

282  GATE NU XH23,MMM2

ONE OF THE COMPILEFS IN THE MAIN MEMORY IS FREE.

283  SAVEVALUE XH23,J,XH

284  SAVEVALUE 23,PF33,XH

REQUEST THE LOADER TO LOAD THE COMPILER.

285  TRANSFEP ,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
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  OP  LINK   PF33,41PF,MM5

OBTAIN THE COMPILER.

293  MM5  SEIZE   PF33

294  PRIOFITY   PF2

295  MARK   48PF

PF1 IS THE NUMBER OF I/O REQUESTS.

296  ASSIGN  1,V10,PF

REQUEST THE CHANNEL 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.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>BBA</td>
<td>LOGIC S</td>
<td>FN3G0SP</td>
<td></td>
</tr>
<tr>
<td>302</td>
<td></td>
<td>SPLIT</td>
<td>1, LIST</td>
<td></td>
</tr>
<tr>
<td>303</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td></td>
<td>ASSIGN</td>
<td>18, FN3G, PF</td>
<td></td>
</tr>
<tr>
<td>305</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>306</td>
<td></td>
<td>LIST ASSIGN</td>
<td>20, 7, PF</td>
<td></td>
</tr>
<tr>
<td>307</td>
<td></td>
<td>ASSIGN</td>
<td>4, 8, PF</td>
<td></td>
</tr>
<tr>
<td>308</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>309</td>
<td>BBA</td>
<td>GATE LR</td>
<td>FN3G0RSP</td>
<td></td>
</tr>
</tbody>
</table>

REQUEST THE CPU TO COMPIL 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.

PF18 IS THE TIME REQUIRED TO COMPILE THE BLOCK OF SOURCE CODE.

SAVE RETURN ADDRESS. IN THIS ADDRESS WE WILL RETURN FROM PROCESS SCHEDULER.

GO TO THE PROCESS SCHEDULER.

SAVE RETURN ADDRESS.

REQUEST OUTPUT DISK.

GO TO I/O DEVICE MANAGEMENT.

RETURN FROM I/O DEVICE MANAGEMENT.
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>310</td>
<td></td>
<td>TRANSFER</td>
<td>,ASSIB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETURN FROM PROCESS SCHEDULER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>AAA8</td>
<td>LOGIC P</td>
<td>FNACORSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASSEM</td>
<td>ASSEMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN</td>
<td>1-,1,PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIND THE NUMBER OF STATEMENTS IN THE SOURCE CODE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>314</td>
<td></td>
<td>ASSIGN</td>
<td>34+, FNISOURCE, PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEE IF MORE SOURCE CODE REMAINS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>315</td>
<td></td>
<td>TEST NE</td>
<td>PF1,0, TELOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REQUEST THE CHANNEL TO TRANSFER THE OBJECT CODE PRODUCED BY THE COMPILED ON THE OBJECT CODE BUFFER TO THE INPUT DISK, AND AT THE SAME TIME TRANSFER THE NEXT BLOCK OF THE SOURCE CODE FROM THE INTERNAL SOURCE TO THE SOURCE CODE BUFFER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVE RETURN ADDRESS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>316</td>
<td></td>
<td>ASSIGN</td>
<td>23,4,PF</td>
<td></td>
</tr>
<tr>
<td>317</td>
<td></td>
<td>ASSIGN</td>
<td>24,6,PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REQUEST INPUT DISK.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>318</td>
<td>PANA</td>
<td>ASSIGN</td>
<td>4,2,PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GO TO DEVICE MANAGEMENT.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK NUMBER</td>
<td>LOC OPERATION</td>
<td>A, B, C, D, E, F, G, H, I, J</td>
<td>COMMENTS</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-----------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>319</td>
<td>TRANSFER</td>
<td>DMAN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NC 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 |ielos          | ASSIGN | 20,13,PF |
| 321 | ASSIGN         |        | 24,6,PF   |
| 322 | TRANSFER       |        | MAMA      |

RETURN FROM I/O DEVICE MANAGEMENT.
RETURN THE COMPILER.

| 323 | DB913          | RELEASE | PF33 |

ASSIGN TO PF49 THE COMPILATION TIME OF THE JOB.

| 324 | ASSIGN         |        | L9, MP4, 8PF, PF |
| 325 | UNLINK         |        | PF33, MMHS, 1, 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 | ASSIGN         |        | 35,2,PF    |

THE RELOCATABLE LOADER.
WAIT FOR THE LOADER.

