Systems engineers have come to recognize simulation as a valuable tool in their work. However, writing simulation programs can be a difficult, time consuming task requiring intricate and extensive programming. For simulation to be most useful, it must be possible to carry out a simulation quickly and be possible to change the simulation easily as the system design proceeds.

□ *This paper describes a general purpose simulation program designed to simplify the task of simulating systems. It is applicable to a wide variety of important problems. The program features a simple block diagram language with which to describe the system to be simulated. Given this description, the program will automatically simulate the system.*

A general purpose systems simulator

by G. Gordon

In the course of his work the systems engineer is constantly faced with the problem of analyzing the performance of proposed systems configurations. He needs to design the operating conditions of the proposed system to give optimum performance and determine whether the configuration will meet the specifications laid down for the system. The performances of individual components of the system are usually well understood but, with the increasing size and complexity of systems, it has become very difficult to estimate the performance of the many possible combinations of components that need to be considered in the course of designing a system.

In the face of these difficulties emphasis is being placed upon the use of computer simulation as a technique for aiding systems engineering. Suitably programmed, a digital computer can indicate how a proposed system will perform by computing, step by step, the response of a mathematical model of the system. By itself, simulation does not design systems, it merely shows how a model of a particular configuration performs. The problems of selecting the configuration of a system and judging the performance from the results indicated by the simulation remain with the systems engineer. However, properly used, simulation can provide valuable design information by showing in some detail how the components of a system can be expected to behave.

Care must be taken in the way simulation is used, particularly on the question of how the model used in the simulation cor-

responds to the system being studied. Some of the problems to be considered in employing simulation in systems engineering are discussed in another paper.[1] The purpose of this paper is to describe a specific program that has been designed to simplify the task of creating and running system simulations.

the simulator

The program to be described is called the General Purpose Systems Simulator, or GPSS, and it has been written for operation on the IBM® 704, 709 and 7090. The program allows the user to study the logical structure of the system, to follow the flow of traffic through the system and to measure the effects of blocking caused by the need to time-share parts of the system or caused by limiting the capacity of parts of the system. Outputs of the program give information on:

1. The amount of traffic that flows through the complete system or through part of the system.
2. The average time and the distribution of time for traffic to pass through the complete system or between selected points of the system.
3. The extent to which elements of the system are loaded, together with the distribution of the occupancy of storage elements in the system.
4. The maximum and the average queue lengths occurring at various parts of the system, together with the distribution of the occupancies of the queues.

Statistical variations can be introduced in the simulation and arrangements are made to sample the state of the system at various points of time. The effect of assigning levels of priority to units of traffic can be studied. It is also possible to simulate the effects of peak loads by varying the load on the system with time or by varying speeds of operation with load.

simulation preparation

The problem of preparing a computer simulation falls into two main tasks. First a model of the system to be simulated must be constructed and then a computer program must be written to "run" the model. The model is usually expressed as a set of mathematical and logical equations that represent the response of the individual system components and describe the interactions between the components. From such a description of the model, a programmer can produce a computer program that will run the model.

The process can be simplified by providing a language or a prescribed method of describing a model and writing a program that will run any model that is defined in this manner. The task of writing individual programs is then removed or greatly reduced. Usually such programs are restricted to use on a particular class of problems, such as, for example, a job shop simulator. GPSS is a program of this general type that employs a language designed for describing simulation models of systems. The program will run a simulation directly from a description of the model given in this language. The user, therefore, need only know the language used for describing the models and does not need to be able to

program for the computer on which the program operates.

The language has not been chosen for any particular class of systems. The aim has been to define certain basic actions that are characteristic of systems so that the program can be applied to a wide variety of different systems. Following a description of the principles and the language of the program, two examples of its application will be given, one to a typical traffic flow problem and one to a simple production system. Two other examples of its application to data-processing systems will be found in a paper[1] by E. C. Smith, Jr.

block diagrams as simulator language

The program is based on the use of block diagrams as a means of describing the structure and action of a system. The method of describing systems with a block diagram is well-known. Each block of the diagram represents a step in the action of the system and lines joining the blocks indicate the sequence of events than can occur. To make a block diagram description of a system serve as the input to a simulation program, however, a number of conditions must be met. First, the meaning of the blocks used in the diagram cannot be left to the individual user but must be clearly defined to the program. Secondly, it is necessary to associate with each block a number, that will be called the *block time,* to represent the execution time of the action the block is simulating. Thirdly, conventions need to be established to control the way in which the selection of the succession of blocks is made.

To meet the requirements for a well-defined block diagram a set of 25 specific block types has been designed, each chosen to be representative of some basic system action. To use the program, the system must be described in terms of a combination of blocks selected from this set of 25 block types. Each type may be employed any number of times subject only to the overall limit that the total block diagram may not contain more than 2047 blocks.

