IGPSS

A General Purpose Simulation

Language for the Interdata

by

Martha Hoflich

B. A. North Texas State University, Denton, Texas, 1970

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1977

Approved by:

Fred Maryanski - Major Professor
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>General Structure</td>
<td>2</td>
</tr>
<tr>
<td>IGPSS Entities</td>
<td>5</td>
</tr>
<tr>
<td>Dynamic Entities</td>
<td>5</td>
</tr>
<tr>
<td>Equipment Entities</td>
<td>6</td>
</tr>
<tr>
<td>Statistical Entities</td>
<td>7</td>
</tr>
<tr>
<td>Operational Entities</td>
<td>7</td>
</tr>
<tr>
<td>PROGRAM STRUCTURE</td>
<td>8</td>
</tr>
<tr>
<td>STANDARD NUMERICAL ATTRIBUTES</td>
<td>8</td>
</tr>
<tr>
<td>EQUIPMENT ORIENTED BLOCK TYPES</td>
<td>9</td>
</tr>
<tr>
<td>TRANSACTION-ORIENTED BLOCK TYPES</td>
<td>10</td>
</tr>
<tr>
<td>Advance</td>
<td>10</td>
</tr>
<tr>
<td>Generate</td>
<td>11</td>
</tr>
<tr>
<td>Terminate</td>
<td>12</td>
</tr>
<tr>
<td>Assign</td>
<td>12</td>
</tr>
<tr>
<td>Priority</td>
<td>12</td>
</tr>
<tr>
<td>FLOW MODIFICATION</td>
<td>13</td>
</tr>
<tr>
<td>Transfer</td>
<td>13</td>
</tr>
<tr>
<td>Test</td>
<td>13</td>
</tr>
<tr>
<td>COMPUTATIONAL AND REFERENCE ENTITIES</td>
<td>14</td>
</tr>
<tr>
<td>Functions</td>
<td>14</td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>15</td>
</tr>
<tr>
<td>TABLE OF CONTENTS (Continued)</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>STATISTICAL BLOCK TYPES</td>
<td>15</td>
</tr>
<tr>
<td>Queue Block</td>
<td>15</td>
</tr>
<tr>
<td>Depart</td>
<td>16</td>
</tr>
<tr>
<td>CONTROL CARDS</td>
<td>16</td>
</tr>
<tr>
<td>Start Card</td>
<td>16</td>
</tr>
<tr>
<td>End Block</td>
<td>17</td>
</tr>
<tr>
<td>PROGRAM OUTPUT</td>
<td>17</td>
</tr>
<tr>
<td>INTERNAL ORGANIZATION</td>
<td>18</td>
</tr>
<tr>
<td>Current Events Chain</td>
<td>18</td>
</tr>
<tr>
<td>Future Events Chain</td>
<td>19</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>19</td>
</tr>
<tr>
<td>INTERNAL FLOW</td>
<td>20</td>
</tr>
<tr>
<td>CODING IGPSS</td>
<td>22</td>
</tr>
<tr>
<td>RUNNING IGPSS</td>
<td>23</td>
</tr>
<tr>
<td>RESTRICTIONS</td>
<td>23</td>
</tr>
<tr>
<td>TESTING</td>
<td>24</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>25</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>27</td>
</tr>
</tbody>
</table>
APPENDIX A - IGPSS CODING SHEET

APPENDIX B - IGPSS BLOCK TYPES

APPENDIX C - STANDARD NUMERICAL ATTRIBUTES

APPENDIX D - QUANTITY LIMITATIONS ON IGPSS ENTITIES

APPENDIX E - DATA STRUCTURES

APPENDIX F - PROGRAM LISTING

APPENDIX G - SAMPLE SIMULATIONS

APPENDIX H - BLOCK TYPE FLOWCHARTS
INTRODUCTION

The purpose of this project is to develop a simulation language which is portable to different kinds of hardware systems. The language developed is called IGPSS and resembles IBM's General Purpose Simulation System. The program is written in Fortran since Fortran is available on most systems and is fairly well standardized. Although developed for use primarily on the Interdata 8/32, IGPSS is also implemented on the IBM/370.

This report will provide a description of the IGPSS internal organization and its user requirements. The user is assumed to have a knowledge of simulation.

The parser for IGPSS remains to be written, so the current input required for IGPSS is the anticipated output from the parser. The special requirements for this input are discussed in the section entitled "Coding IGPSS."
General Structure

There are three methods of building a simulation model. The event scheduling approach emphasizes a detailed description of the steps that occur when an individual event takes place. Each type of event has a distinct set of steps associated with it. The activity scanning approach emphasizes a review of all activities in a simulation to determine which can be begun or terminated each time an event occurs. The process interactive approach emphasizes the progress of an entity through a system from its arrival event to its departure event. (1)

IGPSS is a simulation language based on GPSS. It uses the process interactive approach to organizing behavior in a simulation. In IGPSS active objects called transactions act on a system of passive objects which include facilities and storages. For example, a job requiring service is a transaction, whereas a single server is a facility and a group of servers is a storage unit. The transaction passes through the system, encountering a facility or storage, depending on whether the simulation is a single or multiserver representation.

The program maintains a record of when each transaction in the system is due to move. The transactions move through the system on a first-come, first-served basis within a priority class. The program moves a transaction through the system until a blocking condition arises. When a transaction is moved as far as it can go, the program moves any other transaction due to move at the same time. If all movements are completed, the program advances the clock to the time of the next most imminent event and repeats the movement process.

A system to be simulated in IGPSS can be described as a block diagram in which the blocks represent activities and the lines joining the blocks indicate the sequence in which the events are to occur. If a choice of
activities can occur depending on a condition, more than one line leaves a block and the condition for the choice is stated at the block. The sequence of events in real time is reflected in the movement of transactions from block to block in simulated time.

Transactions are created at one or more GENERATE blocks and are removed from the simulation at TERMINATE blocks. There can be many transactions moving simultaneously through the system. Each transaction is always positioned at a block and most blocks can hold many transactions simultaneously. The movement of a transaction from one block to another occurs instantaneously at a specific time or when some change of system condition occurs. A block number is given to each block, and the movement of transactions is usually from one block to the block with the next highest location. The TRANSFER block allows a transaction to move to a block other than the next sequential block.

A block type called ADVANCE is used to represent a time delay. IGPSS computes an action time for each transaction as it enters an ADVANCE block, and the transaction remains at that block for the calculated period of simulated time before it attempts to proceed to the next block. Queues may be established to gather statistics on delays transactions may encounter in the system.

The first example in Appendix G is a simulation of jobs as they progress through an office. Jobs arrive at the office at the rate of one every 10 to 20 minutes. Normally they go to clerk A, who takes 7 to 13 minutes and then go to clerk B, who takes 3 to 7 minutes. However, if clerk B is busy at the time the job is brought to the office, the entire job is given to clerk C, who takes 10 to 30 minutes. Assuming that each clerk handles one job at a time, the simulation runs until 1000 jobs have been completed. The example shows the model run on GPSS followed by the
same model run on IGPSS.

In the GPSS program, the GENERATE block creates jobs every 10 to 20 minutes. The TEST block determines if clerk B is busy. If so, the job is sent to clerk C. The QUEUE block indicates entering into clerk A's stack of jobs to be processed. The stack is maintained in a FIFO arrangement. The SEIZE block indicates clerk A has begun processing the job; thus, the job leaves clerk A's stack of work in the DEPART block. The ADVANCE block indicated clerk A's processing time. The job leaves clerk A in the RELEASE block and enters clerk B's stack of work in the next QUEUE block. The job is processed by clerk B in the same manner as with clerk A. It then leaves the office in the TERMINATE block.

The output produced by both GPSS and IGPSS contains a section labeled 'BLOCK COUNTS'. This section indicates the total number of jobs that are currently in each block and the total number that has passed through each block. The facilities section shows how many jobs were processed by each clerk, the average processing time of each job, and the utilization for each clerk. The queues section indicates the maximum contents in each clerk's stack of jobs waiting to be processed was one. Most of the jobs had a zero waiting time, and the waiting time for those jobs that had to wait was 1.588 minutes for clerk A and 2.8 minutes for clerk C. No jobs had to wait for clerk B for processing.

The second example in Appendix G is a simulation of a grocery store checkout line. Thirty-two percent of the customers will use the express checkout line and the other sixty-eight percent will use one of the two regular checkouts. Customers arrive uniformly from two to fourteen minutes apart. Checkout time at the express counter requires 2 minutes and the regular lines require from 2 to 14 minutes. Queues are maintained to gather statistics on the waiting lines which form at each checkout.
counter. The simulation is run for 100 customers. The example shows the model run on GPSS followed by the same model run on IGPSS.

In the GPSS program, the STORAGE block defines 2 regular checkout lines. The GENERATE block creates a customer uniformly every 2 to 14 minutes. The TRANSFER block sends 32% of the customers to EXPRS, the rest to REGCK. The express customer is first entered into the line at the QUEUE block, is serviced at the counter in the SEIZE block, thus leaving the line in the DEPART block. The ADVANCE block indicates a two-minute checkout time. The customer then leaves the counter in the RELEASE block and is recorded as leaving the store at the TERMINATE block. The customer is serviced at a regular checkout counter in a similar manner. The START block indicates the simulation will halt when the 100th customer enters the terminate block.

The output of the second example is similar to that of the first example. The additional section entitled STORAGES shows the statistics on the two regular checkout counters.

**IGPSS Entities**

IGPSS is built around a set of simple entities which are divided into four classes: dynamic, equipment, statistical, and operational.

**Dynamic Entities.** The dynamic entities are called transactions. Transactions are the items being simulated, such as customers in a grocery store. They are created and destroyed during a simulation run. Associated with each transaction are ten parameters which can be assigned values by the user to represent characteristics of the transaction. For example, a transaction representing a customer in a grocery store might have a parameter indicating the number of grocers at checkout time. This
number could be used in the simulation to determine how long the checkout process would take.

Equipment Entities. The second type of entity, the equipment entity, can be acted on by transactions. Facilities and storages are equipment entities. A facility can handle only one transaction at a time and represents a potential bottleneck in the system. A facility could be used to represent a grocery store checkout counter. A record of which transaction is using each facility within the system is maintained. If a transaction attempts to use a facility that is already in use, that transaction is delayed until the facility is available. IGPSS also maintains a record of the total number of time units for which each facility was in use. At the end of the simulation, this figure is used to compute the utilization of each facility. A count is also maintained of the total number of times each facility was used and the average time duration of each use.

Storages are used to represent parallel processing equipment which can handle more than one transaction at a time. A storage entity could be used to represent a parking lot. The user defines the capacity of each storage that he uses and IGPSS maintains a record of the number of storage units that are occupied at any given point in time. If a transaction attempts to occupy more space than is available in a given storage, the transaction is delayed until sufficient space becomes available. IGPSS maintains a record of the total number of transactions that enter each storage. The average number of clock units each transaction remained in each storage is reported at the simulation's end.
Statistical Entities. Queues are the third type of entity. In any system, transactions may be delayed by the unavailability of an equipment entity. Transactions that are delayed in their attempt to obtain a facility or a storage constitute such a queue. Queues are used to gather statistics on system behavior. Each queue maintains a list of transactions delayed at one or more points in the system, and keeps a record of average number of transactions delayed and the length of these delays. A grocery store checkout line could be represented by a queue. The user must specify selected points in the model where queue statistics are desired. IGPSS will then gather all queue statistics; a count of total number of transactions that enter the queue and also a count of the transactions that were delayed in each queue, the average time the transactions were delayed in each queue, and the maximum number of transactions in the queue at any time.

Operational Entities. The operational entities are called blocks. Like the blocks of a flowchart, they provide the logic flow of a system, instructing the transactions where to go and what to do next.

Transactions move from block to block in the simulation model in a manner similar to the way the items they represent would progress in the real system. Each movement of a transaction is considered as an event that is due to occur at some particular point in time. The program maintains a record of the times at which these events are due to occur in their correct time sequence (first come, first served, except when different priorities exist). In those cases where the action called for cannot be performed at the time originally scheduled (such as attempting to seize a facility which is already in use), processing the transaction ceases. When the blocking condition changes, the transaction is again
moved into its proper processing sequence; that is, transactions with the earliest event time are serviced first.

IGPSS maintains a clock that records the instance of real time that has been reached in the simulation. It is referred to as the clock time. All times are represented as integer numbers. The unit of system represented by a unit of clock time is implied by the user. The time unit chosen should be the smallest time difference between event occurrences and this same unit must be maintained consistently throughout the model.

PROGRAM STRUCTURE

An IGPSS program consists of a set of block definition cards, entity definition cards, and control cards.

Each defined block is entered on a separate 'block definition card'. Each block is assigned a sequential number as the program is read. The block type is entered in the card followed by the subfields specifying the block arguments. The meaning associated with each subfield (A-E) is a function of the particular block type and will be discussed later.

The entity definition cards define storage capacities or functions. The basic format is the same as the block definition card. The function definition card is followed by cards specifying the X and Y coordinate values for the function.

The control cards tell IGPSS the length of the run desired and when the end of the input deck is reached.