327 LOAD QUEUE LOADQ

OBTAIN THE LOADER.

328 SEIZE LOADQ

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 AGAIN 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 B881L ASSIGN 17,14,PF

335 ASSIGN 18,3,PF
TRANSFER TO THE CPU SCHEDULER.

RETURN FROM CPU SCHEDULER.

LEAVE THE LOADER.

RELEASE LOADER.

TRANSFER FN, ADDRS

DECREASE THE NUMBER OF ENTRIES IN THE INPUT DRUM BY ONE.

TEST TO SEE IF THE INPUT DRUM IS FULL.

OPEN THE GATE.

ASSIGN THE INTERNAL PRIORITY OF THE JOB.

THE JOB IS READY TO BE EXECUTED.

SEE IF THE MEAN OF THE I/O INTERREQUEST INTERVAL (PF16) IS LESS
LOC OPERATION A,B,C,D,E,F,G,H,I,J

* THAN THE CPU TIME REMAINING FOR THE JOB.

345 GEOR TEST L PF16,PF27,INTER

346 ASSIGN 16,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 16,PF27,PF

350 TRANSFER ,MARIA

* RETURN FROM CPU SCHEDULER.
* REDUCE THE CPU TIME OF THE JOB BY ONE I/O INTERVAL.
351 AAA9 ASSIGN 27-,PF18,PF

* TEST TO SEE IF NO INPUT DATA REMAINS.
352 INPUT TEST NE PF14,6,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 | 26, 5, PF | 
356 | | Transfer | , Disk | Go to I/O device management.
357 | BBB5 | Assign | 14, 1, PF | Return from I/O device management.
358 | | Test | if 27, 0, Input | Reduce input blocks by one.
359 | | Transfer | , Geo2 | Test for zero CPU time.
360 | Outp | Test | if 12, 0, Viki | Test to see if no output blocks remain.
361 | | Assign | 4, 6, PF | Obtain the output disk to print out this block.
362 | | Assign | 24, 4, PF | Request output disk.
363 | | Assign | 24, 4, PF | Assign the time required to transfer the block.
* BLOCK NUMBER 4

* LOC OPEATION A, B, C, D, E, F, G, H, I, J

* COMMENS

* SAVE THE RETURN ADDRESS FROM I/O DEVICE MANAGEMENT.

363

* ASSIGN 26, E, PF

* GO TO I/O DEVICE MANAGEMENT.

364

* TRANSFER, DISK

* RETURN FROM I/O DEVICE MANAGEMENT.

* REDUCE THE OUTPUT BLOCKS BY ONE.

365

* ASSIGN 12-1, PF

* TEST FOR ZERO CPU TIME.

366

* TEST ME PF27, O, OUTP

367

* TRANSFER, GEOR

* TEST FOR ZERO CPU TIME.

368

* VIKY TEST E PF27, O, GEOR

* THE JOB HAS COMPLETED EXECUTION AND IS READY TO BE PRINTED.

* OPEN THE GATE FOR THE SCHEDULER TRANSACTION.

369

* LOGIC P 15

* LEAVE THE MEMORY.

370

* LEAVE MEMOR

* RETURN THE PARTITION.

371

* LOGIC P PF28
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>372</td>
<td></td>
<td>TEST NE</td>
<td>PH3=1, JUMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TABULATE THE FRAGMENTATION OF THE PARTITION (THIS IS A WEIGHTED TABLE).</td>
</tr>
<tr>
<td>373</td>
<td></td>
<td></td>
<td>TABULATE PF41,HP40PF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TABULATE THE FRAGMENTATION OF THE PARTITION (THIS IS NOT WEIGHTED).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TABULATE FN8TABL</td>
</tr>
</tbody>
</table>

*************** 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 AAA16 MARK 53PF
**BLOCK NUMBER**  **LOC**  **OPERATION**  **A,B,C,D,E,F,G,H,I,J**  **COMMENTS**

