# UNITED ARAB REPUBLIC

## THE INSTITUTE OF NATIONAL PLANNING

Memo. No. 912

# GENERAL PURPOSE SYSTEMS SIMULATOR (GPSS).

BY

ALWALID ELSHAFEI

OPERATIONS RESEARCH GROUP

JUNE 1969

CAIRO

3, MOHAMED MAZHAR - ZAMALEK

"Opinions Expressed and Positions Taken by Authors are Entirely Their Own and do not Necessarily Reflect the Views of the Institute of National Planning".

INTRODUCTION:

The General Purpose Systems Simulator (GPSS) is principally a language chosen to define basic actions characteristic of discrete systems in general. In GPSS, a physical system can be described completely enough for simulation by block diagrams familiar to many engineers. These diagrams, interpreted directly on the computer, produce automatically the various statistical quantities necessary to measure the performance of the system.

The primary elements in GPSS are, simply, the blocks, connections between them, and transactions which flow through them. A typical model may have hundreds of blocks. Each of the many different block types describes a particular kind of action, which may possibly be a simple time delay. The time delay, which is the time that the transaction must spend in that block, may be specified in many different ways. Some blocks represent queues, others represent the use of facilities or equipment, others create and control transactions which enter the network. When a transaction enters a block, the action defined by that block is performed and the transaction is routed to another block in this way, widely diverse systems can be depicted and simulated.

The basic idea in GPSS-as in most discrete systems simulations- is to create a flow of transactions and to control them as they move step by step through the network. There may be hundreds of these transactions in the system at any time, including some of which, for example, might be competing for the service of a particular facility. A transaction, which is a quantity of information, may represent a message entering a communications network, a pulse in an electrical circuit, a work-order entering a shop, or a shopper entering a supermarket.

Each transaction may carry with it certain information necessary to describe its characteristics. In GPSS, this information is contained in the parameters included in the description of the transaction. Where the transaction represents a message, for example, the parameters might contain message length, destination, and the kind of equipment it is to use. The time to process this transaction (delay time), or the decision as to what equipment it should use, or the selection of a successor block for example, can be a function of the characteristics of the transaction.

The routing of the transaction through the network may be accomplished in many ways - on the more or less predetermined basis indicated above, for example, or on a dynamic basis, where the next block selection depends on a state of the system not known in advance, or on the basis of a statistical distribution. The point is that GPSS permits this variety in the ways transaction flow is regulated -a necessity if complex systems are to be represented.

Among the other features in GPSS which facilitate the writing of simulation programs is a built-in timing device, the main pulse of the program. This device, a counter representing a clock, is automatically updated to the time when the next event is to take place in the system; that is, the "clock" is stepped in variable increments, the unit of time being determined by the user. The queueing mechanism in the program is structured on a first-in-first-out basis within priority classes, and all queues are maintained automatically by the program.

During the running of a model, GPSS accumulates all the statistics required to reflect the state of the system at various points of time. Summarized and printed out at the end of every run, these statistics answer many questions about the performance of the system. They contain information about queues, system loading, flow of traffic, and other system variables

vital to systems analysis. The level of detail is seen if we take queue statistics as an example: total transactions entering each queue, average time spent in the queue, average size and maximum size of the queue, etc. More frequent printouts, easily requested by the user, can provide an even more detailed picture of the system. GPSS has been applied successfully to a wide variety of simulation studies by many who are not programmers. Its success lies in the fact that it is easy to learn and apply, and that it is versatile. It has proved that a language based on block diagrams can accommodate many complex systems, representing them in manageable dynamic models without oversimplifying them.

TWO IMPORTANT FUNCTIONS:

Simulating message handling. The shown example (FIG 1) of the GPSS program illustrates the use of some of the blocks to simulate part of a computing system. It involves a central processor unit (CPU), data channels, random access disc files, and the several types of messages entering the system for processing. The size of the model depends on the complexity of the system and the detail desired in the simulation.

The Originate block creates and enters transactions which represent messages into the system. The entry rate may vary or be constant, according to the function specified. The Assign block adds a code to a transaction to identify the message type. If other characteristics of the messages are needed-length or destination-additional Assign blocks are used.

The Queue block, which receives the transation, represents the queues of messages waiting for the CPU. The GPSS program gathers and prints out statistics about the queue, such as its maximum size and the average retention time for the transaction. Representing the CPU is the Hold block, which normally has a delay time, the actual time the CPU needs to process a message at this stage.