STANDARD NUMERICAL ATTRIBUTES

Quantities such as function (FN) and clock time (C1) are generally termed standard numerical attributes (SNA). They represent properties or states of the simulation model and are provided to give the user dynamic
access to the results of the simulation. For example, whenever a reference is made to Q12, the contents of queue 12 are obtained. By making such a reference, a transfer can be made by a transaction if, for example, more than 20 transactions are currently in a certain queue. Thus, a user may simulate a system whose logic depends dynamically on any standard numerical attributes.

All standard numerical attributes consist of a one- or two-letter mnemonic followed by a number. The mnemonic indicates the type of attribute desired and the number indicates the specific attribute desired.

A list of all SNA's available in IGPSS is shown in Appendix C. Standard numerical attributes may be used as arguments of almost any block types.

EQUIPMENT ORIENTED BLOCK TYPES

As discussed earlier, a facility can handle only one transaction at a time and storages can handle multiple transactions. This section discusses the various block types used to deal with these equipment entities.

Two block types, SEIZE and RELEASE, are provided so that transactions may make use of a facility. When a transaction enters a SEIZE block it is recorded as using the facility. The facility then remains in use until the seizing transaction enters a RELEASE block. Upon entering the SEIZE block, the usage count of the facility is incremented. The facility to be seized is specified in field A of the block definition card. When the seizing transaction enters a RELEASE block with the same facility specified in field A, the facility is returned to the available state, and the in-use time for the facility is incremented by the number of clock
units that the transaction remained in possession of that facility. A transaction may seize any number of facilities.

Two block types, ENTER and LEAVE, provide for usage of storages. The STORAGE definition card is used to define the capacity of a storage. Field A specifies the maximum storage capacity and the location field specifies the storage number. Each ENTER and LEAVE block specifies a storage number in field A and the number of units to be occupied or returned in field B. When a transaction encounters an ENTER block, the record of the number of units occupying the storage is incremented by field B. If field B is blank it is assumed to be one. If there is not enough space available for a particular transaction, it will be refused entry until sufficient storage becomes available. If there is room for a subsequent transaction, that transaction will be admitted before the rejected one. When a transaction encounters a LEAVE block, the number specified in field B is returned to the available storage. A transaction need not remove the same number of units that it added; in fact, the transaction entering a LEAVE block need not have entered the storage previously. The contents of the storage cannot be allowed to be made negative, however.

TRANSACTION-ORIENTED BLOCK TYPES

IGPSS provides five block types which modify transaction attributes.

Advance

The first of these, the ADVANCE block, provides for the explicit specification of a time delay. This block requests a computation of the number of clock units which a transaction is to spend in the block. This time may have any integer value, including zero. If the computed time is
zero, the simulator continues processing the entering transaction and immediately attempts to move it to the next block in its path. To specify the delay time at an ADVANCE block, the user states the mean in field B. If the modifier is omitted, a constant time delay is produced.

Two types of ADVANCE block modifiers exist: a spread and a function. A spread modifier is used when the delay times are equiprobable within a given range. For example, a spread of 5, used with a mean of 10, produces action times in the range of $10 \pm 5$. The delay time then becomes one of eleven possible integers since clock time only assumes integer values. All values are computed with equal probability. If delay times in a system follow a more complex distribution pattern than the simple uniform distribution produced by the spread modifier, a function modifier must be used. A value for the specified function is computed and multiplied by the mean. If the result is noninteger, the decimal portion is truncated and the resulting integer is the computed delay time.

**Generate**

The second transaction oriented block type is the GENERATE block. This block is used to create transactions for entry into the system. Field A is the mean time between generations. This mean can be modified by specifying either a spread or a function modifier in field B, as in the ADVANCE block. Each transaction contains ten parameters which are set equal to zero when the transaction is created. Field C is an offset interval. If specified it is used without modification as the time interval preceding the creation of the first transaction by that block. Field D is a limit count on the number of transactions that can be created by the GENERATE block. If this field is omitted, the block will continue to create transactions indefinitely. Field E is the priority of the
transactions to be created by this block. Newly created transactions have a priority of zero unless another value is specified in this field.

**Terminate**

The TERMINATE block removes transactions from the system. It represents the completion of a path of flow, such as the customer leaving the grocery store. Field A of the TERMINATE block specifies whether or not this block contributes to the total termination count and, if so, how many units it will contribute. The termination count is specified on the START card which will be discussed later. If the field is not specified it is interpreted to be 0, and transactions entering the block do not decrement the termination count. There must be at least one TERMINATE block with a value of 1 or more in field A of every model to terminate the simulation. If this does not happen, the model runs indefinitely.

**Assign**

Each transaction has 10 parameters associated with it. Contents of parameters are signed integers ranging from $-(2^{31})$ to $(2^{31}-1)$. At the time of creation of a transaction, these parameters are initialized to zero. The ASSIGN block is the principal means of entering values into the parameter fields of a transaction. The value of field A specifies which parameter is to be modified. Field B specifies the integer value the parameter is to become.

**Priority**

The PRIORITY block is used to set the priority of a transaction to a specified value. Field A specifies the priority the transaction will assume.
FLOW MODIFICATION

In IGPSS, the flow of transactions through blocks is generally sequential. That is, it is assumed that a transaction goes from its current block to the next sequentially numbered block. However, in an actual system, the flow may be diverted.

Transfer

The TRANSFER block is generally used to direct the transaction to a nonsequential next block. This may be done in a variety of ways. The choice is specified by means of a mnemonic selection mode specified in field A of the block. If the selection mode at the TRANSFER block is blank, every transaction that enters the block will be sent to the block specified in field B. If the selection mode is a fraction, a random choice between the blocks specified in fields B and C is made. The probability of selecting the next block in field C is given by the fraction. For example, if a selection mode of .370 were specified in field A, 37% of the transactions would proceed to the next block specified in field C and 63% would proceed to the next block specified in field B.

Test

The TEST block also provides a possible modification of the sequential block flow. The TEST block specifies a condition specified in the form of an algebraic comparison between two arguments. If the desired relation is satisfied, the transaction proceeds to the next sequential block. If not satisfied, the transaction proceeds to the block specified in field C. The arguments are specified in fields A and B. The following six mnemonic relations may be specified:
L  Less than. If the first argument is greater than or equal to the second argument, the relation is not satisfied.

LE  Less than or equal to. The relation is satisfied unless the first argument is greater than the second.

E  Equal to. The relation is satisfied only if the two arguments are equal.

NE  Not equal. The relation is satisfied unless the two system variables are equal.

G  Greater than. If the first argument is less than or equal to the second argument, the relation is not satisfied.

GE  Greater than or equal to. The relation is satisfied unless the first argument is less than the second argument.

COMPUTATIONAL AND REFERENCE ENTITIES

Very often, the complex logical and mathematical interrelations which exist among items within a model may be best expressed in terms of a mathematical function.

Functions

A FUNCTION returns a numerical value that is computed by a rule defined by the user. IGPSS provides for two types of functions: continuous and discrete. A unique relationship is defined between the independent and dependent variables. The independent variable can be any of the standard numerical attributes and is specified in field A. Field B contains the number of pairs of points desired, preceded by an
indication of whether a discrete or continuous function is desired. The X and Y values are entered in cards immediately following the FUNCTION block card. These values are entered by specifying \( X_1, Y_1, X_2, Y_2, \ldots, X_n, Y_n \), until the desired number of pairs is given.

**Random Number Generation**

There are eight random number seeds in IGPSS. The user may specify any one of these seeds (RN1-RN8). In cases where reference to a random number is implied (such as TRANSFER .5, albl, blbl), RN1 is used.

**STATISTICAL BLOCK TYPES**

The IGPSS user will primarily be interested in observing the values of numerical attributes which characterize a system in order to determine how the system is performing in simulated operation. Some statistics are gathered automatically by the program, but others which might be of interest can be gathered by inserting various queues into the system.

Queues are measured by inserting QUEUE and DEPART blocks into the model. A QUEUE block may be placed before any block that might cause a delay. The transactions increment the contents of a queue when they enter a QUEUE block and decrement the contents of that queue when they succeed in entering a DEPART block.

**Queue Block**

The QUEUE block is analogous to the ENTER block and signals the program that statistics should be gathered for a queue at the specified point in the model. The number of the queue the transaction is to enter is specified in field A. The length of time for which the queue specified has been at its present content is then computed. The product of that interval and the present contents of the queue are added to a sum that
represents the total unit occupancy (cumulative time integral) of the queue. At the end of the simulation run, this sum is divided by the length of the run to form the average contents of the queue. The number of units to be added to the queue is specified at field B in the same manner as in the ENTER block and LEAVE block. If field B is left blank it is assumed to be 1. When a transaction enters the block, the number of units specified in field B is added to the current contents of the queue. IGPSS then compares the new contents with the previous maximum length. If the new contents is larger, the new queue length is recorded as the maximum. The count of the total number of units that have passed through the queue is also incremented by the amount in field B. This total is used to compute the weighted average time per transaction in the queue.

**Depart**

The DEPART block is analogous to the LEAVE block. Field A specifies the queue number and field B, the number of units to leave the queue. When a transaction enters a DEPART block, the number of units specified in field B is subtracted from the contents of the queue. IGPSS also computes the length of time which transactions remained in the queue. If the time interval is 0, the specified number of units is added to a count which records the total number of units to pass through the queue with no delay.

**CONTROL CARDS**

**Start Card**

This card indicates to the simulator that all input data has been received and that the run may now proceed. Field A specifies the number of terminations to be executed before printing the final summary statistics. This quantity is called the run termination count and when it has been
decremented to 0 or less, the run will terminate. The count is
decremented by the amount specified in field A of a TERMINATE block each
time a transaction enters that block.

End Block

This card specifies the end of the input deck.

PROGRAM OUTPUT

Program output occurs when an error condition arises or upon normal
completion of a run.

The output produced upon normal completion is as follows:

1) Clock time - This is the actual clock time at the end of the
simulation run.

2) Block counts - The entry and wait counts of all defined blocks is
printed out. This printout includes block number, the number of
transactions currently waiting at that block and the total number of
transactions which have entered that block in the course of the simulation.

3) Facility statistics - The statistics which are gathered for each
facility referred to in the simulation are listed. These statistics
include utilization (the fraction of time that each facility was seized),
the total number of entries, the average number of time units that a
transaction held each facility, and the number of the transaction seizing
the facility at the clock instance the run ended.

4) Storage statistics - A listing is provided of the statistics
gathered for each storage that was referenced during a run. These include
the capacity of each storage as defined by the user, the average contents
of each storage, the average utilization (average contents divided by
maximum capacity), the total number of entries, the average length of
time transactions remained in each storage, the current contents at the end of the run, and the maximum occupancy recorded for each storage during the run.

5) **Queue statistics** - The statistics gathered for each queue referenced in the simulation. These statistics include the maximum contents of each queue, the average contents, the total number of entries, the number of transactions which entered each queue but were not delayed, the average time per transaction, the average time per transaction for those transactions which were delayed (excluding zero delay entries), and the current contents of each queue at the time of the simulation run termination.

**INTERNAL ORGANIZATION**

Transactions in IGPSS are maintained in chains. There are two such chains and a transaction may be in either one at a given time. The *current event chain (CEC)* contains those transactions whose scheduled event time is less than or equal to the current value of the clock. The *future events chain (FEC)* contains those transactions whose scheduled event times are greater than the current clock time; that is, transactions whose event time is in the future.

**Current Events Chain**

The current events chain is organized in descending order of priority. Each transaction in the CEC is either in the active status or in a delay status. The scan status indicator specifies the status condition. If the transaction is in the active status IGPSS attempts to move it to its next block. If the transaction has been blocked for some reason it is put into delay status. Transactions in this status are
skipped by the overall scan. When the blocking situation is changed, the transaction is returned to the active status.

Future Events Chain

The future events chain transactions are strictly in ascending block departure time order. There are no distinctions made by priority in this chain.

DATA STRUCTURES

The data structures in IGPSS are implemented through the use of arrays in a common block. A maximum of 200 transactions are allowed at any one time. Each transaction word contains a pointer to the next block to be executed, a scan status indicator, the priority of the transaction, the time of the last status change, the current block, and 10 parameters which are initially zero. A maximum of 120 blocks are permitted. Each block word contains the block type, field A, B, C, D, and E if applicable, and a current and total block entry count. Thirty-five facilities can be used. Each facility word contains the number of the transaction seizing the facility, the cumulative time usage, the clock time of the last status change, an indicator as to whether a delay chain exists, and an entry count. Each storage word of the 35 possible storages contains the current contents, the available contents, the cumulative time usage, the clock time of the last status change, the entry count, the maximum contents, and a field indicating the existence of a delay chain. Each of the 70 queue words contains the clock time of the last status change, the entry count, the zero delay entries, the cumulative time usage, the current and the maximum contents. Twenty functions are allowed with a maximum of twenty-five X and Y values. Each function word also indicates
the number of X and Y values present, whether it is continuous or discrete, and the SNA associated with function. Also contained in the common block are the future and current events chains. Each chain may contain fifty transactions at any one time. Each entry in the chains contains pointers to the previous and next entry in the chain, the priority of the transaction and the transaction number. The future events chain entries also contain the block departure time. A pointer is maintained to the first and last entry in each chain. A pointer also exists which indicates the next available array location into which a new entry is to be placed when a transaction is added to a chain. The common block also contains the clock time and the status change flags. Eight random number seeds are available and are located in the common area.

