[redacted]

510.84
Il6r
no 374
Cop 2

Report No. 374

A COMPUTER SIMULATION OF A LARGE SCALE ACADEMIC COMPUTER SYSTEM*

by

Henry Russell Adams

February, 1970

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 374

A COMPUTER SIMULATION OF A LARGE SCALE ACADEMIC COMPUTER SYSTEM*

by

Henry Russell Adams

February, 1970

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois, 1970.

TABLE OF CONTENTS

1. INTRODUCTION

The implementation and operation of a large scale computer system requires that several decisions be made with regard to system configuration and operating techniques. Some of the questions which arise are answered by outside agencies. For example, can we afford another computer? Others are determined by the state of the art, be it programming or hardware. For example, can we get a faster computer or compiler? Still other questions involve the utilization of the system on hand and the "trade-off" between gains and losses when certain options are utilized. For example, should dispatching priority be recalculated during the operation of a job? What length of input job queue search gives the best utilization of the available core?

It is this last type of question which has no fixed answer. As the needs of the computer installation vary, the "best" configuration changes. It is time consuming and difficult, if not impossible, to try all new ideas on the real system because the real system is needed for the standard job stream. There is no gain if you must operate the system at 50 percent capacity for days in order to collect data so that you may improve operation 5 percent.

This paper describes the implementation, construction, and use of a computer model, SIM/360, which can be used in order to make comparisons of techniques and system configurations without degradation of the physical system.

2. SIM/360 MODEL PHILOSOPHY AND IMPLEMENTATION

Modeling is the formation of an abstraction from a real problem. Methods of modeling can be classified on one of two bases. The first, method of solution, consists of (1) analysis, as in linear programming, inventory models, or queueing theory; (2) physical simulation, with wind tunnels, equivalent circuits, or prototypes; and (3) computer simulation, with either discrete or continuous models. Discrete computer simulation models only sense the condition of the model at specified time intervals. Continuous computer simulation models constantly sense the condition of the model and are effectively analog programs. Both of these techniques use Monte Carlo (random number) techniques during operation.

The second basis of classification is the relationship of the model to the actual problem. The major classes of this method of description are (1) iconic models, which are scaled replicas of the physical problem, such as wind tunnels or prototype factories; (2) analog models, which are formed by a functional transformation of the physical problem, such as equivalent circuits of hydraulic problems; and (3) symbolic models, which are approximations of the physical problem with mathematical or logical symbols used to express interrelationships, such as linear programming, queueing theory, and computer simulation.

2.1 SIM/360 Modeling Techniques

SIM/360 is a symbolic, discrete computer simulation, as defined above. As such, many of the decisions within the model are made by evaluation of a pseudo-random number. (Pseudo-random numbers are used so that the sequence of numbers can be duplicated as required.)

A computer simulation of the IBM System/360 is advantageous in several ways, both direct and indirect.

Direct advantages are those such as (1) a first level increase in the efficient use of the computer (that is, the demonstration that a technique previously derived actually will improve system operation if implemented); (2) the pinpointing of system bottlenecks and/or over-supplied points where savings may be made; and (3) the ability to model new additions to the computer system without the cost incurred in the physical acquisition of such additions.

Indirect advantages come about when a new technique is suggested as an analysis is made of the model or as a user of the model gains deeper insight into the operation of the physical system by using the model.

SIM/360 was written in IBM's General Purpose Simulation System (GPSS) in order to take advantage of the capabilities and construction techniques provided by a language which was written specifically for computer simulation. A GPSS program is written in terms which closely parallel the flow chart of a model. For example, if an element _X_ is to be made busy, or seized, the GPSS instruction is SEIZE X. Not only does this simplify programming, but it aids in explanation of the program to others.

Perhaps the most important reason for chosing a simulation-oriented language over other types of languages is that the simulation language automatically calculates, maintains, and records statistical results for every major operation within the model.

2.2 Implementation of SIM/360

The principle objective of the SIM/360 model is to provide statistical output, under selected options, which will provide a realistic comparison of the various system hardware configurations, Operating System configurations, and input job streams.

In order for the output to be as accurate as possible, the input job stream has been made independent of the remainder of the program. Unless specifically altered during the simulation, the job stream is the same under all system tests. Additionally, the option of running under external job streams is provided.

In order to implement the SIM/360 program in GPSS, a minimum time interval of one millisecond was chosen. Within one GPSS model, only one time scale can be used. With the one millisecond minimum time interval, events which take less than this time must be accounted for in one of three ways. (1) They are ignored; (2) their cumulative effect is summed until it reaches one millisecond; or (3) a random choice of zero to one millisecond is allowed. In the SIM/360 model, all methods are used at various points. If the event directly effects the CPU (which is assumed to be the prime resource), method (2) or (3) is used, as in initiator action. In all other cases the delay is ignored, as in the time a channel is busy issuing a seek command.

2.2.1 Basic Model

The SIM/360 model creates "jobs" which enter the job queue, awaiting selection by an initiator for processing. After selection, the job creates I/O actions for the model and uses the CPU. Upon completion, accounting is accomplished, and the job is terminated. A more complete description will be found in the chapter on program operation.