The Compare block routes each transaction according to its type as it leaves the Hold block. The message type determines which stream of blocks will receive it for processing. Some transactions then enter a Queue block for the random access disc file.

Representing the actuator is the combination of a Seize block and a Release block. The actuator comes into use when the transaction enters the Seize block. Then the transaction enters the Hold block, which represents a data channel, finally leaving it and entering the Release block. This action returns the actuator to the next transaction that needs it. The other transactions continue in a similar manner, flowing through the blocks which represent the real system.

Transaction flow and control. As in most simulations, the basic idea in GPSS is to create a flow of transactions, and control them as they move step-by step through the network. A transaction, which is a quantity of information, may represent a message entering a communications network, a pulse in an electrical circuit, or a work order entering a shop it depends on the system being simulated. Each transaction carries the information necessary to describe its characteristics.

In GPSS, this information is contained in prameters which are actually a part of the transaction's description. In cases where a transaction represents a message, the parameters might contain the message length, its destination, and the kind of equipment it uses. The time to process this transaction (delay time), or the decision as to what equipment it should use-or the selection of a successor block can be a function of the transaction. There may be hundreds in the system at any time, some of them competing for the same facility.

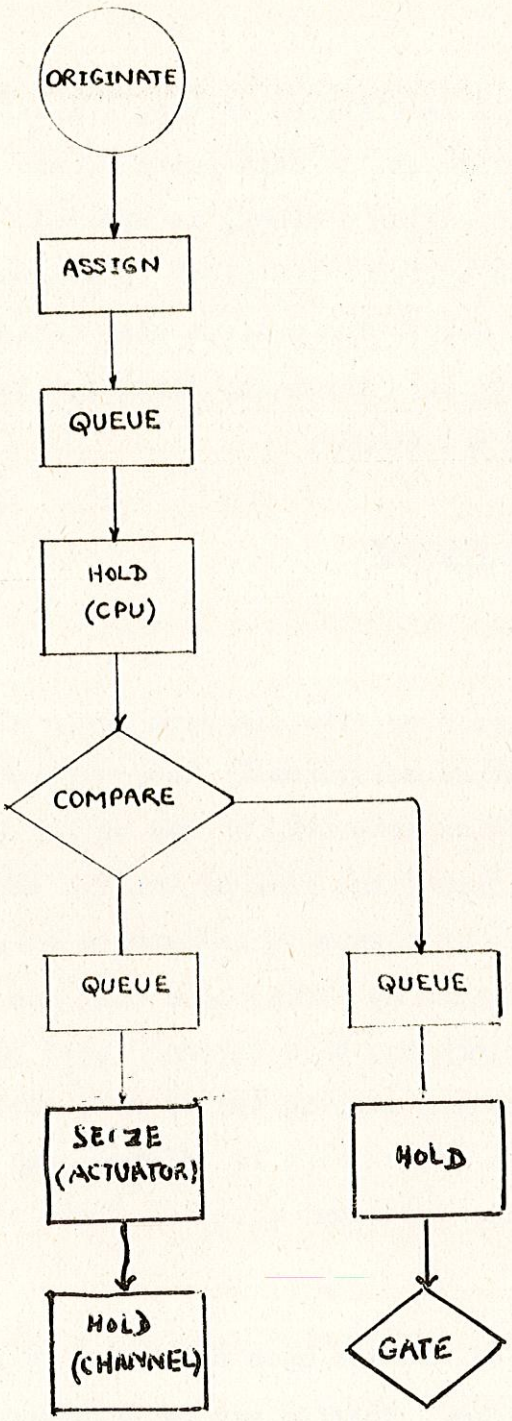
ORIGINATE
ASSIGN
QUEUE
HOLD
(CPU)
COMPARE
QUEUE
QUEUE
SEIZE
(ACTUATOR)
HOLD
HOLD
(CHANNEL)
GATE

FIG. I

One way GPSS facilitates writing simulation programs is a built-in timing device serving as the main pulse of the program. This device, which is a counter representing a clock, is updated to the time the next event takes place in the system. The clock is stepped in variable increments determined by the user. The program also contains a queuing mechanism which is structured on a first-in, first-out basis, with priorities. The program maintains all queues.