Each block type is distinguished by a name which is descriptive of its action and it is also given a characteristic symbol to be used when drawing block diagrams. When the block diagram is complete, each individual block is also given an identification number called the *block number.* These block numbers can usually be assigned in an arbitrary manner between the values 1 and 2047.

block time

The block time is an integer giving the number of time units required for the action represented by the block. The magnitude of the time unit is determined by the program user. The unit is not specifically entered in the program but is implied by entering all times throughout the block diagram in terms of this same basic time unit.

The time required for the action being represented by a block is not always well-defined. It may be an ill-defined quantity or it may vary over a range of values, often in a random manner. Arrangements are made to introduce random variations in the block times where necessary. A simple rectangular distribution of time can be introduced at any block by specifying two numbers, the

mean and the *spread* for the block. Whenever the program refers to the block, it will choose at random a block time between the values of mean minus spread and mean plus spread, with equal probability being given to each integer in this range. If the block time does not vary, the spread is set zero to make the block time a constant value equal to the mean. If desired, the mean may also be set zero to represent actions that take no time or take a negligible amount of time compared with the basic time unit.

There are many occasions when the random selection of a block time from a simple rectangular distribution is an adequate representation of the system action or is the only representation of a random process that can be made on the available information. At other times, a more precise random distribution is known or the variation of block times is not random but depends upon some factor associated with the system. In these cases use can be made of tables of numbers referred to as *functions* to introduce a more accurate representation of the block time. The definition of these functions and their use will be described later.

alternative actions

To represent the alternative courses of action that may be followed by the system, more than one line may leave a block. Correspondingly, one block may be entered by way of more than one line to indicate that the step represented by that block occurs in more than one sequence of events. The convention is made that, with the exception of one block type called the BRANCH block, not more than two lines may lead from a given block. On the other hand, no restriction is placed on the number of lines leading into a given block. If a larger number of paths is required to represent the results of a particular decision, it is always possible to use a network of blocks with zero block time, that will lead to the required number of alternative paths. The BRANCH block type that was just mentioned is a block that allows up to 127 alternative paths and can be used to avoid such networks when the choice between paths follows some simple rules.

block exits

All block types other than those that represent terminal points of the system, can have two exits. The exits are distinguished by referring to them as exits 1 and 2 and they are defined by giving the number of the block to which the exit leads. The blocks the exits lead to are referred to as *next blocks* 1 and 2. If there is to be only one exit from a block it must be defined as next block 1. If there are two possible exits from a block both next block 1 and 2 must be defined and, in addition, a number called the *selection factor, S,* must be given at that block to determine the choice between the exits. In the case of the BRANCH block, the block numbers entered for exits 1 and 2 determine the range of block number from which the selection is to be made. The set of blocks that can be reached from a BRANCH block must, therefore, be numbered consecutively, otherwise there is no restriction on the manner of numbering blocks.

The selection factor can be used to make two types of decisions in determining the exit to be followed from a block. If the selection factor is set between 0 and 1 then a random choice will be

made on every occasion an exit is made. The probability of choosing exit 1 will be 1-S, and the probability of choosing exit 2 will be S.

The other method of deciding between exits is to choose exit 1 if next block 1 is available at the time of the decision and take exit 2 if exit 1 is not available. If neither exit is available then the first exit to become available is selected. This mode of operation is indicated by setting S equal to 1. It is an important mode of operation that can be used to simulate alternative lines of action in the system when some component of the system is found to be busy.

transactions and items of equipment

The system represented by the block diagram is operating upon certain basic units that move through the system. The nature of these units depend upon the system. For example: in a communication system the units might be messages; in a traffic study they might be people or vehicles; in a data processing system they might be records and so on. For convenience, the unit is referred to in the simulation as a *transaction.* The simulation proceeds by creating transactions to represent these units and moving the transactions through the block diagram in the same manner as the units would progress through the system represented by the block diagram.

The system being studied will also involve certain physical components which operate upon the transactions individually or in groups as they proceed through the system. To simulate the effects of these components, the simulation includes elements referred to as *items of equipment.* Certain of the block types are concerned with the interaction between transactions and items of equipment.

The principal property of an item of equipment that is of interest in a simulation study is the limit of its capacity to handle transactions simultaneously. The existence of such limits can cause congestion and a significant part of any system simulation is concerned with measuring the effects that these limits have on the overall performance of a system.

stores and facilities

A distinction is made between two types of equipment according to whether the capacity for handling transactions is limited to one transaction or more than one transaction. An item of equipment that can handle only one transaction at a time is called a *facility.* An item of equipment that can handle many transactions simultaneously, up to a specified limit, is called a *store.* Up to 511 facilities and 511 stores can be employed in the simulation. They are identified by number and the block types that are concerned with equipment refer to the number of the particular item of equipment they employ. Because of the capacity limits set by equipment, blocks that employ equipment can cause congestion when the equipment is fully engaged by transactions. These blocks will then be unavailable to a transaction attempting to enter. In these circumstances, the transaction can be made to wait until room is made available or it can be diverted to some other course of action by use of the selection factor described before.