**380**

**IF THE LINE PRINTER IS NOT FREE INSERT THE JOB INTO THE OUTPUT QUEUE.**

**TRANSFER BOTH,LINPF,QLENG**

**381**

**OBTAIN THE LINE PRINTER**

**LINPF SEIZE PF31**

**382**

**MARK 56PF**

**ASSIGN TO PF54 THE TIME THAT THE JOB HAS TO WAIT IN THE OUTPUT QUEUE.**

**ASSIGN 54,MP53PF,PF**

**384**

**SAVE THE NUMBER OF OUTPUT LINES.**

**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.**

**ASSIGN 4,8,PF**

**TRANSFER TIME IS 44S.**

**ASSIGN 24,4,PF**

**387**

**SAVE RETURN ADDRESS.**

**ASSIGN 20,10,PF**

**388**

**GO TO I/O DEVICE MANAGEMENT.**

**TRANSFER ,DISK**
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>389</td>
<td></td>
<td>OLENG</td>
<td>CH*PF31,6,AUXIL</td>
<td>IF MORE THAN 6 JOBS ARE WAITING FOR THE LINE PRINTER GO TO AUXIL.</td>
</tr>
<tr>
<td>390</td>
<td></td>
<td>DELTA</td>
<td>PF31,15PF</td>
<td>TEST TO SEE IF THE AUXILIARY LINE PRINTER IS REQUESTED.</td>
</tr>
<tr>
<td>391</td>
<td></td>
<td>AUXIL</td>
<td>BV3,1,DELTA</td>
<td>IF THE QUEUE OF THE AUXILIARY LINE PRINTER IS LESS THAN 5 ASSIGN THE JOB TO AUXILIARY LINE PRINTER.</td>
</tr>
<tr>
<td>392</td>
<td></td>
<td>ASSIGN</td>
<td>31,9,PF</td>
<td>RETURN FROM I/O DEVICE MANAGEMENT.</td>
</tr>
<tr>
<td>393</td>
<td></td>
<td>GATE U</td>
<td>3,LINPR</td>
<td>TEST IF THE TRANSFERRED BLOCK HAS 10 RECORDS OR LESS.</td>
</tr>
<tr>
<td>394</td>
<td></td>
<td>LINK</td>
<td>3,15PF</td>
<td>THE BLOCK HAS 10 RECORDS.</td>
</tr>
<tr>
<td>395</td>
<td></td>
<td>BBB1C</td>
<td>TEST GE PF29,10,LESS</td>
<td>REDUCE THE NUMBER OF OUTPUT LINES BY PF36</td>
</tr>
<tr>
<td>396</td>
<td></td>
<td>ASSIGN</td>
<td>30,10,PF</td>
<td>REQUEST THE CHANNEL TO TRANSFER A RECORD FROM THE BUFFER OF THE OUTPUT SPOOLER TO THE BUFFER OF THE LINE PRINTER.</td>
</tr>
<tr>
<td>397</td>
<td></td>
<td>THETA</td>
<td>29-,PF30,PF</td>
<td>SAVE RETURN ADDRESS.</td>
</tr>
</tbody>
</table>
**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**  ,OMAN

**THE BLOCK HAS LESS THAN 10 RECORDS.**

400  **LESS**  **ASSIGN**  30,PF2?,PF

**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  **BBBB11**  **TEST E**  0V5,1,PPP1

403  **ASSIGN**  2,-31,PH

404  **SPLIT**  1,PPP2

**TIME TAKEN BY THE LINE PRINTER TO PRINT OUT ONE LINE.**

405  **PPP1**  **ADVANCE**  60

**TEST E**  PH2,-31,PPP3

407  **ASSIGN**  2,0,PH

**LOGIC R**  FNSLINPE

409  **PPP4**  **ASSEMBLE**  2
410 TRANSFER 99910

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 ALLOCATE 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 46,41,PF

THIS SAVEVALUE IS USED TO CALCULATE THE AVERAGE TURNAROUND TIME.