## PRINCIPLES OF THE PROGRAM

### BLOCK DIAGRAMS

Block diagrams or flow diagrams are a widely used method of describing the structure of systems. They consist of a series of blocks, each block describing some step in the action of the system. Lines joining the blocks indicate the flow of traffic through the system and describe the sequence of events to be carried out. Alternative courses of action are represented by having more than one line leaving a block. Conversely, one block may have several lines entering it to indicate that this block is a common step in two or more sequences of events. The choice of paths, where an alternative is offered, may be either a statistical event or a logical choice dependent upon the state of the system at the time of choice.

The units of traffic upon which the system operates depend on the nature of the system. Traffic may be messages in a communication system, electrical pulses in a digital circuit, work items in a production line, or any number of other factors. The units upon which a system operates will be described in this program as "transactions".

Although a block diagram is a commonly used means of describing systems, the notation used in normal block diagrams depends upon the system and the person describing the system. For the purpose of this program certain conventions have been established for constructing block diagrams. A number of block types and system concepts have been defined, each corresponding to some basic action or condition that may occur in a system. To be simulated, a system must first be represented in terms of these concepts and block types. The program will then create transactions, move them through the specified blocks, and execute the actions associated with each block.

CLOCK TIME

The program operates by moving transactions from block to block of the simulation model in a manner analogous to the way in which the units of traffic they represent progress in the real system. Each such movement is an event that is due to occur at some point in time. The program maintains a record of the times at which these events are due to occur, and it proceeds by executing the events in their correct time sequence. Where transactions are obstructed and cannot move at the time they should, the program moves them as soon as the obstructing conditions change.

In order to be able to maintain the events in the correct time sequence, the program simulates a clock that is recording the instant of real time that has been reached in the real system. The number shown by this clock is referred to as the "clock time".

All times in the simulation are given as positive integers. The unit of system time represented by a unit change of clock time is implied by the user, who enters all program data relating to times in terms of the unit he has selected. Whatever unit of time is chosen, such as the millisecond or the tenth of an hour, it must be used consistently throughout a simulation.

## BLOCKS

A system to be simulated must be reducible to a series of operations performed on units of traffic. The program provides block types which perform operations in a model of the system equivalent to the actions occurring in the real system.

## ACTION TIMES

Each block may specify a number of clock units that the transactions are to spend at that block. The number may be constant, or computed from a statistical distribution.

## ROUTING

Every block specifies the next step in the block diagram to which a transaction will be sent at the completion of its computed time interval. The user assigns each block an identifying number and uses the numbers to designate the path of flow.

If desired, two such numbers, termed next block A and next block B, may be used at one block to indicate alternative exits from the block. In this case the block must also designate a selection mode to be used in choosing between the two paths.

## TRANSACTIONS

The units of traffic that are created and processed by the simulator are called transactions. They have certain properties which correspond to characteristics of traffic in a system. This section will describe those properties and the ways in which specific data may be introduced in the block diagram.

Each transaction possesses a transit time. This number normally represents the total time for which the transaction has been in the system. In many situations the overall transit time in a system is a less significant number than the transit time between selected points. A block type, the MARK block, is provided to set the transit time of a transaction to zero so that a new interval of time may be measured. Since the simulator clock only assumes the integral values, the transit time of a transaction is always an integer.

Each transaction also carries eight parameters. These numbers, which are integers, may have a variety of interpretations. The user decides which of the possible interpretations are relevant, and introduces parameter values accordingly. In contrast to the transit time of the transaction, which has the simulation time unit implicitly associated with it, the eight parameters do not have any built-in significance. For example, the user may reserve one parameter for the character length of a message in a communication system, another to represent the terminal from which the message was sent, and a third to indicate the destination of the message. Another user may wish a parameter to contain the dollar value of an order, and a secord parameter to carry the due date for the order. It is not necessary for the simulation program to know what significance the user attributes to each parameter. A block type, the ASSIGN block, is provided to set the values of the transaction parameters.

Each transaction has a priority level associated with it. The priority level is an integer between 0 and 7 inclusive. Whenever transactions compete for equipment, the one with the numerically higher priority will be the first to be processed. If both transactions have the same priority, however, the transaction that has been delayed longer will be selected first. A transaction retains its priority level until it enters another PRIORITY block.