INTERNAL FLOW

The main routine initializes all necessary common block data then calls the input routine. The input routine reads all the simulation input data. If a generate block type is read, the GENERATE routine is called and a transaction is entered into the future events chain. If a START block type is encountered, the start count is initialized. If a FUNCTION definition is encountered, the function initialize routine is called and the function X, Y cards are read and the function common block is initialized. If a STORAGE definition is read, the appropriate storage available area is initialized. When an END block is read, the input routine returns control to the main program. If all input is read before an END block is encountered, the program encounters an end of file error and execution is halted. The main routine then calls the SCAN routine which controls the overall transaction scanning. A check is first
made to determine if the start count is greater than zero. If not, an error is written and the program is stopped.

The overall scan is divided into three phases: clock updating, current event chain scan, and transaction move. The overall scan begins by increasing the clock time to the block departure time of the first transaction in the future events chain. Next the first transaction is moved from the future events chain to the current events chain. Succeeding future events chain transactions are moved to the current events chain as long as their block departure time equals the new clock time. Transactions are inserted into the current events chain maintaining the priority ordering. New transactions are added behind existing ones with the same priority. When all transactions have been moved, the status change flag is turned off and the current events chain scan begins. The first current events chain transaction is obtained. If the scan status indicator (SSI) is on, the transaction is not processed since it is waiting on an unavailable entity. The next current events chain transaction is then obtained. When all transactions are processed the clock time is again updated to the block departure time of the first future events chain transaction and the process is repeated. When a current events chain transaction is obtained whose SSI is not on, IGPSS attempts to move the transaction through the next block. The block number is obtained from the transaction word and the appropriate block type routine is called. If the transaction did not succeed in entering the next block, the status change flag is tested. If on, a facility or storage was released which had a transaction waiting on it. IGPSS then returns to the first of the current events chain and repeats processing of the entire chain. If the status change flag is not on, the next transaction in the CEC is obtained for processing. If the transaction did move through the current block, a test
is made to determine if the block terminated the transaction or if it entered a positive time advance block. If the test is true, IGPSS then checks the status change flag and proceeds as above. If the test is false, an attempt is made to move the transaction through the next block by calling the appropriate block routine. Thus each CEC transaction is in turn moved through as many blocks as possible. Then movement is stopped; control passes to the next CEC transaction. If a transaction frees a facility or storage which has waiting transactions, control returns to the first of the CEC after the freeing transactions movement is stopped.

The flow of the routines which process each individual block type is illustrated by the flowcharts in Appendix F.

CODING IGPSS

The parser of IGPSS remains to be written. Currently IGPSS uses as its input the anticipated output from the parser. A coding sheet to simplify data entry is provided in Appendix A. The NUM field, columns 1-3, is an integer variable which contains the storage or function number. The remaining eight fields correspond to the fields 1-8 of the block word. They are all real numbers. The block type entered in columns 4-5 is shown in Appendix B. The SNA for field A is entered in columns 6-7. A table of SNA's is shown in Appendix C. Field A is contained in columns 8-17. The SNA for field B is placed in 18-19. Fields B, C, D, and E are placed in columns 20-29, 30-39, 40-49, 50-59, respectively. If the SNA's for fields A and B are constants, the SNA field may be left blank.
RUNNING IGPSS

To run IGPSS on the Interdata 8/32 the user must sign on, load IGPSS, select the logical units desired for input and output, then start the program. Following is an example of some possible methods of running IGPSS.

To use card input and the terminal as output:

```
SIGNON yourname,5,GRAF
V USR6
LO IGPSS
AS 5,CR:
AS 6,.ME
ST
SIGNOFF
```

To use terminal as input and printer as output:

```
SIGNON yourname,5,GRAF
V USR6
LO IGPSS
AS 5,.ME
AS 6,pr:
ST
SIGNOFF
```

The terminal will return:

```
STOP
*
```

when execution is completed. If another run is desired the program must be reloaded and the units reassigned.

RESTRICTIONS

IGPSS is a subset of GPSS. The available block types in IGPSS are identical to those described in the GPSS Introductory User's Guide (3) with the following exceptions. The number of parameters set in field F of the GENERATE block in GPSS is set to 10 in IGPSS. The function modifier in field C of the ASSIGN block is not implemented. The TRANSFER block of IGPSS does not recognize the BOTH, ALL, or SIM options. Functions
are limited to 25 X,Y coordinates and the standard numerical attributes are limited to parameters, random numbers, storage contents, storage remainder, and queue length. The quantity limitations for all IGPSS entities are shown in Appendix D.

TESTING

IGPSS was first implemented on the IBM/370. A test case was first made to test correctness of the simplest overall scan using only a GENERATE, TERMINATE, START, and END blocks. All options of the GENERATE were then tested. Next a facility was added followed by simple advance block. The modifier of the ADVANCE block was then tested. Queues were added, followed by storages and functions. The transfer was then tested and the resulting simulation was compared to the same model using GPSS. The results were very similar and any variations were attributed to the difference in the random number generators. Next the TEST block, the ASSIGN, and the PRIORITY blocks were tested. When the program appeared to be working, several test cases were run to compare results between GPSS and IGPSS. All results were very similar.

In the meantime, IGPSS was compiled on the Interdata 8/32. Minor differences were discovered between compilers on the two machines. The Interdata would not allow comparison of integers and reals or the use of a real as a subscript.

When the program was successfully working on the 370, a test run was run on the Interdata. A problem existed in passing an argument to a subroutine. The implicit real statement apparently had no effect on changing the mode of the argument. When this problem was corrected, the models tested on the 370 produced identical results to those run on the Interdata.
The run times for all test models are less than a minute. There is no means of determining CPU time on the 8/32 so no direct time comparisons could be made. The compile time on the Interdata, however, was over fifteen minutes while it was less than one minute on the 370. No apparent reason was found for this long a compile time required on the Interdata.

CONCLUSION

IGPSS is currently implemented on the IBM/370 and the Interdata 8/32. Minor compiler differences were encountered between the two hardware configurations during development. Simulation results from the test models were identical on both systems.

Some IGPSS standard numerical attributes such as variable and function can contain subfields which are also standard numerical attributes. This would require a recursive call to the routine which determines the SNA. Since recursion is not possible in Fortran, the variable block type was not implemented and the standard numerical attributes available with functions are limited. The variable block type is very useful in GPSS and could possibly be added as a future project through the use of an assembler subroutine. The SNA's for use with functions could be changed fairly easily if the need for additional SNA's arises.

Input to IGPSS is somewhat clumsy, but this was anticipated since the parser remains to be written. The parser will allow the user to enter his data in a format similar to that used for GPSS. This format is more free form than that necessary for IGPSS. The parser will also accept variable names for entities and labels and resolve them to the absolute numbers required by the IGPSS program.
Most of the restrictions on internal limitations shown in Appendix D could be increased with little difficulty. The dimensions of the arrays in common would have to be increased and some minor program changes might have to be made, depending on the entity that is to be increased. The limitations set for IGPSS are the same as the 64K version of GPSS.
REFERENCES


THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE. THIS IS AS RECEIVED FROM CUSTOMER.
<table>
<thead>
<tr>
<th>Num</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 - 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 - 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 - 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 - 19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 - 29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 - 39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 - 49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 - 59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK TYPE</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - Generate</td>
<td>mean</td>
<td>(spread)</td>
<td>(offset)</td>
<td>(count)</td>
<td>(priority)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - seize</td>
<td>facility no.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - release</td>
<td>facility no.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 - storage</td>
<td>capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - enter</td>
<td>storage</td>
<td>(units)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 - leave</td>
<td>storage</td>
<td>(units)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 - advance</td>
<td>mean</td>
<td>(spread)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 - terminate</td>
<td>(units)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 - assign</td>
<td>parameter</td>
<td>source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 - priority</td>
<td>priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 - transfer</td>
<td>selection mode</td>
<td>next block a</td>
<td>next block b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 - test</td>
<td>arg1</td>
<td>arg1</td>
<td>next block b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 - function</td>
<td>ind. variable</td>
<td>no points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 - variable</td>
<td>not implemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 - queue</td>
<td>queue no.</td>
<td>(units)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 - depart</td>
<td>queue no.</td>
<td>(units)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 - start</td>
<td>units</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 - end</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STANDARD NUMERICAL ATTRIBUTES (SNA)

1 Pj  parameters
2 PR  priority
3 Kj  constant
4 RN(x)  1 x 8 random numbers
5 C1  clock time
6 FNj  function
7 Vj  variable
8 Sj  contents of storage
9 RJ  remaining storage
10 SRj  utilization of storage
11 SAj  average of storage
12 SMj  maximum of storage
13 SCj  number of entries
14 STj  average time in storage
15 Fj  status of facility
16 FRj  utilization of facility
17 FCj  number of entries in facility
18 FTj  average time in facility
19 Qj  queue length
20 QAj  average length of queue
21 QMj  maximum length of queue
22 QCj  number of entries in queue
23 QZj  number of zero entries in queue
24 QTj  average time with zero entries in queue
25 QXj  average time with no zero entries in queue
QUANTITY LIMITATIONS ON IGPSS ENTITIES

Internal Limitations

| Parameters  |   10  |
| Transactions|  200  |
| Block       |  120  |
| Facilities  |   35  |
| Storage     |   35  |
| Queues      |   70  |
| Functions   |   20  |
| Variables   | not implemented |

Currents Events Chain (CEC)

| CEC1          | previous # in chain |
| CEC2          | next # in chain     |
| CEC3          | priority            |
| CEC4          | tran #              |

Future Events Chain (FEC)

| FEC1          | previous # in chain |
| FEC2          | next # in chain     |
| FEC3          | block departure time (BDT) |
| FEC4          | priority            |
| FEC5          | tran #              |
DATA STRUCTURES

Facility word

F1 tran # seizing
F2 cumulative time usage
F3 clock time of last status change
F4 delay chain
F5 entry count

Storage word

S1 current contents
S2 available
S3 cumulative time usage
S4 last clock status change
S5 entry count
S6 maximum contents
S7 delay chain

Queue word

Q1 time of last status change
Q2 entry count
Q3 zero-delay entries
Q4 cumulative time
Q5 current contents
Q6 maximum contents

Transaction word

T1 next block for transaction
T2 scan status indicator
T3 priority
T4 time of last status change
T5 current block
T6-T15 parameters
Block word

B1       block type
B2       sna for field A
B3       field A
B4       sna for field B
B5       field B
B6       field C
B7       field D
B8       field E
B9       current block count
B10      total block count

Function word

Func1-Func25     x values
Func26-Func50     y values
Func51            number of points
Func52            continuous(C) or discrete(D)
Func53            sna for field C
Func54            field C
THE MAIN ROUTINE Initializes all necessary common block variables, calls the input routine, call the scan routine and calls the output routine routine.

C******************************************************************************
C
C
C
C******************************************************************************
C
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANC,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
1 QUEUE(70,6),FEC(50,5),FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2 STATUS,MOVE,C1,TRANO,BLCKCN,STARTC,RN(8),TRANA,FUNC(20,54)
DO 1 I=1,8
1 PN(I) = 13672563
DO 10 I = 1,35
DO 20 J = 1,5
FACIL(I,J) = 0
20 QUEUE(I,J) = 0
STOR(I,1) = 0
STOR(I,2) = -1
DO 30 J = 3,7
30 STOR(I,J) = 0
10 QUEUE(I,6) = 0
DO 35 I=36,70
DO 35 J=1,6
35 QUEUE(I,J) = 0
C1 = 0
MOVE = 0
STATUS = 0
DO 40 I=1,199
40 TRAN(I,1) = I+1
TRAN(200,1) = 0
TRANA = 1
DO 50 I = 1,49
CEC(I,2) = I + 1
50 FEC(1,2) = I + 1
CECA = 1
FECA = 1
FECC = 0
FECL = 0
CECF = 0
CEC(50,2) = 0
CECL = 0
FEC(50,2) = 0
STARTC = 0
DO 60 I = 1,120
   BLOCK(I,9) = 0
   BLOCK(I,10) = 0
60 BLOCK(I,1) = 0
CALL INPUT
CALL SCAN
CALL OUTPUT
STOP
END
SUBROUTINE ADVANC(CECT, ID)

******************************************************************************
*                                                                      *
*                        ADVANCE                                               *
*                                                                      *
* SUBROUTINE ADVANCE DETERMINES THE BDT OF THE TRANSACTION              *
* IF THE CALCULATED TIME IS GREATER THAN C1 IT, IS PLACED IN           *
* THE FEC. IF TIME IS = C1, MOVE IS SET TO ZEPO INDICATING              *
* THE TRANSACTION HAS MOVED THROUGH THE BLOCK.                          *
******************************************************************************
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER CECT
INTEGER TYPE,TRANQ,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15), BLOCK(120,10), FACIL(35,5), STOR(35,7),
   QUEUE(70,6), FEC(50,5), FECT, FECT, FEC, CFC, CECF, CEC, CECL, CECA,
   STATUS, MOVE, CL, TRANQ, BLOCKN, STARTC, RN(8), TRANA, FUNC(20,54)

C UPDATE BLOCK COUNTS
ILST = TRANQ(TRANQ,5)
TRANQ(TRANQ,5) = BLOCKN
BLOCKQ(BLOCKN,9) = BLOCKQ(BLOCKN,9) + 1
BLOCKQ(ILST,9) = BLOCKQ(ILST,9) - 1
BLOCKQ(BLOCKN,10) = BLOCKQ(BLOCKN,10) + 1
TRANQ(TRANQ,1) = TRANQ(TRANQ,1) + 1