418 SAVEVALUE 26+,?F46,xf
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>419</td>
<td>HELPA</td>
<td>REPORT, PF45, PF52, PF5, PF3, PF2, PF33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>420</td>
<td>ASSIGN</td>
<td>45, 4, PF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>421</td>
<td>HELPA</td>
<td>REPORT, PF45, PF7, PF8, PF3, PF47, PF36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td>ASSIGN</td>
<td>45, 5, PF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>423</td>
<td>HELPA</td>
<td>REPORT, PF45, PF37, PF39, PF54, PF49, PF51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>424</td>
<td>ASSIGN</td>
<td>45, 6, PF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>425</td>
<td>HELPA</td>
<td>REPORT, PF45, PF46, PF52, PF13, PF14, PF15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>426</td>
<td>TRANSFEP</td>
<td>, OKY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>NONST</td>
<td>ASSIGN, 45, 7, PF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>428</td>
<td>HELPA</td>
<td>REPORT, PF45, PF15, PF16, PF12, PF35, PF19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>429</td>
<td>ASSIGN</td>
<td>45, 4, PF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>430</td>
<td>HELPA</td>
<td>REPORT, PF45, PF7, PF8, PF3, PF47, PF36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>431</td>
<td>ASSIGN</td>
<td>45, 5, PF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
432 HELPA REPORT, PF45, PF37, PF39, PF54, PF49, PF51
433 OKY TERMINATE
434 PPP2 ASSIGN 4, 6, PF
435 TRANSFER TIME IS 4MS.
435 ASSIGN 24, 4, PF
436 SAVE RETURN ADDRESS.
436 ASSIGN 20, 15, PF
437 LOGIC 5 FNBLINPE
438 TRANSFER , DISK
439 GATE LR FNBLINPE
440 TRANSFER , PPP4
441 THE TIME ARRIVES AT TIME X$TIME.
441 GENERATE X$TIME,, 1,, 54PF, 4PH
442 SPLIT 1, PETR
DO NOT ALLOW MORE JOBS TO COME INTO THE SYSTEM.

WAIT UNTIL THE QUEUE FOR THE CARD READERS AND THE CARD READERS THEMSELVES ARE FREE.

WAIT UNTIL ALL THE REGULAR JOBS HAVE BEEN SERVED.

PRINT OUT THE FRAGMENTATION MATRIX 2.
<table>
<thead>
<tr>
<th>BLOCK NUMBEP</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>455</td>
<td></td>
<td>TEST NE</td>
<td>PF1, -9, TEPM</td>
<td></td>
</tr>
<tr>
<td>456</td>
<td></td>
<td>TEST NE</td>
<td>CH3WAIT1, 0, TERM</td>
<td></td>
</tr>
</tbody>
</table>

---

**JOB SCHEDULER FOR NON STANDARD JOBS**

* ISSUE AN I/O PROCESS TO THE I/O DEVICE MAN. ITS TASK IS TO TRANSFER 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                   |          |
| 458         |     | ASSIGN    | 4, 2, PF                    |          |
| 459         |     | ASSIGN    | 24, 4, PF                   |          |
| 460         |     | TRANSFER  | DRUM                        |          |
| 461         |     | RETURN    | FROM I/O DEVICE MANAGEMENT  |          |
| 461         |     | REQUEST   | THE CPU TO EXamine THE ENTRIES OF THE TABLE. |          |
| 461         |     | SAVE      | RETURN ADDRESS              |          |
| 461         |     | ASSIGN    | 17, 12, PF                  |          |
**LOC**  OPERATION   A,B,C,D,E,F,G,H,I,J  COMMENTS

462  **ASSIGN:**  16,1,PF

463  **TRANSFER** ,PROSC

RETURN FROM CPU SCHEDULER.
TAKE THE FIRST JOB FROM THE QUEUE AND SCHEDULE IT FOR EXECUTION.

464  AAA12  **UNLINK**  WAIT1,TO=PA,1,,UNT

465  **LOGIC $**

466  **WAIT UNTIL THE MEMORY BECOMES FREE.**

467  **GATE L3**

468  **TRANSFER** ,BBB

ASSIGN TO THIS JOB THE COMPLETE MEMORY WHICH IS AVAILABLE TO THE
USERS.