## EQUIPMENT

In a system there are components that interact with the units of traffic. Some components, such as lathes, perform operations on the traffic. Others, such as communication trunks, may interact more passively with the system traffic. In order to simulate a system, the program user defines a correspondence between these components and elements of the block diagram. The great variety of components found in systems may be classified into unit-processing equipment and batch-processing equipment. The convention used in this program is to term the unit-processing elements "facilities" and the bactch-processing elements "storages". A unit-processing component of a system, such as a lathe, may thus be assigned an identifying number J, and thereafter be referred to in the block diagram as facility J. Similarly a communication trunk capable of transmitting several messages simultaneously would correspond to a numbered storage in the block diagram convention. The following two sections will describe the manner in which the program treats each type of equipment.

## FACILITIES

A facility may be utilized by only one transaction at any moment in time. The program retains a record of which transaction is utilizing each facility in the block diagram. If another transaction attempts to use a facility that is already in use, that transaction will be delayed until the facility becomes available. A record is also maintained of the total number of clock units for which each facility was in use. This figure is used to compute an average utilization. In addition, a count is made of the number of times each facility was used, and the average duration of each use is computed. Thus the simulated activity of the unit-processing elements is measured in much the same way as the corresponding activity in the real system would be measured.

Six block types, the HOLD, SEIZE, RELEASE, INTERRUPT, PREEMPT and RETURN blocks, are provided for transactions to make use of unit-processing elements in the block diagram. The basic principle of their action is as follows. When a transaction enters a HOLD block, for example, that transaction is recorded as using the facility specified at the HOLD block. When the transaction leaves the HOLD block, the facility is returned to the available state, the usage count for the facility is incremented by one, and the in-use time for the facility is incremented by the number of clock units that the transaction remained in the HOLD block.

Unit-processing components of a system may be subject to interruption in their processing of traffic. A lathe, for example, may be used for small corrections to a finished part, after which work on an unifinished part will continue. This occurrence can be represented in the block diagram. The use of a facility by one transaction can be temporarily interrupted by another transaction. That facility will still be considered in-use for logical and statistical purposes, and at the completion of the interrupt, the transaction that was originally using the facility will continue its usage.

STORAGES

Batch-processing elements of the block diagram are termed "storages". The user defines the capacity of each storage that he refers to in his block diagram, and the program retains a record of the number of units of this space that are occupied at any moment in time. If a transaction attempts to occupy more space than is available, that transaction will be delayed until sufficient space becomes available. A record is kept of the total number of units that enter or leave each storage. The program also measures the average contents of each storage, and the average number of clock units each transaction remained in that storage. These statistics may be

used to determine how efficiently the batch-processing elements of the block diagram are being utilized, and whether sufficient capacity has been provided.

Three block types, ENTER, LEAVE and STORE, are provided so that batch-processing elements may be represented in the block diagram. Each such block specifies a storage number and a number of units to be occupied or returned. Each transaction entry will usually correspond to one storage unit, but if desired a transaction may specify the number of units it represents in one of its parameters. When a transaction enters an ENTER block, the appropriate number of units is added to the contents of the storage specified at the ENTER block. The record of the total number of units to enter that storage is also incremented by the same value. When the transaction exits from the ENTER block, no change is made in the contents of the storage. A separate block, the LEAVE block, is used to reduce the contents of a storage. The STORE block combines the actions of ENTER AND LEAVE.

LOGICAL CONDITIONS

In a system the flow of traffic may be conditionally delayed or diverted. Orders are delayed if a lathe is already occupied. Messages may be sent via an auxiliary circuit if a particular trunk is overloaded. This section will describe how the program treats such logical conditions.

Before a transaction is permitted to enter a HOLD block the program tests the status of the facility specified at the HOLD block. This facility must be in the available state in order for the transaction to enter the block. Other block types also have logical conditions which must be satisfied before a transaction is permitted to enter them. In order to simulate the decisions made in a system, blocks with conditional entries may be placed in the path of transactions.

The program provides two alternatives if a transaction is refused entry at a block. The transaction may simply be delayed until conditions are met, or an alternate path may be tested. The choice is not specified, however, at the block causing the delay. The alternate path, if one is given, must be specified at the block from which the transaction is attempting to exit. The following figure illustrates the two method:

Method A:
Each transaction uses facility 5 for 500 clock units. Transactions will be delayed in block 10 if they find facility 5 in use when they attempt to exit from block 10.

Method B:

This method allows transactions to be diverted if they find facility 5 in use when they attempt to exit from block number 10. The transactions will enter block number 30 if they are refused entry to the HOLD block. The selection mode at block 10 is "BOTH", meaning that Next Block A is tried first.