C DETERMINE FIELD A
SNAC = BLOCKQ(BLOCKN,2)
SNAN = BLOCKQ(BLOCKN,3)
CALL SNA(SNAC,SNAN,A)
IF(BLOCKQ(BLOCKN,5).EQ.0.0) GO TO 200

C DETERMINE FIELD B
SNAC = BLOCKQ(BLOCKN,4)
SNAN = BLOCKQ(BLOCKN,5)
CALL SNA(SNAC,SNAN,B)
IF(BLOCKQ(BLOCKN,4).EQ.0.0) GO TO 100
CALL RANU(1,1Y,YFL)
IR = 2*A*B+YFL+A-B+C1+.5
GO TO 300

C FIELD B MODIFIER IS FUNCTION

100 IR = A*B+C1
GO TO 300

200 IR = A+C1
GO TO 300

300 IF(IR.EQ.C1) GO TO 700

C PLACE IN FEC IF POSITIVE ADVANCE BLOCK
IF(FECT.FEQ.0.0) GO TO 350
FECT = FEC(FECA,2)
FEC(FECA,1) = 0
FEC(FECA,2) = 0
FEC(FECA,3) = IR
FEC(FECA,4) = CEC(CECT,3)
FEC(FECA,5) = CEC(CECT,4)
FECF = FECA
FECL = FECA
FECA = FECT
GO TO 600
350 FECT = FECF
400 IF(FEC(FECT,3).GT.IR) GC TO 500
   FECT = FEC(FECT,2)
   IF(FECT.NE.0) GO TO 400
C ADD TO ENC OF FEC
   FECT = FEC(FECA,2)
   FEC(FECA,1) = FECL
   FEC(FECA,2) = 0
   FEC(FECA,3) = IR
   FEC(FECA,4) = CEC(CECT,3)
   FEC(FECA,5) = CEC(CECT,4)
   FEC(FECL,2) = FECA
   FECL = FECA
   FECA = FECT
   GO TO 600
C ADD INTO FEC
500 FECAT = FEC(FECA,2)
   FEC(FECA,2) = FECT
   FEC(FECA,1) = FEC(FECT,1)
   FEC(FECA,3) = IR
   FEC(FECA,4) = CFC(CECT,3)
   FEC(FECA,5) = CEC(CECT,4)
   IF(FECT .EQ. FECF) FECF = FECA
   OLDPRE = FEC(FECT,1)
   IF(OLDPRE.NE.0) FEC(OLDPRE,2) = FECA
   FEC(FECT,1) = FECA
   FECA = FECAT
C UNLINK FROM CEC
600 OLD1 = CEC(CECT,1)
   OLD2 = CEC(CECT,2)
   IF(OLD1.NE.0) CEC(OLD1,2) = OLD2
   IF(OLD2.NE.0) CFC(OLD2,1) = OLD1
CEC(CECT,2) = CECA
CECA = CECT
IF(CECT.EQ.CECF) CECF = OLD2
IF(CECT.EQ.CECL) CECL = OLD1
IF(CECL.NE.0) GO TO 650
CECT = 0
ID = 1
RETURN
650 ID = 1
CECT = OLD2
RETURN
700 MOVE = 1
RETURN
END

SUBROUTINE ASSIGN

C************************************************************************************************************
C
C ASSIGN

C
C SUBROUTINE PLACES THE RESULT DETERMINED FROM FIELD B IN THE
C PARAMETER SPECIFIED IN FIELD A.
C
C************************************************************************************************************

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
1 QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2 STATUS,MOVE,C1,TRANO,BLOCKN,STARTC,RN(8),TRANA,FUNC(20,54)
MOVE = 1
C DETERMINE SNA
SNAC = BLOCK(BLOCKN,4)
SNAN = BLOCK(BLOCKN,5)
CALL SNA(SNAC,SNAN,RESULT)
C MOVE VALUE TO PARAMETER
IPARM = BLOCK(BLOCKN,3) + 5
TRAN(TRANC,IPARM) = RESULT
C
UPDATE BLOCK COUNTS
ILST = TRAN(TRANO, 5)
TRAN(TRANC, 5) = BLCKN
BLOCK(BLOCKN, 9) = BLOCK(BLOCKN, 9) + 1
BLOCK(ILST, 9) = BLOCK(ILST, 9) - 1
BLOCK(BLCKN, 10) = BLOCK(BLOCKN, 10) + 1
TRAN(TRANO, 1) = TRAN(TRANO, 1) + 1
RETURN
END
SUBROUTINE DEPART

C************************************************************************************************************
C DEPART
C
C SUBROUTINE DEPART REMOVES THE NO OF UNITS SPECIFIED IN C FIELD B FROM THE QUEUE SPECIFIED IN FIELD A.
C
C************************************************************************************************************
C
IMPLICIT INTEGER(C, F, Q, S, 0)
REAL FUNC
INTEGER TYPE, TRANO, BLOCKN, BLOCKT
COMMON /A/ TRAN(200, 15), BLOCK(120, 10), FACIL(35, 5), STOR(35, 7),
1 QUEUE(70, 6), FEC(50, 5), FCEF, FECL, FECA, CEC(50, 4), CECF, CECL, CECA,
2 STATUS, MCVE, C1, TRAN, BLOCKN, STARTC, RNN(8), TRANA, FUNC(20, 54)
C
Determinate Q
SNAC = BLOCK(BLOCKN, 2)
SNAN = BLOCK(BLOCKN, 3)
CALL SNA(SNAC, SNAN, RES)
Q = RES
C ARE YOU TRYING TO REMOVE MORE UNITS FROM QUEUE THEN IN QUEUE
C
Determinate NO TO RELEASE
SNAC = BLOCK(BLOCKN, 4)
SNAN = BLOCK(BLOCKN, 5)
CALL SNA(SNAC, SNAN, RESULT)
IR = RESULT
IF(IR.EQ.0) IR = 1
IF(IR.GT.QUEUE(Q, 5)) GO TO 1000
C CALCULATE NEW CUMULATIVE TIME
C CALCULATE NEW CURRENT IN QUEUE
    QUEUE(Q,4) = QUEUE(Q,4) + (C1 - QUEUE(Q,1)) * QUEUE(Q,5)
    QUEUE(Q,5) = QUEUE(Q,5) - IR
    QUEUE(Q,1) = C1
C CALCULATE ZERO TIME ENTRIES
    N1ME = TRA(N,0,4)
    IF(C1.EQ.N1ME) QUEUE(Q,3) = QUEUE(Q,3) + IR
    MOVE = 1
C UPDATE BLOCK COUNTS
    ILST = TRA(N,0,5)
    TRA(N,0,5) = BLC
    BLC(BLCN,9) = BLC(BLCN,9) + 1
    BLC(ILST,9) = BLC(ILST,9) - 1
    BLC(BLCN,10) = BLC(BLCN,10) + 1
    TRA(N,0,1) = TRA(N,0,1) + 1
2000 RETURN
1000 WRITE(6,1001)
1001 FORMAT('OTRYING TO REMOVE MORE UNITS FROM QUEUE THEN EXIST')
STOP
END
SUBROUTINE ENTER

C**************************************************************************C

  ENTER

C**************************************************************************C

  SUBROUTINE ENTER ATTEMPTS TO OBTAIN THE NO OF STORAGE UNITS
  SPECIFIED IN FIELD B FROM THE STORAGE SPECIFIED IN FIELD A.
  IF THE UNITS ARE OBTAINED, THE TRANSACTION PROCEEDS THROUGH
  THE BLOCK, OTHERWISE THE TRANSACTION IS BLOCKED.

C**************************************************************************C

IMPLICIT INTEGER(C,F,Q,S,0)
REAL FUNC
INTEGER TYPE,TRNO,BLCN,BLC
COMMON /A/ TRA(200,15),BLC(120,10),FACIL(35,5),STOR(35,7),
  IQEUE(70,6),FEC(50,5),FECF,FECL,FECA,CFC(50,4),CECF,CECL,CECA,
  ZSTATUS,MOVE,C1,TRAN,BLCN,STARVC,PRN(8),TRANA,FUNC(20,54)
C Determine Storage
10 SNAC = BLOCK(BLOCKN,2)
SNAK = BLOCK(BLOCKN,3)
CALL SNA(SNAC,SNAK,RES)
SNO = RES
C Has Storage Been Defined
IF(STCR(SNO,2).NE.-1) GO TO 100
WRITE(6,50) SNO
50 FORMAT('Storage No.',I3,', has not been Defined')
STOP
100 SNAC = BLOCK(BLOCKN,4)
C Determine No to Enter
C Can You Get Desired No of Storage Units
SNAK = BLOCK(BLOCKN,5)
CALL SNA(SNAC,SNAK,RESULT)
IR = RESULT
IF(IR.EQ.0) IR = 1
IF(STCR(SNO,2).GE.IR) GO TO 1000
C Set Up Delay Chain
STOR(SNO,7) = 1
GO TO 2000
C Calculate New Cumulative Time
1000 STOR(SNO,3) = STOR(SNO,3) + (C1 - STOR(SNO,4)) * STOR(SNO,1)
STOR(SNO,4) = C1
STOR(SNC,2) = STOR(SNO,2) - IR
STOR(SNC,1) = STOR(SNO,1) + IR
STOR(SNO,5) = STOR(SNO,5) + IR
IF(STCR(SNO,1).GT.STCR(SNO,6)) STOR(SNO,6) = STOR(SNO,1)
C Xact was Moved Thru Block
C Update Block Counts
ILST = TRAN(TRANO,5)
TRAN(TRANO,5) = BLOCKN
BLOCK(BLOCKN,9) = BLOCK(BLOCKN,9) + 1
BLOCK(ILST,9) = BLOCK(ILST,9) - 1
BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
TRAN(TRANO,2) = 1
MOVE = 1
TRAN(TRANC,1) = TRAN(TRANO,1) + 1

RETURN
END

SUBROUTINE FUNCT(RESULT, IN)
C*****************************************************************************
C
C
SUBROUTINE FUNCT DETERMINES THE VALUE OF THE FUNCTION
SPECIFIED AS AN SNA.

C*****************************************************************************

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANC,BLOCKA,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
1 QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2 STATUS,MOVE;CI,TRANO,BLOCKA,STARTC,RN(8),TRANA,FUNC(20,54)

C DETERMINE FUNCTION X VALUE
SNAC = FUNC(IN,53)
SNAN = FUNC(IN,54)
C CALL SNA(SNAC,SNAN,A)
C PARAMETER
IF(SNAC.EQ.1) A = TRAN(TRANO,SNAN+5)
RANDOM NO.
IF(SNAC.EQ.4) CALL RANDU(SNAN,IY,A)
C STORAGE CONTENTS
IF(SNAC.EQ.8) A = STCR(SNAN,1)
C STOPAGE REMAINDER
IF(SNAC.EQ.9) A = STOR(SNAN,2)
C QUEUE LENGTH
IF(SNAC.EQ.19) A = QUEUE(SNAN,5)
IF(FUNC(IN,52).EQ.1.0) GO TO 1000
C DISCRETE FUNCTION
IF(A.CT.FUNC(IN,1)) GO TO 100
RESULT = FUNC(IN,26)
RETURN
100  N = FUNC(IN,51)
      DO 200 I=2,N
      IF(A.GT.FUNC(IN,I)) GO TO 200
      RESULT = FUNC(IN,I+25)
      RETURN
200  CONTINUE
      RESULT = FUNC(IN,N+25)
      RETURN
C  CONTINUOUS FUNCTION
100  IF(A.GT.FUNC(IN,1)) GO TO 1100
      RESULT = FUNC(IN,26)
      RETURN
1100  N = FUNC(IN,51)
      DO 1200 I=2,N
      SLOPE = (FUNC(IN,I-25) - FUNC(IN,I+24))/
               (FUNC(IN,I) - FUNC(IN,I-1))
      U = (A.GT.FUNC(IN,I)) GO TO 1200
      IF(A.GT.FUNC(IN,I)) GO TO 1200
      RESULT = SLOPE * (A-FUNC(IN,I-1)) - FUNC(IN,I-24)
      RETURN
1200  CONTINUE
      RESULT = FUNC(IN,N+25)
      RETURN
END

SUBROUTINE FUNCTI(NUM,IN)
C******************************************************************************
C  FUNCTI
C  SUEROUTINE FUNCTI READS AND STORES THE X, Y VALUES FOR THE SPECIFIED FUNCTION.
C******************************************************************************
C  IMPLICIT INTEGER(C,F,O,S,D)
  REAL FUNC
  INTEGER TYPE,TRANO,BLOCKN,BLOKCT
  COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
               IOFUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
               2STATUS,MOVE,C1,TRAN,BLOCKN,STARTC,RN(9),TRANA,FUNC(20,54)
IXY = BLOCK(IN, 5)
FUNC(NUM, 51) = BLOCK(IN, 5)
FUNC(NUM, 52) = BLOCK(IN, 4)
FUNC(NUM, 53) = BLOCK(IN, 2)
FUNC(NUM, 54) = BLOCK(IN, 3)
READ(5, 100) (FUNC(NUM, I), FUNC(NUM, I+25), I=1, IXY)
100 FORMAT(8F10.5)
RETURN
END

SUBROUTINE GENERA(ID, IN)
C******************************************************************************
C
C
C
C SUBROUTINE GENERA CREATES A TRANSACTION, DETERMINES A
C BDT, AND ENTERS THE TRANSACTION INTO THE FEC.