468  TO=PA  **SAVEVALUE**  3,1, XF

469  **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

472  **ASSIGN**  33,"PF39PF,PF

TRANSFER TO COHLD WHERE WE COMPILE AND LOAD THE JOB.
MASTER UNIV. GPSS V/CDC

<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>472</td>
<td></td>
<td>TRANSFER</td>
<td>,COM-0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WAIT UNTIL ALL THE NON-STANDARD JOBS ARE PRINTED.</td>
<td></td>
</tr>
<tr>
<td>473</td>
<td></td>
<td>TEST E</td>
<td>HEWAIT, NSOKY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>474</td>
<td></td>
<td>ASSIGNED</td>
<td>45, 8, PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>475</td>
<td></td>
<td>HELPA</td>
<td>REPORT, PFL5, XF25, XF26, PF3, PF1, PF2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>476</td>
<td></td>
<td>TERMINATE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>THIS SEGMENT IS TO MAKE SURE THAT THE SUM OF THE SIZE OF THE PARTITIONS DOES NOT EXCEED THE MEMORY AVAILABLE TO THE USERS.</td>
<td></td>
</tr>
<tr>
<td>477</td>
<td></td>
<td>GENERATE</td>
<td>, , , , 1, 76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>478</td>
<td></td>
<td>TEST LE</td>
<td>V13, XF20, NSOKY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>479</td>
<td></td>
<td>TERMINATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>480</td>
<td></td>
<td>NSOKY</td>
<td>ASSIGN 45, 11, PF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>481</td>
<td></td>
<td>HELPA</td>
<td>REPORT, PFL5, PF1, PF2, PF3, PFL4, PF5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>482</td>
<td></td>
<td>TERMINATE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BLOCK NUMBER</td>
<td>LOC</td>
<td>OPERATION</td>
<td>A, B, C, D, E, F, G, H, I, J</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>-----------</td>
<td>----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>CLO</td>
<td></td>
<td>REPORT</td>
<td></td>
<td>CLOCK'S STATISTICS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPACE</td>
<td></td>
<td>BLOCK COUNTS.</td>
</tr>
<tr>
<td>BLO</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSV</td>
<td></td>
<td>SPACE</td>
<td>20, THE MEMORY AVAILABLE TO THE USERS EXPRESSED IN K D</td>
<td></td>
</tr>
<tr>
<td>FSV</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"This is the number of partitions in the main memory."
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAC</td>
<td>TITLE</td>
<td>LOAD2, STATISTICS FOR THE RELOCATABLE LOADER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td>SPACE</td>
<td>CHANNEL, STATISTICS FOR THE CHANNEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAC</td>
<td>SPACE</td>
<td>CPU, STATISTICS FOR THE CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>TITLE</td>
<td>MEMORY, STATISTICS FOR THE MAIN MEMORY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>TITLE</td>
<td>STATISTICS FOR THE QUEUE OF THE INPUT DRUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE INPUT DISK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE OUTPUT DISK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE CARDBREADER 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE CARDBREADER 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA LINE PRINTER</td>
<td>TITLE</td>
<td>STATISTICS FOR THE OUTPUT QUEUE OF THE AUXILIARY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA P 1.</td>
<td>SPACE</td>
<td>STATISTICS FOR THE OUTPUT QUEUE OF THE LINE PRINTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE COMPILER 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE COMPILER 18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE COMPILER 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE COMPILER 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>SPACE</td>
<td>STATISTICS FOR THE QUEUE OF THE COMPILER 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>TITLE</td>
<td>CHANQ, STATISTICS FOR THE QUEUE OF THE CHANNEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK NUMBER</td>
<td>LOC</td>
<td>OPERATION</td>
<td>COMMENTS</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td>TITL</td>
<td>LOADQ, STATISTICS FOR THE QUEUE OF THE RELOCATABLE LOAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUE</td>
<td></td>
<td>RAEDY, STATISTICS FOR THE QUEUE OF THE CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>1, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS A</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>2, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS B</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>3, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS C</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>4, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS D</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>5, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS E</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>6, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS F</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>7, STATISTICS FOR THE INPUT QUEUE OF JOBS THAT BELONG 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO CLASS G</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>TITL</td>
<td>WAIT1, STATISTICS FOR THE INPUT QUEUE OF NON-STANDARD J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOBS</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMS</td>
<td>TITL</td>
<td>MATRIX 1.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| TEXT         | 5    | THIS TABLE RECORDS THE Fragmentation OF THE PARTITION |
| TEXT         |      | WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS        |
| TAB INCLUDE  | 11/1 | 5, 7, 8, 9, 10, 11, 12, 13, 14, 16                  |