QUEUES

In a system the traffic may be delayed by the unavailability of an item of equipment. When this occurs the delayed traffic is said to form a queue. This same term is used in the block diagram convention. Transactions that are delayed in their attempt to enter a block constitute a queue.

One of the major functions served by the simulation program is the measurement of queues, but not all queues in the block diagram are automatically measured. The user may specify QUEUE blocks at selected points where significant queues are expected, and the maximum and average length of these queues will be measured. The program also maintains a count of the number of units that enter each queue, in much the same manner that it does for storages. A separate count is made of the number of units that were delayed in each queue, since it may be significant in a system that a certain percentrage of the traffic was processed without delay. A record is retained of the average delay encountered by the transactions that entered each queue. These delays may also be tabulated in the form of a frequency distribution.

PROGRAM TABLES

Thus far we have described the basic concepts of the block diagram convention used by the simulation program. The user may find it helpful to know how his block diagram is stored in the computer. A table is reserved within the program for each type of information associated with the block diagram. There is a table for the blocks that are used, as well as tables for the facilities, storages and queues. The program uses the identifying number assigned by the user when information is placed in the tables. For example, if the user assigns block number 10 to a HOLD block, information concerning that block will be entered in the tenth location in the tenth location in the block table. The information will consist of the block type, the block time specification, the selection mode, next blocks A and B, and the facility number that is to be held. If the user specifies facility number 20, the facility records maintained in the twentieth location in the facility table will be updated whenever block 10 is entered. Similarly, the information relating to queue number 50, such as its current length, is recorded in the fiftieth location of the table reserved for queues.

The simulation program thus converts a block diagram model of a system into numerical entries in memory tables. It simulates the occurrence of an event by changing one or more of the table entries. The information contained in the prog.am tables may also be referenced by the user. For example, he may obtain the current length of queue number 50 by means of the symbol Q50. The program supplies the queus length, which is simply an integer stored in the fitieth location of one of the queue tables.

## SYSTEM VARIABLES

In the preceding section reference was made to the fact that items of information about the system can be obtained from the program tables by the use of mnemonic references such as Q for the queue length. These quantities are generically termed "system variables". They represent the current properties or states of the system as the simulation proceeds. For example, whenever a reference is made to Q120 the current contents of queue number 120 are obtained. By making such a reference at a COMPARE block, for example, the user may simulate a system whose flow logic depends dynamically on the length of a queue within the system.

EXAMPLE : A GAS STATION:

At this point, let us consider an application of GPSS to a simulation problem. The problem chosen simplifies the basic concepts and, when grasped, will help the reader in extrapolating the ideas to other applications. We are concerned with the operational performance of a gas station: economics of time, space, equipment, money. This particular station has one pump, one attendant and space on forecourt for 4 waiting cars. We are interested in learning what happens if the loads on the system are varied, say, by increasing the arrival rate of customers. What is the capacity of the system, that is, how many cars can be serviced bourly or daily, based on 9-hours active day? What service delays might be expected and what would be their effect? What would be the effect of adding a second attendant or more pumps, or other devices, to speed service, And so on.

Steps of The Simulation Process:

1 -Identify the significant elements in the system and define their operational characteristic:

- The customers
- The pumps
- The space at the pumps
- The waiting space
- The attendant.

Customer characteristics:

. Arrival rate

. Service request

Attendant characteristics:

. Work procedure

. Time he takes to service each customer.