C******************************************************************************
C
IMPLICIT INTEGER(C, F, Q, S, O)
REAL FUNC
INTEGER TYPE, TRAN, BLCKNA, BLCKTB
COMMON /A/ TRA(N200, 15), BLOCK(120, 10), FACIL(35, 5), STOR(35, 7),
IQUEUE(70, 6), FEC(50, 5), FECF, FECL, FECA, CEC(50, 4), CECF, CECL, CECA,
2STATUS, MOVE, C1, TRANO, BLCKKN, STARTC, RN(8), TRANA, FUNC(20, 54)
MOVE = 1
IF(BLCKIN(IN, 7).LE.0.0.AND.ID.EQ.2) GO TO 1000
C CREATE TRANSACTION
C UPDATE BLCK COUNTS
BLCKIN(IN, 9) = BLCKIN(IN, 9) + 1
BLCKIN(IN, 10) = BLCKIN(IN, 10) + 1
OLDT = TRAN
TRANO = TRANA
TRANA = TRATRAN(TRANO, 1)
TRANSI(TRAN, 1) = IN
TRANSI(TRAN, 2) = 1
IF(BLCKIN(IN, 8).EQ.0.0) BLOCK(IN, 8) = 1
TRANSI(TRAN, 3) = BLCKIN(IN, 8)
DO 100 I=6, 15

200 IR  = A * B + C1
       GO TO 400
300 IR  = A + C1
        IF(BLOCK(IN,6).EQ.0.0) GO TO 450
        IR  = IR  + BLOCK(IN,6)
        BLOCK(IN,6) = 0
C       PLACE IN FEC
450 IF(FECF.NE.0) GO TO 500
        FECTA = FEC(FECA,2)
        FEC(FECA,1) = 0
        FEC(FECA,2) = 0
        FEC(FECA,3) = IR
        FEC(FECA,4) = TRAN(TRANO,3)
        FEC(FECA,5) = TRAN
        FECF = FECA
        FFCL = FECA
        FECA = FECTA
       GO TO 800
500 FECT = FECF
600 IF(FEC(FECT,3).GT.IR) GO TO 700
        FECT = FEC(FECT,2)
        IF(FECT.NE.0) GO TO 600
100 TRAN(TRANC,1) = 0
C       DETERMINE FIELD A
        SNAC = BLOCK(IN,2)
        SNAN = BLOCK(IN,3)
        CALL SNA(SNAC,SNAN,A)
        IF(BLOCK(IN,5).EQ.0.0) GO TO 300
C       DETERMINE FIELD B
        SNAC = BLOCK(IN,4)
        SNAN = BLOCK(IN,5)
        CALL SNA(SNAC,SNAN,B)
        IF(BLOCK(IN,4).EQ.0.0) GO TO 200
        CALL RANDU(1,1Y,YFL)
        IR  = 2*B*YFL+A-B+C1+.5
       GO TO 400
C       FIELD B MODIFIER IS FUNCTION
C ADD TO ENC OF FEC
  FECT = FEC(FECA,2)
  FEC(FECA,1) = FECL
  FEC(FECA,2) = 0
  FFC(FECA,3) = IR
  FEC(FFCA,4) = TRA(TRANO,3)
  FFC(FECA,5) = TRANO
  FEC(FECL,2) = FECA
  FECL = FECA
  FECA = FECT
GO TO 800
C ADD INTO FEC
  700 FECAT = FEC(FECA,2)
       FEC(FECA,1) = FEC(FECT,1)
       FFC(FECA,2) = FECT
       FEC(FECA,3) = IR
       FEC(FECA,4) = TRA(TRANO,3)
       FEC(FECA,5) = TRANO
       OLDPRE = FEC(FECT,1)
       IF(OLDPRE.NE.0) FEC(CLDPRE,2) = FECA
       FEC(FECT,1) = FECA
       IF(FEC(T .EQ.FECF) FECF = FECA
       FECA = FECAT
  800 IF(ID.EQ.2) GO TO 900
       IF(BLOCK(IN,7).EQ.0.0) BLOCK(IN,7) = 999999999
RETURN
  900 BLOCK(IN,7) = BLCK(IN,7) -1
       TRANO = OLDT
  1000 TRA(TRANO,1) = TRA(TRANO,1) + 1
       TRA(TRANO,5) = IN
RETURN
END
SUBROUTINE INPUT
C**********************************************************************
INPUT
SUBROUTINE INPUT READ IN THE IGPSS INPUT.
C**********************************************************************
C**********************************************************************
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRANO(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
1 QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2 STATUS,MOVE,C1,TRANO,BLCCKN,STARTC,RN(8),TRANA,FUNC(20,54)
CREAD IN ALL GPSS STATEMENTS
WRITE(6,50)
50 FORMAT(*1',9X,'BLOCK TYPE A',19X,'B',19X,
1 'C', 10X,'D',10X,*E*)
I = 1
10 READ(5,100) (NUM,(BLOCK(I,J),J=1,8))
100 FORMAT(13,F2.0,F2.0,F10.0,F2.0,4F10.0)
WRITE(6,150) (NUM,(BLCCKN(I,J),J=1,8))
150 FORMAT(*',15,8F10.3)
TYPE = BLOCK(I,1)
IF(TYPE.EQ.18) RETURN
IF(TYPE.EQ.1) CALL GENERA(I,1)
IF(TYPE.EQ.17) STARTC = BLOCK(I,3)
IF(TYPE.EQ.13) CALL FUNCTI(NUM,I)
IF(TYPE.EQ.4) STOR(NUM,2) = BLOCK(I,3)
IF(TYPE.EQ.17.OR.TYPE.EQ.13.OR.TYPE.EQ.4) GO TO 10
I = I + 1
GO TO 10
END
SUBROUTINE LEAVE

C*****************************************************************************
C
LEAVE

C

SUBROUTINE LEAVE RELEASES THE NO OF UNITS SPECIFIED IN FIELD B FROM THE
STORAGE SPECIFIED IN FIELD A.

C*****************************************************************************
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANC,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2STATUS,MOVE,C1,TRANO,BLCCKN,STARTC,RN(8),TRANA,FUNC(20,54)

C DETERMINE STORAGE
SNAC = BLCK(BLOCKN,2)
SNAN = BLOCK(BLOCKN,3)
CALL SNA(SNAC,SNAN,RES)
SNO = RES

C ARE YOU TRYING TO RELEASE MORE STORAGE THAN IN USE
C DETERMINE NO TO RELEASE
SNAC = BLCK(BLOCKN,4)
SNAN = BLCK(BLOCKN,5)
CALL SNA(SNAC,SNAN,RESULT)
IR = RESULT
IF(IR.EQ.0) IR = 1
IF(IR.GT.STOR(SNO,1)) GC TO 1000

C CALCULATE NEW CUMULATIVE TIME
STOR(SNO,3) = STOR(SNO,3) + (C1 - STOR(SNO,4)) * STOR(SNO,1)

C DETERMINE NEW CURRENT AND REMAINING STORAGE UNITS
STOR(SNC,2) = STOR(SNO,2) + IR
STOR(SNO,1) = STOR(SNO,1) - IR
C DID A DELAY CHAIN EXIST
IF (STOR(SNO,7).EQ.0) GO TO 2000
STOR(SNO,7) = 0
C SET STATUS CHANGE FLAG
STATUS = 1
GO TO 2000
1000 WRITE(6,1001)
1001 FORMAT('TRANSACTION TRYING TO RELEASE MORE STORAGE THEN IN USE')
STOP
2000 MOVE = 1
C UPDATE BLOCK COUNTS
ILST = TRAN(TRANC,5)
TRAN(TRANO,5) = BLOCKN
BLOCK(BLOCKN,9) = BLOCK(BLOCKN,9) + 1
BLOCK(ILST,9) = BLOCK(ILST,9) - 1
BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
TRAN(TRANC,1) = TRAN(TRANO,1) + 1
RETURN
END
SUBROUTINE OUTPUT
C*******************************************************************************
C
C*******************************************************************************
C OUTPUT
C*******************************************************************************
C*******************************************************************************
C SUBROUTINE OUTPUT PRINTS THE SIMULATION STATISTICS.
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLACKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
 QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
 2STATUS,MOVE,C1,TRANO,BLOCKN,STARTC,RN(8),TRANA,FUNC(20,54)
C OUTPUT BLOCK COUNTS
XC1 = C1
WRITE(6,10) C1
10 FORMAT('1CLOCK',I10)
WRITE(6,20)
20 FORMAT(' BLOCK COUNTS')
25 WRITE(6,30)
30 FORMAT(' BLOCK CURRENT TOTAL ')
DO 50 I = 1,120
   IF(BLOCK(I,1),EQ.,0.) GO TO 155
   I2 = BLOCK(I,9)
   I3 = BLOCK(I,10)
   WRITE(6,70) I,I2,I3
70 FORMAT(' ',I5,I7,I8)
50 CONTINUE
155 DO 156 I=1,35
   IF(FACIL(I,5),NE.,0.) GO TO 160
156 CONTINUE
   GO TO 205
160 WRITE(6,170)
170 FORMAT('OFACILITY AVERAGE',10X,'NUMBER',7X,'AVERAGE',7X,
   1'SEIZING')
   WRITE(6,180)
180 FORMAT(' ',12X,'UTILIZATION',6X,'ENTRIES',6X,'TIME/TRAN',5X,
   1'TRAN NO.')
   DO 200 I = 1,35
      IF(FACIL(I,5),EQ.,0.) GO TO 200
      XF2 = FACIL(I,2)
      XF5 = FACIL(I,5)
      UTIL = XF2/XC1
      AVG = XF2/XF5
      WRITE(6,190) I,UTIL,FACIL(I,5),AVG,FACIL(I,1)
190 FORMAT(I5,F12.3,I16,F16.3,I12)
200 CONTINUE
C STORAGE OUTPUT
205 CONTINUE
DO 206 I=1,35
   IF(STOR(I,5).NE.0) GO TO 207
206  CONTINUE
   GO TO 255
207  WRITE(6,210)
210  FORMAT(0,23X,'AVERAGE AVERAGE',19X,'AVERAGE',
          16X,'CURRENT MAXIMUM'
   WRITE(6,220)
220  FORMAT(* STORAGE CAPACITY CONTENTS UTILIZATION',
          14X,'ENTRIES TIME/TRAN CONTENTS CONTENTS')
   DO 240 I=1,35
       IF(STOR(I,5).EQ.0) GO TO 240
      XS3 = STOR(I,3)
      XS5 = STOR(I,5)
      XCAP = STOR(I,1) + STOR(I,2)
      CAP = XCAP
      AVGCCN = XS3/XC1
      UTIL = XS3/(XC1*XCAP)
      AVGMT = XS3/XS5
   WRITE(6,230) I,CAP,AVGCCN,UTIL,STOR(I,5),AVGMT,STOR(I,1),STOR(I,6)
240  CONTINUE
C QUEUE OUTPUT
255  CONTINUE
   DO 256 I=1,70
       IF(QUEUE(I,2).NE.0) GO TO 257
256  CONTINUE
   GO TO 300
257  WRITE(6,250)
250  FORMAT(0,9X,'MAXIMUM',5X,'AVERAGE',4X,'TOTAL',8X,'ZERO',7X,
          1*PERCENT',5X,'AVERAGE',6X,'$ AVERAGE CURRENT')
   WRITE(6,260)
260  FORMAT(* QUEUE CONTENTS ENTRIES ENTRIES ENTRIES',5X,
          1 ' ZEROS TIME/TRAN TIME/TRAN',5X,'CONTENTS')
   DO 280 I = 1,70
       IF(QUEUE(I,2).EQ.0) GO TO 280
XQ3 = QUEUE(I,3)
XQ2=QUEUE(I,2)
XQ4 = QUEUE(I,4)
AVGCON = XQ4/XC1
AVGTM = XQ4/XQ2
AVGZTM = 0.0
IF(XQ2.NE.XQ3) AVGZTM = XQ4/(XQ2-XQ3)
PCTZ = XQ3/XQ2*100.
WRITE(6,270) I,QUEUE(I,6),AVGCON,QUEUE(I,2),QUEUE(I,3),PCTZ,
AVGTM,AVGZTM,QUEUE(I,5)
280 CONTINUE
WRITE(6,290)
290 FORMAT('0% AVERAGE TIME/TRAN EXCLUDES ZERC DELAY ENTRIES')
300 RETURN
END
SUBROUTINE PRIORI(CECP)

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

PRICRI

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

SUBROUTINE PRIORI DETERMINES IF THE PRIORITY IN FIELD A IS
DIFFERENT FROM THAT OF THE TRANSACTION. IF SO, THE TRANSACTION
IS DELETED FROM THE CEC AND READDED AT THE PROPER LOCATION.