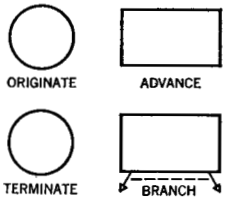
The interpretation placed upon stores and facilities depends upon the system being simulated. In a communication system, for example, a trunk might be represented as a store with capacity equal to the number of lines in the trunk while a terminal passing one message at a time would be a facility. In a vehicle traffic study, a road might be represented by a store while a toll booth would be represented by a facility and so on. The representation of system components by items of equipment will also depend upon the type of study being conducted. In one study, for example, a computer complex may be represented as a single item of equipment. Another study may require more detail and treat the major parts of the computer complex such as the memory, channels, disk files, etc., as separate items of equipment, each with its characteristic capacity.

Items of equipment that do not correspond to a system component are sometimes introduced into a simulation to effect control over the flow of transactions. If, for example, there are two or more parts of a system such that only one part can be in use at a time, entrance to any one part can be made contingent upon seizing a facility that will then block off the other parts. Similarly, taking up capacity in a store can be made to limit the number of transactions allowed in any one part of a system. Some of the 25 basic block types will now be described and an example using these blocks will then be given to illustrate the method of using the program. Not all the block types will be described; a more complete description of the program can be found in reference 2.

creation, destruction and movement of transactions

Figure 1 illustrates four block types concerned with creating, destroying and moving transactions without involving equipment. The block type called ORIGINATE creates transactions and enters them into the simulation. The block time of this type represents the interval of time between successive creations. Creation of transactions continues even if the transactions are unable to leave the ORIGINATE block when the exit blocks are unavailable. For this reason, the mean at an ORIGINATE block may not be set at zero since this would represent an infinite rate of generation. A TERMINATE block removes transactions from the system the instant they enter the TERMINATE block. This block therefore has no block time nor does it have any exits.

Figure 1



An ADVANCE block is used to represent any action requiring time but not involving equipment. Since it does not involve equipment it cannot cause congestion; it is often used, therefore, with a zero block time as a buffer at the exit of a block using equipment to ensure that the equipment is released. It may also be used with zero time and a selection factor of 1 to precede a block using equipment in order to divert transactions when the equipment is busy.

use of facilities

The BRANCH block is similar to an ADVANCE but it allows up to 127 exits from the block. The selection of the exit follows special rules, however. The selection factor must be either 0 or 1. In the former case, a random selection from all possible exits is made with equal probability being given to each. With $S=1$, an attempt

Figure 2

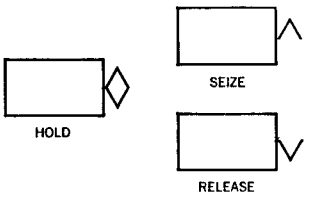
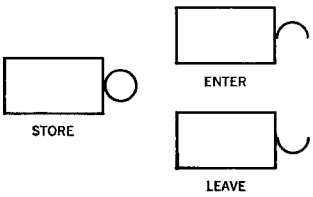


Figure 3



is made to leave by the lowest numbered exit; if this is unavailable, the next highest is tried and so on. If all exits are busy, the transaction waits for the first exit to become available.

Figure 2 shows a group of three blocks concerned with the use of facilities. In each case the flag on the side of the block symbol is used to carry the number of the facility associated with the block.

The HOLD block allows a transaction that enters the block to engage the facility for as long as the transaction remains in the block. The SEIZE block allows a transaction to engage the facility upon entering the block. The transaction, however, keeps control of the facility when it leaves the SEIZE block and remains in control until it enters a RELEASE block associated with the same facility. Between the points at which the transaction seizes the facility and releases the facility any number of blocks may be inserted to represent the actions followed by the transaction while it has control of the facility. The same facility may be mentioned at many different points of the block diagram indicating the different places in the system requiring its use.

use of stores

In Figure 3 there are three blocks associated with stores, the associated store being numbered in the flag attached to the symbol. The actions of these blocks are analogous to the three facility type blocks of Figure 2. The STORE block allows a transaction to occupy space in the store associated with the block for as long as the transaction is in the block. The ENTER block allows a transaction to take up space in the store but the space is not given back until a transaction enters a LEAVE block.

modes of operation

Three modes of operation are allowed for each of these blocks, differing in the amount of space that is controlled by the transaction. In a *normal* mode, only one unit of space is involved. In a *parameter* mode, the amount of space depends upon a number called a parameter that is associated with the transaction. The generation and use of parameters will be described later. In the third mode, called the *total* mode, the entire store is filled or emptied as the transaction takes up or gives back space.