<p>| TEXT         | 6    | THIS TABLE RECORDS THE Fragmentation OF THE PARTITION |
| TEXT         |      | WEIGHTED BY THE TIME THAT A JOB OCCUPIES THIS        |
| TAB INCLUDE  | 12/1 | 5, 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| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE   |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | INCLUDE   | 13/1, 5, 7, 8, 9, 10, 11,|   | 12, 13, 14, 16 |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | INCLUDE   | 14/1, 5, 7, 8, 9, 10, 11,|   | 12, 13, 14, 16 |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | INCLUDE   | 15/1, 5, 7, 8, 9, 10, 11,|   | 12, 13, 14, 16 |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | INCLUDE   | 16/1, 5, 7, 8, 9, 10, 11,|   | 12, 13, 14, 16 |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | INCLUDE   | 17/1, 5, 7, 8, 9, 10, 11,|   | 12, 13, 14, 16 |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | TITLE     | 1 (IT IS NOT WEIGHTED).  |   |           |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | TITLE     | 2 (IT IS NOT WEIGHTED).  |   |           |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | TITLE     | 3 (IT IS NOT WEIGHTED).  |   |           |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |
|             | TAB  | TITLE     | 4 (IT IS NOT WEIGHTED).  |   |           |
|             | SPACE| TEXT      | THIS TABLE RECORDS THE   |   | WEIGHTED  |
|             | SPACE| TEXT      | PARTITION FOR THE        |   | BY THE    |
|             | SPACE| TEXT      | TIME THAT A JOB OCCUPIES|   | THIS      |
|             | SPACE| TEXT      | PARTITION.               |   | BLOCK.    |</p>
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>*LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I, J</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB</td>
<td>TITLE</td>
<td>12, THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON 5</td>
<td></td>
<td>IT IS NOT WEIGHTED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB</td>
<td>TITLE</td>
<td>3, THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON 6</td>
<td></td>
<td>IT IS NOT WEIGHTED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB</td>
<td>TITLE</td>
<td>4, THIS TABLE RECORDS THE FRAGMENTATION OF THE PARTITION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON 7</td>
<td></td>
<td>IT IS NOT WEIGHTED)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

GPSS statements required for one additional input station.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAR_J FUNCTION A,B (X_1, Y_1/X_2, Y_2/.../X_n, Y_n)</td>
<td>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.</td>
</tr>
<tr>
<td>K FUNCTION A,B (X_1, Y_1/X_2, Y_2/.../X_m, Y_m)</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 (an 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,FM$CARIJ,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.,

[BRI 70] Brinch Hansen, P.,
"The nucleus of a multiprogramming system",

[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

[FLO 69] Flores, I.,
Computer Organization, Prentice-Hall, Inc.,

[FOX 67] Fox, D., and Kessler, J.L.,
"Experiments in software modeling",

[GLI 67] Glinka, L.R., Brush, R.M., and Ungan, A.J.,
"Design thru simulation of a multiple-access information system",

[GOR 75] Gordon, Geoffrey,
The Application of GPSSV to Design System Simulation,

[HEL 75] Hellerman, H., and Conroy, F.T.,

[HOL 68] Holland, C.F., and Merikallio, A.R.,
"Simulation of a multiprocessing system using GPSS",
[HUE 67] Huesman, L.R., and Goldberg, R.P.,
   "Evaluating computer system through simulation",

[IBM 74] IBM Corporation,
   "General Purpose Simulation System-V", User's

[KAT 66] Katz, J.H.,
   "Simulation of a multiprocessor computer system",
   Vol. 28, pp. 127-139.

[KEC 75] Keech, G.L.,
   "Computer Organization", Course notes for

[KUR 75] Kurzban, S., Heines, T.S., and Sayers, A.P.,
   Operating Systems Principles, Petrocelli/charter,

[LAM 76] Lamie, L.E.,
   "Using GPSS to Teach Operating System Concepts",

[LEH 68] Lehman, H.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",

[MAD 74] Madnick, E.M., and Donovan, J.J.,

[SCH 74] Schriber, T.J.,

[SHA 74] Shaw, A.C.,
   The Logical Design of Operating Systems,

[TSI 74] Tsichritzis, C.D., and Berstein, A.P.,