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

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCT
INTEGER CECP
COMMON /A/ TRA N(200,15), BLOCK(120,10), FACIL(35,5), STOR(35,7),
 QUEUE(70,6), FEC(50,5), FECF, FECL, FECA, CEC(50,4), CECF, CECL, CECA,
  ZSTATUS, MOVE, C1, TRANO, BLCKDN, STARTC, RN(8), TRANA, FUNC(20,54)
MOVE = 1
C UPDATE BLCK COUNTS
ILST = TRAN(TRANO, 5)
TRAN(TRANC, 5) = BLCKN
BLCK(BLOCKN, 9) = BLCK(BLOCKN, 9) + 1
BLCK(ILST, 9) = BLCK(ILST, 9) - 1
BLCK(BLOCKN, 10) = BLCK(BLOCKN, 10) + 1
TRAN(TRANC, 1) = TRAN(TRANO, 1) + 1
RESULT = BLCK(BLOCKN, 3)
IF(RESULT .EQ. TRAN(TRANC, 3)) RETURN
CEC(CECP, 3) = RESULT
TRAN(TRANC, 3) = RESULT
C MOVE TRAN LAST IN NEW PRIORITY CLASS IN CEC
CECT = CECF
IR = RESULT
IF(CECF .EQ. CECL) RETURN
C UNLINK FROM OLD LOCATION
OLD1 = CEC(CECP, 1)
OLD2 = CEC(CECP, 2)
IF(OLD1 .EQ. 0) CECF = CEC(CECP, 2)
IF(OLD2 .EQ. 0) CECL = CEC(CECP, 1)
IF(OLD1 .NE. 0) CEC(OLD1, 2) = OLD2
IF(OLD2 .NE. 0) CEC(OLD2, 1) = OLD1
CECT = CECF
50 IF(CEC(CECT, 3) .GE. IR) GC TO 100
OLD1 = CFC(CECT, 1)
CEC(CECP, 1) = CEC(CECT, 1)
CEC(CECP, 2) = CECT
CEC(CECT, 1) = CECP
IF(OLD1 .NE. 0) CEC(OLD1, 2) = CECP
IF(OLD1 .EQ. 0) CECF = CECP
STATUS = 1
RETURN
100 IF(CEC(CECT, 2) .EQ. 0) GO TO 150
CECT = CEC(CECT, 2)
GO TO 50
150 OLD2 = CEC(CECT, 2)
CEC(CECP,1) = CECT
CEC(CECP,2) = CEC(CECT,2)
CEC(CECT,2) = CECP
IF(OLC2.NE.0) CEC(CECT,1) = CECP
IF(OLD2.EQ.0) CECL = CECP
C
SET STATUS CHANGE FLAG ON
STATUS = 1
RETURN
END
SUBROUTINE QUEUES

**SUBROUTINE QUEUES ENTERS THE NO OF UNITS SPECIFIED IN FIELD B INTO THE QUEUE SPECIFIED IN FIELD A.**

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15), BLOCK(120,10), FACIL35,5, STOR35,7,
QUEUE70,6, FEC50,5, FECF, FEC, FECA, CEC50,5, CECECL, CECA,
2STATUS, MOVE, C1, TRANO, BLOCKN, STARTC, RN8, TRAN, FUNC(20,54)
C
Determine Q
SNAC = BLOCK(BLOCKN,2)
SNAN = BLOCK(BLOCKN,3)
CALL SNA(SNAC, SNAN, RES)
Q = RES
C
CALCULATE NEW CUMULATIVE TIME
QUEUE(Q,4) = QUEUE(Q,4) + (C1 - QUEUE(Q,1)) * QUEUE(Q,5)
C
Determine NO TO QUEUE
SNAC = BLOCK(BLOCKN,4)
SNAN = BLOCK(BLOCKN,5)
CALL SNA(SNAC, SNAN, RESULT)
IF(RESULT.EQ.0.0) RESULT = 1.0
QUEUE(Q,5) = QUEUE(Q,5) + RESULT
TRAN(TRANO,4) = C1
C INCREMENT ENTRY COUNT
   IF(Queue(Q,5).GT.Queue(Q,6)) Queue(Q,6) = Queue(Q,5)
C IS NEW CURRENT > OLD MAX
   Queue(Q,2) = Queue(Q,2) + 1
   Queue(Q,1) = C1
   Move = 1
   Tran(TranO,1) = Tran(TranO,1) + 1
C UPDATE BLOCK COUNTS
   Ilist = Tran(TranO,5)
   Tran(TranO,5) = Block
   Block(BlockN,9) = Block(BlockN,9) + 1
   Block(Ilist,9) = Block(Ilist,9) - 1
   Block(BlockN,10) = Block(BlockN,10) + 1
RETURN
END

SUBROUTINE RANDU(IN, IY, YFL)
C******************************************************************************
C
C
C******************************************************************************
C
IMPLICIT INTEGER(C, F, Q, S, O)
   REAL Func
   INTEGER Type, Tran, BlockN, BlockT
   COMMON /A/ Tran(200,15), Block(120,10), Facil(35,5), Stor(35,7),
   1 Queue(70,6), Tec(50,51), TecF, TecL, TecA, TecC(50,4), TecF, TecL, TecA,
   2 Status, Move, C1, TranC, BlockN, StartC, RN(9), TranA, Func(20,54)
   IX = PN(IN)
   IY = IX * 1220703125
   IF(IY) 1,2,2
1  IY = IY + 2147483647 + 1
2  YFL = IY
   YFL = YFL * 0.4656613E-9
   PN(IN) = IY
RETURN
END

SUBROUTINE RELEAS

C********************************************************************
C
C
C SUBROUTINE RELEAS RELEASES THE FACILITY SPECIFIED IN FIELD A.
C
C********************************************************************

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
STATUS,MOVE,C1,TRANO,BLOCKN,STARTC,RN(8),TRANA,FUNC(20,54)

C DETERMINE FACILITY
SNAC = BLOCK(BLOCKN,2)
SNAN = BLOCK(BLOCKN,3)
CALL SNA(SNAC,SNAN,RES)
FAC = RES

C TRAN# SAME AS SEIZING TRAN
IF(FACIL(FAC,1).NE.TRAN)GO TO 1000

C CALCULATE NEW CUMULATIVE TIME
FACIL(FAC,2) = FACIL(FAC,2) + C1 - FACIL(FAC,3)

C SET FACILITY NOT IN USE
FACIL(FAC,1) = 0
FACIL(FAC,3) = C1
IF(FACIL(FAC,4).EQ.0) GC TO 100

C ELIMINATE DELAY CHAIN
FACIL(FAC,4) = 0

C SET STATUS CHANGE FLAG IF OTHER TRANS WAITING TO SEIZE FACILITY
STATUS = 1

100 MOVE = 1

C UPDATE BLOCK COUNTS
ILST = TRAN(TRANO,5)
TRAN(TRANO,5) = BLCKN
BLOCK(BLOCKN,9) = BLOCK(BLOCKN,9) + 1
BLOCK(ILST,9) = BLOCK(ILST,9) - 1
BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
TRAN(TRANO,1) = TRAN(TRANO,1) + 1
RETURN

1000 WRITE(6,1001)
1001 FORMAT('OFACILITY RELEASED BY A TRANSACTION NOT SEIZING IT')
STOP
END

SUBROUTINE SCAN

C***********************************************************************
C
SCAN
C
SUBROUTINE SCAN CONTROLS THE OVERALL IGPS1 SCAN. IF MOVE
EQUALS 1 THE TRANSACTION ENTERED THE NEXT BLOCK. IF STATUS
EQUALS 1, A STATUS CHANGE OCCURRED AND THE OVERALL SCAN
RETURNS TO THE FIRST OF THE CEC.
C
C***********************************************************************

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
QUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2STATUS,MOVE,C1,TRANO,BLCKN,STARTC,RN(8),TRANA,FUNC(20,54)
CD = 0
IF(STARTC.NE.0) GO TO 10
WRITE(6,101)
101 FORMAT('ONO START CARD, EXECUTION TERMINATED')
STOP
C
INCREASE CLOCK TO BDT OF FIRST TRAN IN FEC
10 C1 = FEC(FECF,3)
CECT = CECF
C IS CURRENT EVENT CHAIN NULL
IF(CECT.NE.0) GO TO 100
C SET UP CEC LINKAGE
CECF = CEC
CECL = CEC
OLDNXT = CEC(CECA,2)
CEC(CECF,1) = 0
CEC(CECF,2) = 0
GO TO 200
50 IF(CEC(CECT,2).EQ.0) GO TO 75
CECT = CEC(CECT,2)
GO TO 100
C WAS TRAN ADDED AT END OF CHAIN
75 OLDNXT = CEC(CECA,2)
OLD2 = CEC(CECT,2)
CEC(CECA,1) = CECT
CEC(CECA,2) = CEC(CECT,2)
CEC(CECT,2) = CECA
IF(OLD2.NE.0) CEC(CECT,1) = CECA
IF(OLD2.EQ.0) CECL =CECA
GO TO 200
C INSERT TRAN IN PROPER PRIORITY IN CEC
100 IF(CEC(CECT,3).GE.FEC(FECF,4)) GO TO 50
OLDNXT = CEC(CECA,2)
C SET UP LINKAGE OF NEW TRAN
OLD1 = CEC(CECT,1)
CEC(CECA,1) = CEC(CECT,1)
CEC(CECA,2) = CECT
CEC(CECT,1) = CECA
IF(OLD1.NE.0) CEC(OLD1,2) = CECA
IF(OLD1.EQ.0) CECF = CECA
C MOVE LP AVAILABLE PTR
200 CEC(CECA,3) = FEC(FECF,4)
CEC(CECA,4) = FEC(FECF,5)
CECA = OLDNXT
C UNLINK FROM FEC
FECT = FEC(FECF,2)
FEC(FECF,2) = FECA
FECA = FECE
FECE = FECT
IF(FECT .NE. 0) FEC(FECT, 1) = 0
IF(FECT .EQ. 0) FECL = 0
CECT = CECF

C IS BDT OF NEXT TRAN EQUAL CLOCK
IF(FECT .EQ. 0) GO TO 300
IF(FEC(FECT, 3) .EQ. C1) GO TO 100
C SET STATUS CHANGE OFF

300 STATUS = 0
C GET FIRST CEC TRAN
CECT = CECF

310 TRANO = CEC(CECT, 4)
C SSI ON
C IF(TRAN(TRANO, 2) .EQ. 1) GO TO 400
GO TO 400

320 CECT = CFC(CECT, 2)
C LAST CEC TRAN
IF(CECT .EQ. 0) GO TO 10
GO TO 310
C MOVE TRAN - DETERMINE BLOCK TYPE

400 BLOCKN = TRAN(TRAND, 1)
BLOCKT = BLOCK(BLOCKN, 1)
IF(BLOCKT .EQ. 1) CALL GENERA(2, BLOCKN)
IF(BLOCKT .EQ. 2) CALL SEIZE
IF(BLOCKT .EQ. 3) CALL RELEASE
IF(BLOCKT .EQ. 5) CALL ENTER
IF(BLOCKT .EQ. 6) CALL LEAVE
IF(BLOCKT .EQ. 7) CALL ADVANCE(CECT, CD)
IF(BLOCKT .EQ. 8) CALL TERMIN(CECT, CD)
IF(BLOCKT .EQ. 9) CALL ASSIGN
IF(BLOCKT .EQ. 10) CALL PRIORI(CECT)
IF(BLOCKT .EQ. 11) CALL TRANSF
IF(BLOCKT .EQ. 12) CALL TEST
IF(BLOCKT .EQ. 15) CALL QUEUES
IF(BLOCKT .EQ. 16) CALL DEPART
C DID TRAN ENTER NEXT BLOCK
IF(MOVE.NE.1) GO TO 500
MOVE = 0
C TERMINATE BLOCK
IF(BLCKT.NE.8) GO TO 450
IF(STARTC.LE.0) RETURN
GO TO 500
C ADVANCE BLOCK
450 IF(BLCKT.NE.7) GO TO 400
C IS STATUS CHANGE CN
500 IF(STATUS.EQ.1) GO TO 300
C IS END OF CEC REACHED
IF(CD.NE.1) CECT = CEC(CECT,2)
IF(CECT.EQ.0) GO TO 10
CD = 0
GO TO 310
END
SUBROUTINE SEIZE

C******************************************************************************
*
* SEIZE
*
*
* SUBROUTINE SEIZE ATTEMPTS TO SEIZE THE FACILITY SPECIFIED
* IN FIELD A. IF THE FACILITY IS CURRENTLY IN USE THE
* TRANSACTION IS BLOCKED.
*
******************************************************************************
*
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANC,BLCKN,BLCKT
COMMON /A/ TRAN(200,15),BLCK(120,10),FACIL(35,5),STOR(35,7),
QUEUE(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
2STATUS,MOVE,C1,TRANO,BLCKKN,STARTC,RN(8),TRANA,FUNC(20,54)
C DETERMINE FACILITY
10 SNAC = BLCKN(BLCKA,2)
SNAN = BLCKN(BLCKN,3)
CALL SAA(SNAC,SNAN,RES)
FAC = RES
C     IS FACILITY IN USE
      IF(FACIL(FAC,1).EQ.0) GC TO 1000
C SET UP DELAY CHAIN
   FACIL(FAC,4) = 1
   GO TO 2000
C NOT IN USE
C PLACE TRAN # IN F1
1000  FACIL(FAC,1) = TRANO
      TRAN(TRANO,2) = 1
C PLACE CLOCK TIME IN F3
   FACIL(FAC,3) = C1
C INCREMENT ENTRY COUNT
   FACIL(FAC,5) = FACIL(FAC,5) + 1
   TRAN(TRANO,1) = TRAN(TRANO,1) + 1
   MOVE = 1
C UPDATE BLOCK COUNTS
   ILST = TRAN(TRANO,5)
   TRAN(TRANO,5) = BLOCKN
   BLOCK(BLOCKN,9) = BLOCK(BLOCKN,9) + 1
   BLOCK(ILST,9) = BLOCK(ILST,9) - 1
   BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
   TRAN(TRANO,2) = 1
2000  RETURN
END