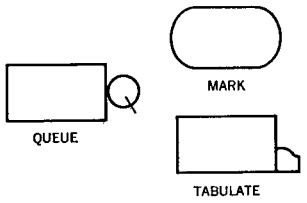
The same store may be referred to by several blocks each employing any of the modes. A common count is maintained of the total space occupied by all such blocks. Any time there is insufficient space available for a transaction it will be refused entrance to a STORE or ENTER block but other transactions requiring less space may still be able to advance. An important distinction exists between the way the program treats transactions engaging facilities and entering stores. The transaction that engages a facility at a SEIZE block is the only transaction that may release the facility. In the case of STORE, however, one transaction can take up the space at an ENTER block while a different transaction can give back the space at a LEAVE block.

gathering statistics

One of the main objects of using a simulation program is to gather statistics about the estimated performance of the system being simulated. Some of the blocks are concerned with gathering statistics rather than representing system actions. These are il-

lustrated in Figure 4. As has been pointed out, blocks involving equipment can cause congestion if the equipment involved is busy. As a result blocks that do not use equipment, such as the ADVANCE, may contain any number of transactions waiting for equipment to become available. The program will maintain such transactions in a queue which is served on a first-come first-served basis. The program will not, however, maintain any statistics about these queues unless they occur in a QUEUE block. These blocks, therefore, are placed in positions where congestion is anticipated, such as immediately in front of a HOLD block. The program will measure the average queue size and the maximum queue size occurring in each QUEUE block. If desired, the program will also give the distribution of the queue length sampled at uniform intervals of time. As many as 511 QUEUEs can be included in the simulation. The number of the queue is indicated in the flag attached to the QUEUE block symbol.

Figure 4



Another important set of statistics that are frequently wanted are the transit times of transactions in getting from one point of the system to another. These statistics can be gathered by using MARK and TABULATE blocks. The program makes a note of the current clock time on each transaction that enters a MARK block. Later, when the transaction arrives at a TABULATE block, the program notes the clock time upon arrival, subtracts the MARK time placed on the transaction by the MARK block and enters the difference in a table. There can be as many as 63 tables in the simulation. The table associated with a TABULATE block is numbered in the flag attached to the block symbol. Each table has 16 tabulation intervals and maintains frequency counts of the number of entries falling in each tabulation interval. The tabulation intervals are set by the program user at the beginning of the simulation run.

supermarket example

As an example of how the program is used, consider the problem of measuring the flow of traffic through a supermarket. Shoppers enter a supermarket, but before shopping they must each get a basket. The number of baskets is limited to 150 and if none is available the shopper leaves the supermarket without shopping. Having obtained a basket, the shopper spends some time shopping, checks out at a counter and then leaves, returning the basket on the way out. Two types of shopper will be assumed, express shoppers and non-express or normal shoppers. Separate check-out facilities are available for the two types of shopper. For the express shopper there is only one counter but for the normal shopper there are 7 counters.

Figure 5 Get basket

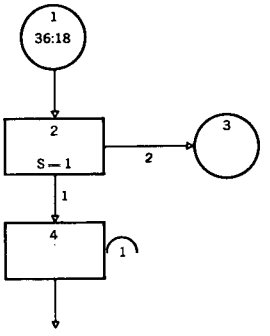
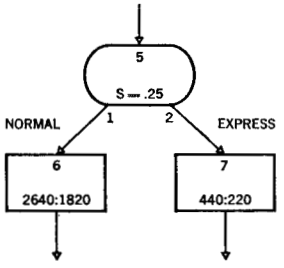


Figure 5 illustrates the section of the block diagram concerned with the actions of getting a basket. An ORIGINATE block creates transactions, each representing one shopper. The convention adopted in drawing the blocks is that the block number is placed at the top center of the block and the mean and spread, if any, are placed at the center separated by a colon. The mean is at the left and the spread on the right. The ORIGINATE block, for example, is block number 1 and has a mean of 36 units and a spread

of 18. The unit of time chosen in this example is 1 second so that the average time between arrivals is 36 seconds, but the time may vary from 18 to 54 seconds.

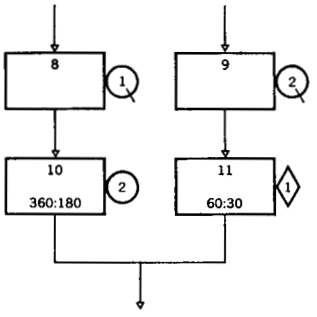
To represent the baskets, a store, number 1, is defined with a capacity of 150, equal to the number of baskets. To get a basket, the transaction representing the shopper must move into an ENTER block, number 4, associated with store number 1. The attempt to enter is made through an ADVANCE block, number 2, with zero block time and S=1. The ENTER block is at exit 1 of this ADVANCE block so the transaction will move into the ENTER block if there is space available in store 1, that is, if there is a basket available. If this is not so, the transaction moves to a TERMINATE block, number 3, indicating that the shopper leaves because no basket is available.