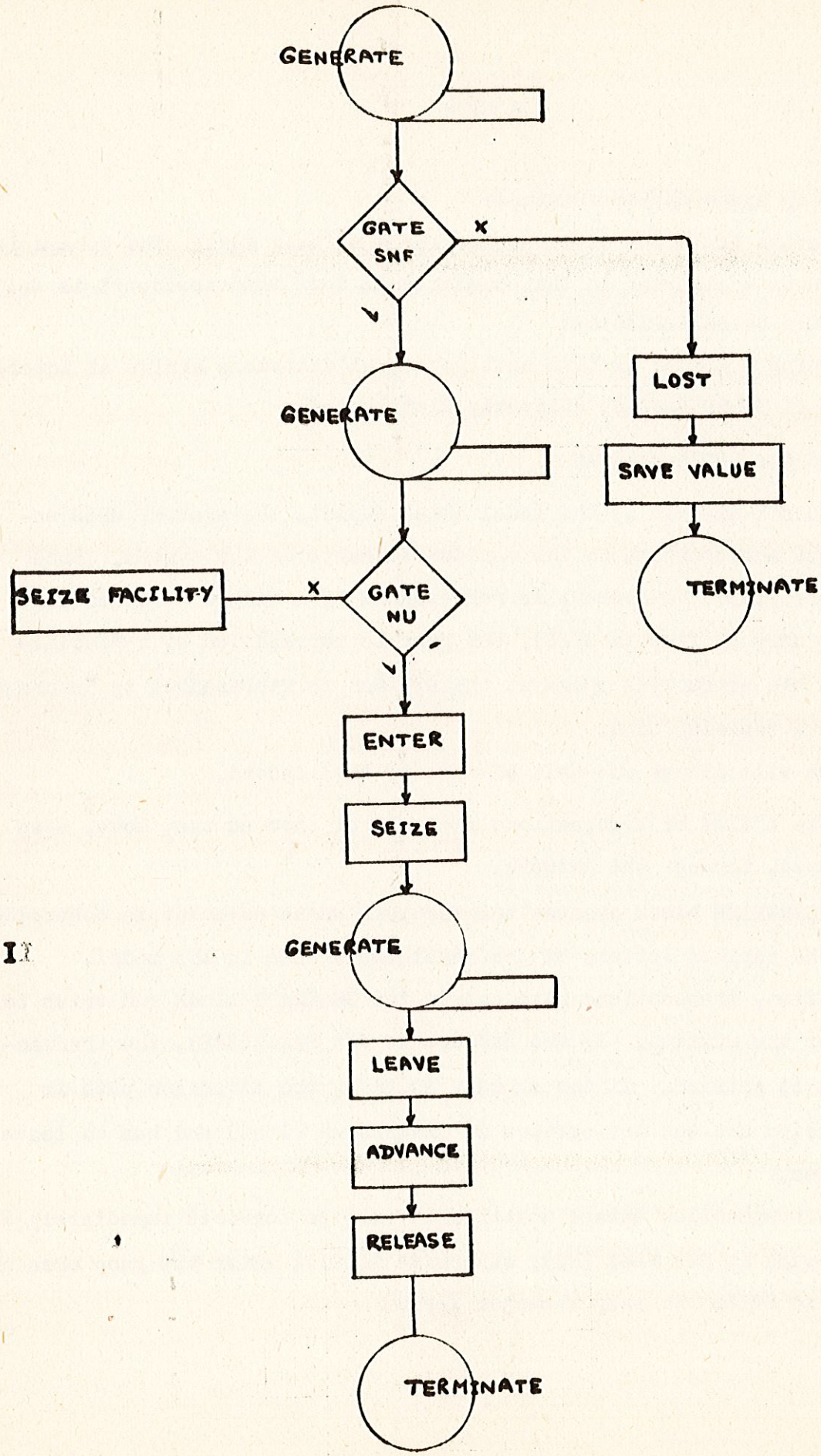
GENERATE
GATE SNF
X
LOST
SAVE VALUE
TERMINATE
GENERATE
SEIZE FACILITY
X
GATE NU
ENTER
SEIZE
GENERATE
LEAVE
ADVANCE
RELEASE
TERMINATE

FIG. II

Waiting space characteristics:

Garage has space on forecourt for 4 waiting cars. The garage is sited on a clearway so that customer is lost when forecourt is full.

Service characteristics:

Using 1 second as time unit potential customers arrive at intervals of 30-60 seconds uniformly distributed.

2 - Build the BLOCK DIAGRAM

The block diagram is the model which depicts the system. GPSS entities are assigned to the various elements of the system. Each automobile, or customer, is represented by a single "transaction" (the unit of flow in GPSS), the pump is represented by a "facility"; the automobile space at the station is represented by "storage" with a capacity of 4.

We will choose the unit of time to be 1 second.

3 - Create a flow of transactions and control them as they move, step by step, through the network.

The GENERATE block creates transactions whose movement is controlled by the various actions of the remaining blocks in the model.

First, transactions move out of the GENERATE block and tries to enter the station. If the Storage is Not Full (SNF), the transaction is entered. If the storage is full, the alternate path is selected and the transaction is terminated (lost) and has to leave system.

When transaction enters station, it will be serviced immediately if the pump is Not Used (NU); otherwise it will enter the pump area and try to SEIZE the pump whenever free.