SUBROUTINE SNA(SNAC,SNAN,RESULT)
C******************************************************************************
C
C          SNA
C
C          SUBROUTINE DETERMINES THE VALUE OF THE SNA.
C
C******************************************************************************
C
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER SNAC,SNAN
INTEGER TYPE,TRANO,BLOCKN,BLO CKT
THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING ON THE PAGE BEING CROOKED. THIS IS THE BEST IMAGE AVAILABLE.
COMMON /A/ TRAN(200,15), BLOCK(120,10), FACIL(35,5), STOR(35,7),
QUEUE(70,6), FEC(50,5), CECF, CECL, CECA,
STATUS, MOVE, C1, TRANG, BlockN, STARTC, RN(8), TRANA, FUNC(20,54)

C PARAMETER
IF(SNAC.NE.1) GO TO 50
RESULT = TRAN(TRANG, SNAN+5)
RETURN

C RANDOM NUMBER
50 IF(SNAC.NE.4) GO TO 100
I = SNAN
CALL RANDU(I, IY, YFL)
RESULT = YFL
RETURN

100 IF(SNAC.NE.3 .AND. SNAC.NE.0) GO TO 200
RESULT = SNAN
RETURN

C PRIORITY
200 IF(SNAC.NE.2) GO TO 300
RESULT = TRAN(TRANG, 3)
RETURN

C CLOCK
300 IF(SNAC.NE.5) GO TO 400
RESULT = C1
RETURN

C FUNCTION
400 IF(SNAC.NE.6) GO TO 500
CALL FUNCT(RESULT, SNAN)
RETURN

C VARIABLE
500 IF(SNAC.NE.7) GO TO 600
CALL VARIAB(RESULT, SNAN)
RETURN

C CONTENTS OF STORAGE
600 IF(SNAC.NE.8) GO TO 700
RESULT = STOR(SNAN, 1)
RETURN
C REMAINING STORAGE
  700 IF(SNAC.NE.9) GO TO 800
  RESULT = STOR(SNAN,2)
  RETURN
C UTILIZATION OF STORAGE
  800 IF(SNAC.NE.10) GO TO 900
  RESULT = STOR(SNAN,3)/IC1*(STOR(SNAN,1)+STOR(SNAN,2))
  RETURN
C AVERAGE OF STORAGE
  900 IF(SNAC.NE.11) GO TO 1000
  RESULT = STOR(SNAN,3)/C1
  RETURN
C MAXIMUM OF STORAGE
  1000 IF(SNAC.NE.12) GO TO 1100
  RESULT = STOR(SNAN,6)
  RETURN
C NO OF ENTRIES IN STORAGE
  1100 IF(SNAC.NE.13) GO TO 1200
  RESULT = STOR(SNAN,5)
  RETURN
C AVERAGE TIME IN STORAGE
  1200 IF(SNAC.NE.14) GO TO 1300
  RESULT = STOR(SNAN,3)/STOR(SNAN,5)
  RETURN
C STATUS OF FACILITY
  1300 IF(SNAC.NE.15) GO TO 1400
  RESULT = 1
  IF(FACIL(SNAN,1).EQ.0) RESULT = 0
  RETURN
C FACILITY UTILIZATION
  1400 IF(SNAC.NE.16) GO TO 1500
  RESULT = FACIL(SNAN,2)/C1
  RETURN
C NO OF ENTRIES IN FACILITY
1500 IF(SNAC.NE.17) GO TO 1600
    RESULT = FACIL(SNAN,5)
    RETURN
C AVERAGE TIME IN FACILITY
1600 IF(SNAC.NE.18) GO TO 1700
    RESULT = FACIL(SNAN,2)/FACIL(SNAN,5)
    RETURN
C QUEUE LENGTH
1700 IF(SNAC.NE.19) GO TO 1800
    RESULT = QUEUE(SNAN,5)
    RETURN
C AVE QUEUE LENGTH
1800 IF(SNAC.NE.20) GO TO 1900
    RESULT = QUEUE(SNAN,4)/C1
    RETURN
C MAXIMUM QUEUE LENGTH
1900 IF(SNAC.NE.21) GO TO 2000
    RESULT = QUEUE(SNAN,6)
    RETURN
C NO OF ENTRIES IN QUEUE
2000 IF(SNAC.NE.22) GO TO 2100
    RESULT = QUEUE(SNAN,2)
    RETURN
C NO OF ZERC ENTRIES IN QUEUE
2100 IF(SNAC.NE.23) GO TO 2200
    RESULT = QUEUE(SNAN,3)
    RETURN
C AVERAGE QUEUE TIME WITH ZERO
2200 IF(SNAC.NE.24) GO TO 2300
    RESULT = QUEUE(SNAN,4)/QUEUE(SNAN,2)
    RETURN
C AVERAGE QUEUE TIME WITH NO ZEROS
2300 IF(SNAC.NE.25) GO TO 2400
    RESULT = QUEUE(SNAN,4)/(QUEUE(SNAN,2)-QUEUE(SNAN,3))
    RETURN
C ERROR
2400 WRITE(6,2500) SNAC
2500 FORMAT(*INVALID SNA *,I4)
STOP
END

SUBROUTINE TERMIN(CECT,ID)

C*******************************************************************************
C
C
C SUBROUTINE TERMIN REMOVES A TRANSACTION FROM THE CEC AND
C DECREMENTS THE START COUNT.
C
C*******************************************************************************

IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER CECT
INTEGER TYPE,TRANO,BLOCKN,BLO CKT
COMMON /A/ TRAN(200,15), BLOCK(120,10), FACIL(35,5), STOR(35,7),
 QUEUE(70,6), FEC(50,5), FECF, FECL, F ECA, CEC(50,4), CECF, CE CL, CECA,
 STATUS, MCVE,C1,TRANQ,BLCKN,STARTC,RN(8), TRA NA, FUNC(20,54)

C UPDATE BLCK COUNTS
ILST = TRAN(TRANO,5)
BLOCK(ILST,9) = BLOCK(ILST,9) - 1
BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
MCVE = 1

C UNLINK BLCK FROM CEC
OLD1 = CEC(CECT,1)
OLD2 = CEC(CECT,2)
IF(OLD1.NE.0) CEC(OLD1,2) = OLD2
IF(OLD2.NE.0) CEC(OLD2,1) = OLD1
CEC(CECT,2) = CECA
CECA = CECT
IF(CECT.EQ.CECF) CECF = OLD2
IF(CECT.EQ.CECL) CECL = OLD1
IF(CECL.NE.0) GO TO 100
CECT = 0
ID = 1
GO TO 200
100  ID = 1
     CECT = OLD2
C DECREMENT START COUNT
200  SNAC = BLOCK(BLOCKN,2)
     SNAN = BLOCK(BLOCKN,3)
     CALL SNA(SNAC,SNAN,RESULT)
     STARTC = STARTC - RESULT
C REMOVE TRANSACTION
     TRAN(TRANC,1) = TRANA
     TRANA = TRANO
     RETURN
     END
SUBROUTINE TEST
C******************************************************************************
C TEST
C******************************************************************************
C SUBROUTINE TEST TESTS THE VALUE IN FIELD A AGAINST THE
C VALUE IN FIELD B. IF THE CONDITION IS MET THE TRANSACTION
C NEXT BLOCK IS CHANGED TO THE BLOCK SPECIFIED IN FIELD C
C OTHERWISE IT IS INCREMENTS BY ONE.
C******************************************************************************
C******************************************************************************
IMPLICIT INTEGER(C,F,Q,S,O)
REAL FUNC
INTEGER TYPE,TRANO,BLOCKN,BLOCKT
COMMON /A/ TRAN(200,15),BLOCK(120,10),FACIL(35,5),STOR(35,7),
       QUEU(70,6),FEC(50,5),FECF,FECL,FECA,CCEC(50,4),CCEF,CCEL,CCEA,
       ZSTATUS,MOVE,C1,TRANO,BLOCKN,STARTC,RN(8),TRANA,FUNC(20,54)
     TRAN(TRANC,1) = TRAN(TRANO,1) + 1
     MOVE = 1
     ILST = TRAN(TRANO,5)
     TRAN(TRANC,5) = BLOCKN
     BLOCK(BLOCKN,9) = BLOCK(BLOCKN,9) + 1
     BLOCK(ILST,9) = BLOCK(ILST,9) - 1
     BLOCK(BLOCKN,10) = BLOCK(BLOCKN,10) + 1
Determine Field A
SNAC = BLOCK(BLOCKN,2)
SNAN = BLOCK(BLOCKN,3)
CALL SNA(SNAC,SNAN,A)

Determine Field B
SNAC = BLOCK(BLOCKN,4)
SNAN = BLOCK(BLOCKN,5)
CALL SNA(SNAC,SNAN,B)