Figure 6 Shop



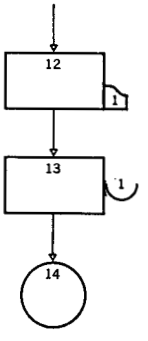
Having got a basket the shopper moves into the supermarket and shops. This part of the simulation is illustrated in Figure 6. One factor to be measured in this simulation will be the time shoppers spend in the supermarket. The transactions are therefore sent to a MARK block to note on each transaction the time at which shopping begins. The MARK block is also used to divide the flow of transactions into two streams representing the express and normal shoppers. It will be assumed that 25% of the shoppers are express so a selection factor of 0.25 is set in the MARK block. Shopping will be represented by simple ADVANCE blocks, one for each type of shopper. The average shopping time is set to be 44 minutes for the normal and 7 minutes 20 seconds for the express shoppers. When the transaction leaves these ADVANCE blocks, shopping is complete and the shoppers move to the check-out counters.

Figure 7 Check-out



The check-out section of the block diagram is represented in Figure 7. It is to be expected that there will be congestion at the counters and the program will be arranged to measure the queues. The transactions therefore go to one of two QUEUE blocks immediately preceding the blocks representing the check-out equipment. Since there is only one check-out counter for express shoppers, this counter will be represented by a facility number 1 and a HOLD block is used to represent the action of occupying the check-out counter. Each transaction representing an express shopper moves, in turn, into the HOLD block for service and occupies the counter for a time ranging from ½ to 1½ minutes. It then moves on making room for the next shopper in the queue.

Figure 8 Leave



For the normal shoppers, there are seven check-out counters and these are represented by a store, number 2, of capacity 7. The STORE block operates in a normal mode so that up to seven transactions at a time can be in the STORE block. Each stays for a time varying between 3 and 9 minutes and then moves on.

Having left the check-out counter the customers prepare to leave the supermarket. Both streams are brought together at this point and move to the section illustrated in Figure 8.

Each transaction was marked at the time of beginning to shop. Since they are now about to leave the supermarket, the

transactions are sent to a TABULATE block to enter into table 1 a measure of the total time the shopper spent in the supermarket. The basket is then returned by entering a LEAVE block that returns the space taken up at the ENTER block. Finally, the transactions are removed at a TERMINATE block.

The complete diagram appears as shown in Figure 9. Each block is numbered and the exits for each block are identified. One card is punched for each block. The cards have a simple fixed field format. Figure 10 illustrates the completed coding form from which the cards for this problem are punched.

Included in the coding form are certain control cards required to describe the simulation run. A JOB card identifies the beginning of the problem deck and a START card defines the end of the problem deck. REMARKS cards allow comments to be printed in the output listing. The START card in this case indicates that the run begins at zero time with an initially empty condition and that the simulation is to run until 1200 shoppers have passed through the supermarket. For each store used in the simulation a CAPACITY card defines the capacity of each store. For each table associated with a TABULATE block a TABLE card defines the tabulation intervals. In this case table 1 has intervals of 500 units beginning at 500 as lower limit.

One output that can be obtained from the program is a count of the number of times a block is entered. If this information is required for a block, a 1 is entered in column 65 of the BLOCK card. The 3 entered for the TERMINATE block indicates that not only is a block count to be gathered but, in addition, the entries to this block count towards the end of the simulation.

When the simulation is completed a series of outputs are printed and these are illustrated in Figure 11. The first output, not shown in Figure 11, is a listing of the input cards defining the

Figure 9 Supermarket problem

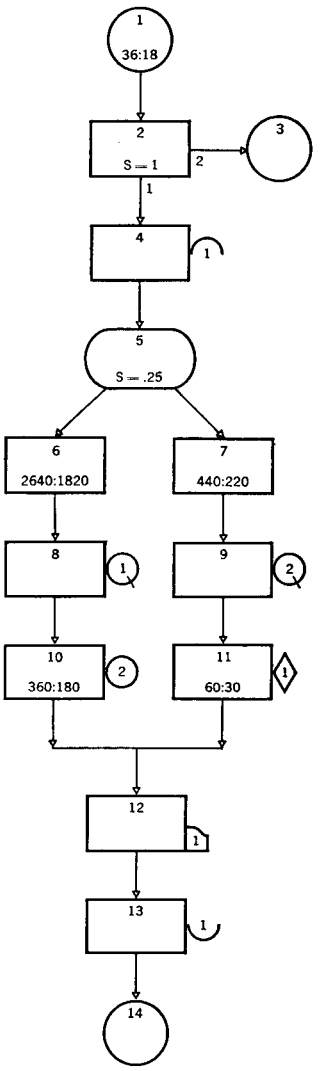


Figure 10 Completed coding form for supermarket problem

IBM General Purpose System Simulator
(Block Cards)