Service time, for the car being serviced, is generated and when the car is readily serviced, the transaction LEAVES the queue, ADVANCES to the facility to be serviced, then RELEASES facility. That far, this transaction is TERMINATED.

It has to be noted that the simulator calculates different times during which the transaction is delayed & serviced. With the aid of a clock it keeps track of simulated time as well.

TABLE I

| QUANTITY OF INTEREST | SMALL STORAGE CAPACITY | LARGE STORAGE CAPACITY |
|---|---|---|
| Capacity (No. of cars) | 4 | 10 |
| Average utilization of facility | .999 | .998 |
| Number of entries (to facility) | 181 | 178 |
| Average time/transaction (inservice) | 178.839 | 181.747 |
| Average contents of storage | 3.850 | 9.783 |
| Average utilization of storage | .962 | .978 |
| Number of entries (to storage) | 185 | 188 |
| Average time/transaction (in storage) | 674.378 | 1686.148 |
| Current contents (of storage) | 4 | 10 |
| Maximum contents (of storage) | 4 | 10 |
| Lost business | 517 | 528 |

When run, this model provides statistical information about system performance, answering the kind of questions we started with. Service delays, if any, would be indicated, as well as component loading statistics. The percentage utilization of the facility is a standard output.

If the facility indicates a high utilization for example, perhaps the addition of another facility (pump) should be considered. The trade-off is between the additional salary of an attendant and the lost profit due to customers leaving. Other information which might also be obtained is the average time a customer remained in the station, the number of customers serviced, and so on.

Once a model is prepared using GPSS, incidentally, it is quite simple to run. One card is punched for each block in the model. These cards, plus several definition and control cards, are placed with the GPSS interpretive program and run directly on the computer.

The printout for the cards, together with results of run are shown hereafter. A slight modification was made, resulted in adding a new part to the logic, that is of increasing the storage space in order to be big enough for 10 waiting cars. The whole program is rerun and results are also shown. Finally, a summary of obtained results, in both cases, appears in table I.

Operations in a real gas station are of course far more complex and uncertain than this simple example shows. Arrival rates of customers will vary over the day, for example, and some may elect not to enter the station if there is a waiting line. The logic to handle such variations could be built into our model. In a dynamic system such as this, changes depend upon the previous activities of the system. When the attendant is tired (predictably, at the end of his shift), customer service time might be longer or he might otherwise alter his service procedure. The more fidelity built into a simulation model, the more complex it will be. The secret is to choose the right degree of fidelity.

This example could just as well have been a model of an integrated steel mill, an air traffic pattern, a telephone exchange, or one of many other industrial and engineering systems. The same concepts apply, and similar approaches are used.

## CONCLUSION

The declining costs of general-purpose digital computers in relation to their improved performance characteristics make feasible, for the first time, the solution of large and complex simulation problems. The basic techniques and approaches discussed here have been demonstrated and reported for many industrial and engineering simulation applications. As our simulation tools become more efficient and enough experience accumulates to suggest and substantiate a "theory" of simulation, we may be able to solve dynamic problems which now seem too intricate for simulation.

```
// JOB EMA
// ASSGN SYS000,X'191'                                          00.01.24
// EXEC GASP1
```

```
BLOCK
NUMBER   *LOC    OPERATION  A,B,C,D,E,F,G          COMMENTS                                  CARD NUMBER
         *       SIMULATE                                                                        1
         *                                                                                       2
1                GENERATE   45,15       CARS PASS AT 1/2 TO 1 MINUTE INTERVALS                  3
2                GATE SNF   1,LOST      TRADE LOST IF FORECOURT FULL                            4
3                ENTER      1           TAKE PLACE ON FORECOURT                                 5
4                SEIZE      1           GET PUMP WHEN AVAILABLE                                 6
5                LEAVE      1           BY MOVING UP TO PUMP                                    7
6                ADVANCE    180,60      FUEL CAR IN 2 TO 4 MINUTES                              8
7                RELEASE    1           LEAVE PUMP                                              9
8                TERMINATE              LEAVE GARAGE                                            10
9          LOST  SAVEVALUE  1+,K1       SUPERFLUOUS BLOCK TO COUNT LOST BUSINESS                11
10               TERMINATE              CAR DRIVES PAST GARAGE                                  12
11               GENERATE   3600        DEVICE TO SIMULATE RUN TIME                             13
12               TERMINATE  1           FOR A TOTAL OF 9 HOURS                                  14
               1 STORAGE    4           ALLOW FOR UP TO 4 WAITING CARS                          15
                 START      9                                                                   16
                 CLEAR                                                                          17
         *       SIMULATE WITH LARGER FORECOURT TO SEE EFFECT ON SALES                          18
               1 STORAGE    10                                                                  19
                 START      9                                                                   20
                 END                                                                            21
```