IF(BLOCK(BLOCKN,8).NE.1.0) GO TO 100

TEST L
IF (A.GE.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

100 IF(BLOCK(BLOCKN,8).NE.2.0) GO TO 200

TEST LE
IF (A.GT.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

200 IF(BLOCK(BLOCKN,8).NE.3.0) GO TO 300

TEST E
IF (A.NE.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

300 IF(BLOCK(BLOCKN,8).NE.4.0) GO TO 400

TEST NE
IF (A.EQ.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

400 IF(BLOCK(BLOCKN,8).NE.5.0) GO TO 500

TEST G
IF (A.LE.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

500 IF(BLOCK(BLOCKN,8).NE.6.0) GO TO 600

TEST GF
IF (A.LT.B) TRAN(TRANQ,1) = BLOCK(BLOCKN,6)
RETURN

ERROR
600 WRITE(6,700) BLOCK(BLOCKN,8)
700 FORMAT(*INVALID TEST RELATION*,I3)
STOP
END
SUBROUTINE TRANSF
C******************************************************************************
C
C
C SUBROUTINE TRANSF - IF FIELD A OF THE TRANSFER BLOCK IS
C BLANK THE NEXT BLOCK FIELD OF THE TRANSACTION IS SET
C TO FIELD B OF THE TRANSFER BLOCK. OTHERWISE, A RANDOM
C NUMBER IS GENERATED. IF THE GENERATED NUMBER IS LT THE
C SELECTION FACTOR IN FIELD A THE NEXT BLOCK IS SET TO FIELD
C C, ELSE TO FIELD P.
C
C******************************************************************************
C
IMPLICIT INTEGER(C,F,Q,S,O)
COMMON /A/ TRA(200,15),BLCK(120,10),FACIL(35,5),STOR(35,7),
IOQUEU(70,6),FEC(50,5),FECF,FECL,FECA,CEC(50,4),CECF,CECL,CECA,
STATU,MOVE,C1,TRANO,BLCCKN,STARTC,RN(8),TRANA,FUNC(20,54)
INTEGER TYPE,TRANO,BLCCKN,BLOKCT
REAL FUNC
MOVE = 1
C UPDATE BLOCK COUNTS
ILST = TRA(TRANO,5)
TRAN(TRANC,5) = BLCKN
BLCK(BLCCKN,9) = BLCK(BLCCKN,9) + 1
BLCK(ILST,9) = BLCK(ILST,9) - 1
BLCK(BLCCKN,10) = BLCK(BLCCKN,10) + 1
IF(BLCK(BLCCKN,3).NE.0.0) GO TO 100
TRAN(TRANC,1) = BLCK(BLCCKN,5)
RETURN
100 CALL RANDU(1,1Y,YFL)
C SNAN = BLCK(BLCCKN,3)
C SNAC = BLCK(BLCCKN,2)
C CALL SNA(SNAC,SNAN,RESULT)
RESULT = BLOCK(BLOCKN,3)
IF YFL .LE. RESULT GO TO 200
RETURN
TRAN(TRANC,1) = BLOCK(BLOCKN,5)
200 TRANS(TRANC,1) = BLOCK(BLOCKN,6)
RETURN
END
<table>
<thead>
<tr>
<th>NUMBER</th>
<th>LCC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I</th>
<th>COMMENTS</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>GENERATE</td>
<td>15, 5</td>
<td>JOBS ARRIVE EVERY 10 TO 20 MINUTES</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>TEST E</td>
<td>F2, 0, CLKC</td>
<td>IS CLERK B BUSY</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>QUEUE</td>
<td>ACLK</td>
<td>ENTER LINE FCR CLERK A</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>SEIZE</td>
<td>ACLK</td>
<td>CLERK A BEGINS PROCESSING</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>DEPART</td>
<td>ACLK</td>
<td>LEAVE LINE FCR CLERK A</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>ADVANCE</td>
<td>10, 3</td>
<td>CLERK TAKES 7 TO 13 MIN TO PROCESS JOB</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>RELEASE</td>
<td>ACLK</td>
<td>CLERK A COMPLETES JOB</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>QUEUE</td>
<td>BCLK</td>
<td>ENTER LINE FCR CLERK B</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>SEIZE</td>
<td>BCLK</td>
<td>CLERK B BEGINS PROCESSING</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>DEPART</td>
<td>BCLK</td>
<td>LEAVE LINE FCR CLERK B</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>ADVANCE</td>
<td>5, 2</td>
<td>CLERK B PROCESSES JOBS IN 3 TO 7 MIN</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>RELEASE</td>
<td>BCLK</td>
<td>CLERK B COMPLETES JOB</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>TERMINATE</td>
<td>1</td>
<td>JOB LEAVES OFFICE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CLKC</td>
<td>QUEUE</td>
<td>CCLK</td>
<td>ENTER LINE FCR CLERK C</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>SEIZE</td>
<td>CCLK</td>
<td>CLERK C BEGINS PROCESSING</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>DEPART</td>
<td>CCLK</td>
<td>LEAVE LINE FCR CLERK C</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>ADVANCE</td>
<td>20, 10</td>
<td>CLERK C TAKES 10 TO 30 MIN TO PROCESS</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>RELEASE</td>
<td>CCLK</td>
<td>CLERK C COMPLETES JOB</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>TERMINATE</td>
<td>1</td>
<td>JOB LEAVES OFFICE</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>START</td>
<td>1000</td>
<td>RUN SIMULATION FOR 1000 JOBS</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
SIMULATE
1 GENERATE 15, 5
2 TEST E F2, 0, 14
3 QUEUE 1
4 SEIZE 1
5 DEPART 1
6 ADVANCE 10, 3
7 RELEASE 1
8 QUEUE 2
9 SEIZE 2
10 DEPART 2
11 ADVANCE 5, 2
12 RELEASE 2
13 TERMINATE 1
14 QUEUE 3
15 SEIZE 3
16 DEPART 3
17 ADVANCE 20, 10
18 RELEASE 3
19 TERMINATE 1
20 START 1000
```
**BLOCK COUNTS**

<table>
<thead>
<tr>
<th>BLOCK CURRENT</th>
<th>TOTAL</th>
<th>BLOCK CURRENT</th>
<th>TOTAL</th>
<th>BLOCK CURRENT</th>
<th>TOTAL</th>
<th>BLOCK CURRENT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>11</td>
<td>0</td>
<td>658</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1660</td>
<td>12</td>
<td>0</td>
<td>658</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>698</td>
<td>13</td>
<td>0</td>
<td>658</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>698</td>
<td>14</td>
<td>0</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>698</td>
<td>15</td>
<td>0</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>698</td>
<td>16</td>
<td>0</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>698</td>
<td>17</td>
<td>0</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>698</td>
<td>18</td>
<td>0</td>
<td>207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>698</td>
<td>19</td>
<td>0</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>698</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FACILITIES**

<table>
<thead>
<tr>
<th>FACILITY</th>
<th>NUMBER</th>
<th>AVERAGE</th>
<th>UTILIZATION</th>
<th>DURING</th>
<th>TRANSACTION NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLK</td>
<td>698</td>
<td>10.624</td>
<td>.469</td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td>BCLK</td>
<td>698</td>
<td>5.617</td>
<td>.229</td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td>CCLK</td>
<td>352</td>
<td>18.563</td>
<td>.469</td>
<td></td>
<td>100.0</td>
</tr>
</tbody>
</table>

**QUEUES**

<table>
<thead>
<tr>
<th>QUEUE</th>
<th>MAXIMUM</th>
<th>AVERAGE</th>
<th>TOTAL</th>
<th>ZERO</th>
<th>PERCENT</th>
<th>AVERAGE</th>
<th>$AVERAGE</th>
<th>TABLE</th>
<th>CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLK</td>
<td>1</td>
<td>.038</td>
<td>698</td>
<td>627</td>
<td>89.8</td>
<td>.180</td>
<td>1.774</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCLK</td>
<td>1</td>
<td>.030</td>
<td>698</td>
<td>698</td>
<td>100.0</td>
<td>.000</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCLK</td>
<td>1</td>
<td>.003</td>
<td>302</td>
<td>287</td>
<td>95.0</td>
<td>.182</td>
<td>3.664</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$AVERAGE = AVERAGE \times \frac{\text{TRANSACTIONS}}{\text{TOTAL SEIZING}}$

**** TOTAL RUN TIME (INCLUDING ASSEMBLY) = .05 MINUTES ****
<table>
<thead>
<tr>
<th>BLOCK TYPE</th>
<th>A</th>
<th>15.000</th>
<th>B</th>
<th>5.000</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>12.000</td>
<td>15.000</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>14.000</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>15.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>16.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>7.000</td>
<td>0.0</td>
<td>10.000</td>
<td>0.0</td>
<td>0.0</td>
<td>3.000</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>15.000</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2.000</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>16.000</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>7.000</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3.000</td>
<td>0.0</td>
<td>5.000</td>
<td>0.0</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>8.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>15.000</td>
<td>0.0</td>
<td>3.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2.000</td>
<td>0.0</td>
<td>3.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>16.000</td>
<td>0.0</td>
<td>3.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>7.000</td>
<td>0.0</td>
<td>20.000</td>
<td>0.0</td>
<td>10.000</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3.000</td>
<td>0.0</td>
<td>3.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>8.000</td>
<td>0.0</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>17.000</td>
<td>0.0</td>
<td>1000.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>18.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>CLOCK</td>
<td>15061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK COUNTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK CURRENT</td>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>323</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>323</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACILITY</th>
<th>AVERAGE UTILIZATION</th>
<th>NUMBER ENTRIES</th>
<th>AVERAGE TIME/TRAN</th>
<th>SEIZING TRAN NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.454</td>
<td>677</td>
<td>10.057</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.226</td>
<td>677</td>
<td>5.019</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.451</td>
<td>324</td>
<td>20.551</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QUEUE</th>
<th>MAXIMUM CONTENTS</th>
<th>AVERAGE</th>
<th>TOTAL ENTRY</th>
<th>ZERO ENTRY</th>
<th>PERCENT ZERO</th>
<th>AVERAGE TIME/TRAN</th>
<th>$ AVERAGE TIME/TRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.004</td>
<td>677</td>
<td>643</td>
<td>94.978</td>
<td>0.080</td>
<td>1.588</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.0</td>
<td>677</td>
<td>677</td>
<td>100.000</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.005</td>
<td>324</td>
<td>299</td>
<td>92.284</td>
<td>0.216</td>
<td>2.800</td>
</tr>
</tbody>
</table>

$ AVERAGE TIME/TRAN EXCLUDES ZERO DELAY ENTRIES
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>OPERATION</th>
<th>A, B, C, D, E, F, G, H, I</th>
<th>COMMENTS</th>
<th>STATEMENT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>SIMULATE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>GENERATE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>TRANSFER</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>QUEUE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>SEIZE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>DEPART</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>RELEASE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>TRANSFER</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>QUEUE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>ENTER</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>DEPART</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>ADVANCE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>LEAVE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>TERMINATE</td>
<td>A, B, C, D, E, F, G, H, I</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- SIMULATE
- GENERATE
- TRANSFER
- QUEUE
- SEIZE
- DEPART
- RELEASE
- TRANSFER
- QUEUE
- ENTER
- DEPART
- ADVANCE
- LEAVE
- TERMINATE

- TWO REGULAR CHECKOUT COUNTERS
- CUSTOMERS ARRIVE FROM 1 TO 5 MINUTES APART
- 3.2
- ENTER EXPRESS QUEUE
- ENTER EXPRESS CHECKOUT
- LEAVE EXPRESS QUEUE
- 2
- TWO MINUTES REQUIRED FOR EXPRESS CHECKOUT
- LEAVE EXPRESS CHECKOUT
- ENTER REGULAR QUEUE
- ENTER REGULAR CHECKOUT
- DEPART REGULAR QUEUE
- 8
- 2 TO 14 MIN REQUIRED IN REGULAR CHECKOUT
- LEAVE REGULAR CHECKOUT
- 1
- CUSTOMER LEAVES STORE
- RUN 100 CUSTOMERS
- 100
### Facilities

<table>
<thead>
<tr>
<th>Facility Number</th>
<th>Average</th>
<th>Avergae Utilization During Total</th>
<th>Current</th>
<th>Percent</th>
<th>Transaction Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Storages

<table>
<thead>
<tr>
<th>Storage Capacity</th>
<th>Average Contents Entries</th>
<th>Average Utilization During Total</th>
<th>Current</th>
<th>Percent</th>
<th>Current Contents</th>
<th>Maximum Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Queues

<table>
<thead>
<tr>
<th>Queue Max</th>
<th>Average</th>
<th>Average</th>
<th>Total</th>
<th>Zero</th>
<th>Percent</th>
<th>Average</th>
<th>Average</th>
<th>Table</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

END

***** Total Run Time (including assembly) = .02 Minutes *****
<table>
<thead>
<tr>
<th>BLOCK TYPE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.000</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1.000</td>
<td>3.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>11.000</td>
<td>0.320</td>
<td>0.0</td>
<td>2.000</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>15.000</td>
<td>1.000</td>
<td>0.0</td>
<td>9.000</td>
<td>3.000</td>
</tr>
<tr>
<td>0</td>
<td>2.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>16.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>7.000</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>11.000</td>
<td>0.0</td>
<td>14.000</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>15.000</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>5.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>16.000</td>
<td>2.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>7.000</td>
<td>8.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>6.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>8.000</td>
<td>1.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>17.000</td>
<td>100.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>18.000</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>BLOCK COUNTS</td>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>102</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLOCK CURRENT</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILIZATION</td>
<td>0.266</td>
<td>2.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ENTRIES</td>
<td>37</td>
<td>64</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACILITY</th>
<th>AVERAGE UTILIZATION</th>
<th>NUMBER ENTRIES</th>
<th>AVERAGE TIME/TRAN</th>
<th>SEIZING TRAN NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.266</td>
<td>37</td>
<td>2.000</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STORAGE</th>
<th>CAPACITY</th>
<th>AVERAGE CONTENTS</th>
<th>AVERAGE UTILIZATION</th>
<th>ENTRIES</th>
<th>AVERAGE TIME/TRAN</th>
<th>CURRENT CONTENTS</th>
<th>MAXIMUM CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.719</td>
<td>0.260</td>
<td>64</td>
<td>7.469</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

| QUEUE CONTENTS | AVERAGE TOTAL ZEROS PERCENT AVERAGE $ AVERAGE CURRENT CONTENTS |
|----------------|-------------------|-------------------|-------------------|------------------|
| 1              | 0.007             | 37                | 35                | 94.555           | 0.054            | 1.000            | 0                |
| 2              | 0.791             | 64                | 29                | 45.313           | 3.438            | 6.286            | 0                |

$ AVERAGE TIME/TRAN EXCLUDES ZERO DELAY ENTRIES
try to move tran

execute block type subroutine

did tran enter next block

yes

termin.

no

advance block

yes

status change on

yes

overall scan

no

next tran

no
ENTER

field B blank

yes

B = 0

no

S2 ≥ B

no

put in push down delay chain set S7

return

yes

S3 = S3 + (C1 - S4) * S1
S4 = C1
S2 = S2 - B
S1 = S1 + B

S1 ≥ S6

yes

S6 = S1

no

S5 = S5 + B

return
Start

1. \( Q_4 = Q_4 + (C1 - Q1) \times Q_5 \)

2. \( Q_5 = Q_5 + 1 \)

3. Check if \( Q_5 > Q_6 \)
   - Yes: \( Q_6 = Q_5 \)
   - No: \( Q_2 = Q_2 + 1 \), \( Q_1 = C1 \)

End
Depart

Q4 = Q4 + (C1 - Q1) * Q5

Q5 = Q5 - B

Q1 = C1

C1 = T1

Q3 = Q3 + B (zero time delay entries)

return
Leave

B blank

no

B > S1

no

S3 = S3 + (C1 - S4) * S1
S2 = S2 + B

S1 = S1 - B
S4 = C1

delay

chain

no

return

yes

set all SSI's off

set status change on

B = 1

write error
B = S1
priority

insert priority in transaction

move tran last in new priority class in CEC

set status change on

return
termin.

unlink block from CEC

decrement start count by field A

return
advance

determine advance time

pos. time advance block

no
return

yes

add to FEC

remove from CEC

return
generate

create transaction

determine time

field C blank

no

add to time and set C to 0

yes

add to FEC

return
IGPSS

A General Purpose Simulation

Language for the Interdata

by

Martha Hoflich

B. A. North Texas State University, Denton, Texas, 1970

AN ABSTRACT OF A MASTER'S REPORT
submitted in partial fulfillment of the
requirements of the degree
MASTER OF SCIENCE

Department of Computer Science
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1977
The purpose of this project is to provide a simulation language that is portable to different hardware configurations. It is designed primarily for use on the Interdata B/32.

IGPSS is written in Fortran. It is based on GPSS and is used like GPSS with the following restrictions:

1. The input to IGPSS will be the anticipated output from the parser which remains to be written. The purpose of this project is to produce a program which will simulate a model given a set of strictly formatted input. A parser to be written as a future project will allow for more flexible input which more closely resembles the input required for GPSS.

2. The language statements available for use in IGPSS are a subset of those described in the GPSS/360 Introductory User's Guide (3).

3. Limits on number of parameters, facilities, etc. in IGPSS will be the same as the 64K version of GPSS.

The testing objective is to provide the same results using IGPSS as the same model would produce with GPSS. Documentation will consist of instructions for the use of IGPSS and any differences between it and GPSS.