PROBLEM ELEMENTARY SUPERMARKET MODEL — PAGE 1 OF 1
CODER Newman — DATE July 2, 1962

FIELD NO. 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
BLOCK TYPE 2	BLOCK NUMBER 11	MEAN 15	SPREAD 21	PRIORITY 27	BOOL. IND. 30	SELECT FACTOR 33	TAG OR MODE 37	NEXT BLOCK 41	NEXT BLOCK 2 45	TABLE NO. OR ASSIGN FUNT. 49	FACILITY NO. 53	STORE NO. 57	QUEUE NO. 61	BLOCK COUNT 65	REMARKS 66
JOB	SUPERMARKET, ELEMENTARY MODEL														
REMARKS	ILLUSTRATES BASIC MODEL BUILDING AND SYSTEMS STUDY METHODS.														
REMARKS	BASIC TIME UNIT IS 1 SECOND														
REMARKS															
ORIGINATE	1	36	18					2						1	
ADVANCE	2					1		4	3					1	
TERMINATE	3													1	
ENTER	4							5				1		1	
CAPACITY	1	150													
MARK	5					.25		6	7					1	
ADVANCE	6	2640	1320					8						1	
ADVANCE	7	440	220					9						1	
QUEUE	8							10					1	1	
QUEUE	9							11					2	1	
STORE	10	360	180					12				2		1	
CAPACITY	2	7													
HOLD	11	60	30					12			1			1	
TABULATE	12							13		1				1	
TABLE	1	500	500												
LEAVE	13							14				1		1	
TERMINATE	14													3	
START	0	1200													
END															

Figure 11 Output of supermarket simulation

NO TIMES BLOCKS ENT TIME END SIM 47640

BLOCK	COUNT	BLOCK	COUNT	BLOCK	COUNT
1	1313	2	1312	3	
4	1312	5	1312	6	965
7	347	8	910	9	345
10	862	11	345	12	1200
13	1200	14	1200		

FACILITY NR 1 FRACTION OF TIME IN USE .4278

STORE NR	STORAGE CAPACITY	AVERAGE UTILIZATION
1	150	.5962
2	7	.9367

QUEUE NR	MAX QUEUE LGTH	AV QUEUE LGTH
1	50	27.1471
2	4	.1360

TABLE NUMBER 1 MODE 0 TOTAL NUMBER IN TABLE 1200

TOTAL TIME IN TABLE 3971800 MEAN OF TABLE 3309.833

VARIANCE OF TABLE 3789437.0938

LIMIT	NO	PER CENT	CUM	MULT OF MEAN
500	148	12.33	12.33	.1511
1000	196	16.33	28.67	.3021
1500		.00	28.67	.4532
2000	4	.33	29.00	.6043
2500	9	.75	29.75	.7553
3000	55	4.58	34.33	.9064
3500	91	7.58	41.92	1.0575
4000	127	10.58	52.50	1.2085
4500	161	13.42	65.92	1.3596
5000	148	12.33	78.25	1.5107
5500	135	11.25	89.50	1.6617
6000	91	7.58	97.08	1.8128
6500	31	2.58	99.67	1.9638
7000	4	.33	100.00	2.1149
7500		.00	100.00	2.2660
		.00	100.00	2.4170

problem. The time at which the simulation ended is shown followed by the block counts that have been mentioned. Information is then given about each facility, store and queue in the simulation. For the facilities the program gives the fraction of total time for which the facility was engaged. For the stores, the average occupancy is given and for the queues, the average and maximum queue lengths are shown. Finally, the tables of statistics that were requested are printed.

refinement of supermarket model

The example that has been given is a relatively simple model of a supermarket. It would be simple to refine the model in a number of ways by including more blocks. For example, the shopping area could be represented by a number of blocks, each representing a different section of the supermarket and each having a certain probability of being entered. The section representing getting the baskets could be elaborated so that a percentage of customers enter the supermarket without bothering to get a basket, or it could be arranged that customers who need a basket wait a while before leaving if no basket is available.