3

```
    BLOCK NUMBER    SYMBOL      REFERENCES BY CARD NUMBER

                9   LOST            4
```

SIMPLE GARAGE-1 PUMP, 4 WAITING

```
*
 1      GENERATE    45      15
 2      GATE SNF    1       9
 3      ENTER       1
 4      SEIZE       1
 5      LEAVE       1
 6      ADVANCE     180     60
 7      RELEASE     1
 8      TERMINATE
 9      SAVEVALUE   1+      K1
 10     TERMINATE
 11     GENERATE    3600
 12     TERMINATE   1
 1      STORAGE     4
        START       9
```

RELATIVE CLOCK 32400 ABSOLUTE CLOCK 32400

BLOCK COUNTS

| BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 702 | 11 | 0 | 9 | | | | | | | | | |
| 2 | 0 | 702 | 12 | 0 | 9 | | | | | | | | | |
| 3 | 4 | 185 | | | | | | | | | | | | |
| 4 | 0 | 181 | | | | | | | | | | | | |
| 5 | 0 | 181 | | | | | | | | | | | | |
| 6 | 1 | 181 | | | | | | | | | | | | |
| 7 | 0 | 180 | | | | | | | | | | | | |
| 8 | 0 | 180 | | | | | | | | | | | | |
| 9 | 0 | 517 | | | | | | | | | | | | |
| 10 | 0 | 517 | | | | | | | | | | | | |

| FACILITY | AVERAGE UTILIZATION | NUMBER ENTRIES | AVERAGE TIME/TRAN | SEIZING TRANS. NO. | PREEMPTING TRANS. NO. |
|---|---|---|---|---|---|
| 1 | .999 | 181 | 178.839 | 5 | |

| STORAGE | CAPACITY | AVERAGE CONTENTS | AVERAGE UTILIZATION | ENTRIES | AVERAGE TIME/TRAN | CURRENT CONTENTS | MAXIMUM CONTENTS |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 3.850 | .962 | 185 | 674.378 | 4 | 4 |

CONTENTS OF FULLWORD SAVEVALUES (NON-ZERO)

| SAVEVALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 517 | | | | | | | | |

```
        CLEAR
*       SIMULATE WITH LARGER FORECOURT TO SEE EFFECT ON SALES
1       STORAGE    10
        START      9
```

RELATIVE CLOCK 32400 ABSOLUTE CLOCK 32400

BLOCK COUNTS

| BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL | BLOCK | CURRENT | TOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 716 | 11 | 0 | 9 | | | | | | | | | |
| 2 | 0 | 716 | 12 | 0 | 9 | | | | | | | | | |
| 3 | 10 | 188 | | | | | | | | | | | | |
| 4 | 0 | 178 | | | | | | | | | | | | |
| 5 | 0 | 178 | | | | | | | | | | | | |
| 6 | 1 | 178 | | | | | | | | | | | | |
| 7 | 0 | 177 | | | | | | | | | | | | |
| 8 | 0 | 177 | | | | | | | | | | | | |
| 9 | 0 | 528 | | | | | | | | | | | | |
| 10 | 0 | 528 | | | | | | | | | | | | |

| FACILITY | AVERAGE UTILIZATION | NUMBER ENTRIES | AVERAGE TIME/TRAN | SEIZING TRANS. NO. | PREEMPTING TRANS. NO. |
|---|---|---|---|---|---|
| 1 | .998 | 178 | 181.747 | 5 | |

| STORAGE | CAPACITY | AVERAGE CONTENTS | AVERAGE UTILIZATION | ENTRIES | AVERAGE TIME/TRAN | CURRENT CONTENTS | MAXIMUM CONTENTS |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 9.783 | .978 | 188 | 1686.148 | 10 | 10 |

CONTENTS OF FULLWORD SAVEVALUES (NON-ZERO)

| SAVEVALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE | NR. | VALUE |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 528 | | | | | | | | |

END