The block times that are not zero or constant in this example have all been represented by means and spreads giving random selections from simple rectangular distributions. One important respect in which the problem can be elaborated is to use functions and parameters to introduce more realism in these block times.

functions and parameters

A *function* is a table of up to eight pairs of numbers defining the relationship between an input quantity X and an output Y. These values define the function at the specified points and the program interpolates linearly between these points. A total of 31 such functions is allowed, each being identified by a number 1 to 31. The functions themselves are defined by entering two cards, one defining the X values and the other defining the corresponding Y values. The functions can operate in one of four modes depending on the nature of variable X used as an input quantity.

The output from a function can be used for two purposes; to control a block time or to supply a parameter. When used to control a block time the function output Y is used to multiply the normal block time determined by the use of a mean and spread. If the mean is set to 1 and the spread to 0, the function value Y becomes the block time directly. Use of a function to control block times in this way can be called for at any block by entering the function number of the BLOCK card.

One mode of operation for the function is a *random number* mode in which the input X is a number selected at random between 0 and 1. Operating in this mode, the functions allow a more complex distribution of block times than the simple rectangular distribution given by a mean and spread alone.

A *parameter* is a number that is derived from a function and attached to any transaction that enters a block type called ASSIGN, designed especially for this purpose. The BLOCK card identifies the function from which the assignment is to be made. The significance placed upon the parameter is determined by the program user. In the supermarket example given above, it could be used to represent the number of items the shopper purchases. With the version of the program described in reference 1 only one parameter can be attached to each transaction. Other versions of the program allow more than one parameter to be attached to any transaction.

One use of parameters has already been mentioned. They may be used to control the number of units of space a transaction occupies in a store. Another use is that they can be used to provide the input X employed when using a function, thus making a block time depend upon the parameter of the transaction entering a block. This is called the *parameter* mode of operating functions. In the supermarket, for example, suppose an ASSIGN block with a function operating in a random number mode has been used to attach a parameter representing the number of purchases. Another function expressing the relationship between the number of purchases and check-out time could be employed at blocks 10 and 11 to control the check-out times according to the number of purchases.

Two other modes of operating the functions are a *clock* mode in which the input X is the current clock time and a *store* mode in which the input X is the current number of spaces occupied in a particular store. The *clock* mode can be used to introduce the effects of peak loads on a system by making the block time at ORIGINATE blocks depend up real time. The *store* mode can be used to introduce the effects of congestion on operating times. For example, the times at the blocks representing the shopping could be made to depend upon the number of people in the supermarket as indicated by the space occupied in store number 1.

When using functions for the purpose of assigning parameters, the function may be used in any of the four modes that have been described. In particular, the parameter mode may be used so that the parameter on a transaction at the time of entering an ASSIGN block can be used to reassign the parameter. In later versions of the program it is also possible to assign numbers directly to parameters or add and subtract numbers to the exiting parameter. For example, an ASSIGN block with the statement P1EK10 will assign the number 10 to parameter 1 or P2-K4 will subtract 4 from the current value of parameter 2.

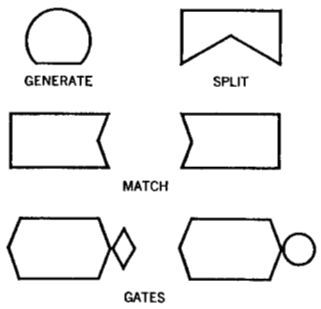
generation and flow of transactions

In the description of the program given so far, the only decisions that can be made in the model are in the choice of exits made with the use of the selection factors. Certain of the block types are concerned with introducing control over the generation and flow of transactions in other ways. These are illustrated in Figure 12.

The ORIGINATE block described before creates transactions irrespectively of whether the transactions can leave the ORIGINATE block and proceed into the system. A GENERATE block also creates transactions but only so long as the transactions can enter the system. This allows the system itself to decide when to enter a transaction by arranging that the exit to the GENERATE block is normally blocked and is only unblocked when a transaction is required. The block time still controls the interval between successive arrivals of transactions. Unlike an ORIGINATE block, however, the block time can be zero so that, if desired, a batch of transactions can be entered simultaneously.

The SPLIT block allows one transaction to be created by another. Every transaction entering a SPLIT block produces a copy which leaves by way of exit 2 while the original transaction leaves by way of exit 1. Pairs of transactions that have been created at a SPLIT block can be synchronized in their movement by the use of a pair of MATCH blocks. The two MATCH blocks are arranged to cross-refer to each other by recording each other's block number as exit 2. When a transaction that has been split arrives at one MATCH block it will wait there until the other member of the pair arrives at the other MATCH block. Both transactions may then proceed by leaving their respective MATCH blocks by way of exit 1. This allows the user to simulate the execution of simultaneous processes that must both be completed before proceeding to another stage of the processing. If desired, transactions may be split sev-

Figure 12



eral times to represent more than two simultaneous processes.

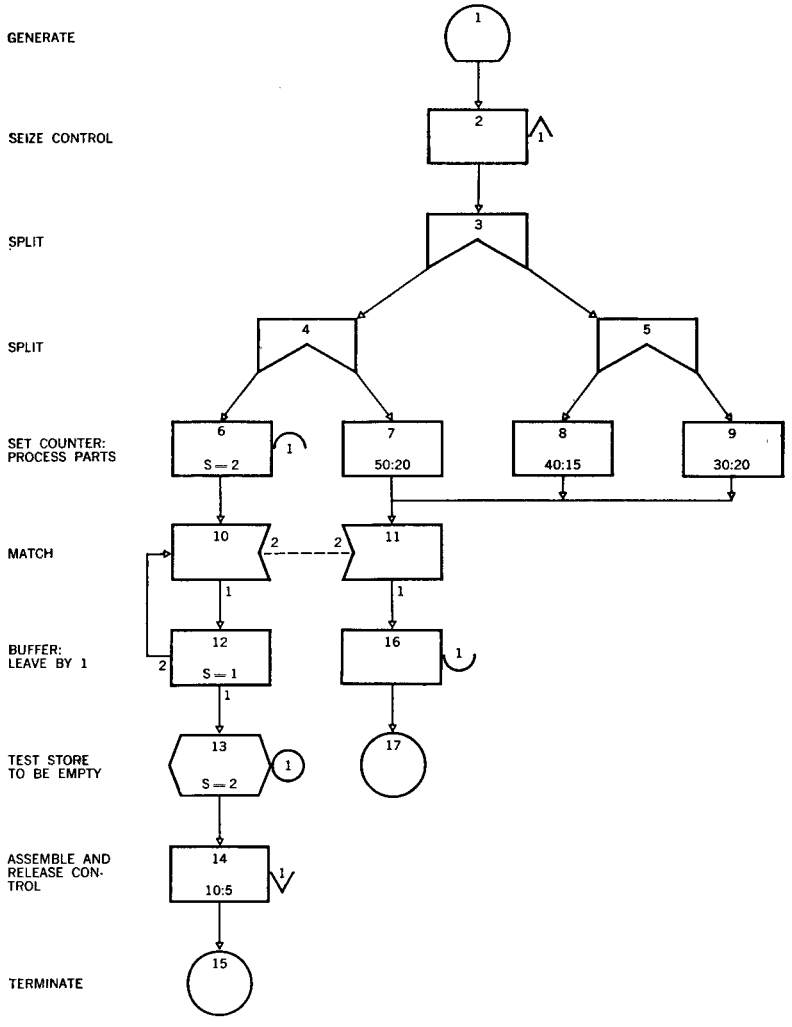
The GATE block is used to test the status of a piece of equipment. The block refers to a facility or store and a transaction may only enter the GATE if the piece of equipment is in one of a number of specified states. The principal conditions that may be tested are whether the facility is in use or not in use and whether a store is full, not full, empty or not empty. In later versions of the program it is also possible to test whether or not a parameter is equal to a particular value.

Used in conjunction with a store where the contents are being decremented by a LEAVE block or with an ASSIGN block that is decrementing a parameter by 1, GATE blocks can be used to make transactions execute a number of loops.

another example

A second example of a model is shown in Figure 13 to illustrate some of the methods by which control over the flow of transactions is exercised. The example represents a simple manufacturing process in which one job at a time is processed. The job requires that three parts be made and these are made concurrently. When

Figure 13 Manufacturing problem



all parts are ready, they are assembled and the next job is started.

Referring to figure 13, the GENERATE block creates a transaction which immediately seizes facility 1. The facility is not released until the job is ready for assembly, thus arranging that the next job is not started until operations on the preceding job have been completed. A network of SPLIT blocks creates four copies of the transaction. Three of these represent the parts to be manufactured and they go to ADVANCE blocks simulating the manufacturing processes. The fourth transaction sets a counter by entering a store of capacity three in a total mode. It then waits at a MATCH block for the processed parts to arrive.

As each part arrives, a MATCH is effected. The transaction representing the part removes the count of 1 from the store and is then destroyed. The control transaction uses a GATE to check whether the store is empty. If not, it returns to wait for another MATCH until finally all parts arrive and the transaction representing the completed parts passes through the GATE to be assembled and release the system for the next job.

Not all features of the program have been described but enough of the concepts have been shown to illustrate the method employed. With a little training and experience the program provides a relatively simple way of constructing and running a simulation quickly and easily. The program is applicable to a substantial set of important problems and thereby allows simulation to play a greater part in systems engineering solutions.

REFERENCES

1. Smith, E. C., Jr., "Simulation in Systems Engineering," IBM Systems Journal, this issue, p. 33.
2. Gordon, G., "A General Purpose Systems Simulation Program," Proc. of the Eastern Joint Computer Conference, Washington, D. C., December 1961. The Macmillan Co.