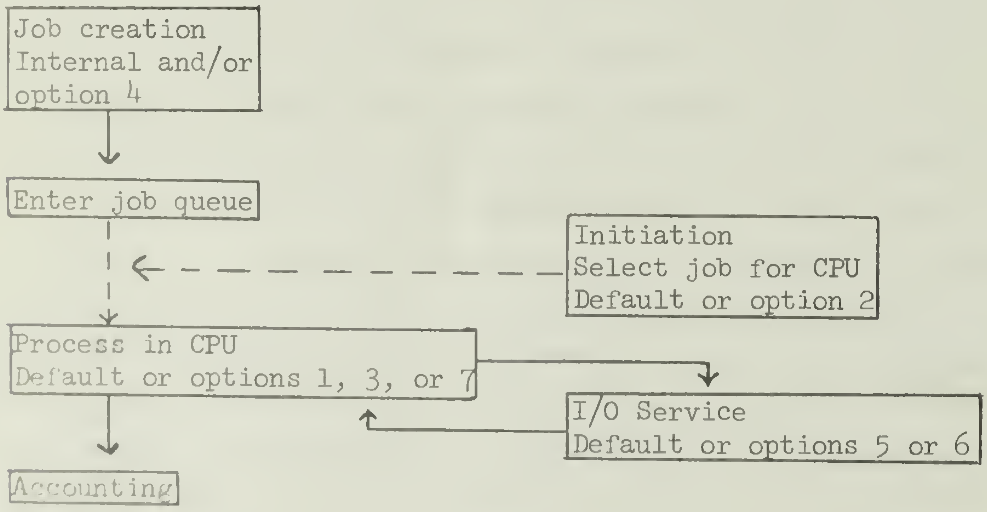

Figure 1. Basic SIM/360 model job flow.

2.2.2 Model Options

When using the SIM/360 model, there are options available by which the system configuration can be altered to satisfy the desired test conditions. Each option is chosen by setting the logic switch associated with it. The following is a brief description of the capabilities provided by each option. A more detailed description may be found in the chapter on program operation, and parameters for each option are found in the chapter on program use.

Option 1 provides the capability to recalculate the dispatching priority of initiated jobs during execution of the jobs.

Option 2 selects the next job to be initiated on the basis of maximum contiguous core block size. Without this option, selection is based on total core free.

Option 3 substitutes an artificial delay for the central processor, without specific I/O requests. This greatly simplifies the model and should be used if the desired model information is external to the CPU, as when simulating various J-Spans on the input job queue.

Option 4 allows the user to construct a specific job stream for the model in order to facilitate accurate job description with individual jobs composed of one or more transactions.

Option 5 simulates the ordering of I/O requests at the channel by job priority instead of by the default first-in, first-out ordering.

Option 6 simulates the ordering of I/O requests at the channel in such a way as to cause minimum seek time.

Option 7 (only used in conjunction with option 1) insures alternate ordering in the CPU of jobs which have the same priority.

3. MODEL CONSTRUCTION AND OPERATION

The model is constructed of eight major segments. These segments are:

1. Model Setup, Definition and Control
2. Job Generation
3. System Job Accounting and Job Queue Control
4. Job Selection and Initiation (including Core Management)
5. Processor Logic with Final Job Accounting
6. Priority Recalculation
7. Input/Output Channel Service
8. Input/Output Devices

The interaction is as shown in Figure 2.

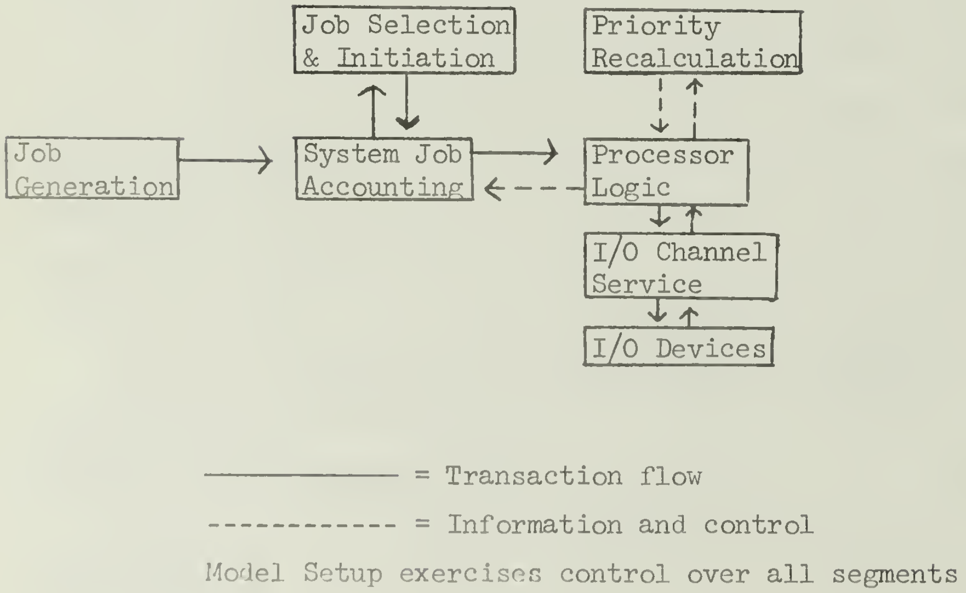

Figure 2. Model segment interaction.

3.1 Model Setup, Definition and Control

The Model Setup, Definition and Control segment exercises overall control of the model operation. It contains no GPSS blocks, but consists entirely of descriptive entities such as global variables and functions, equate statements, and dimension statements. Variables and functions which are used in only one segment are contained in the individual segments. The upper limits on all elements which are of variable numbers are established. These elements include the initiators and the disks. The default values used during program operation are established by INITIAL cards within this segment.

3.2 Job Generation

The Job Generation segment creates the number of jobs indicated in XH$JOBS for use by the remainder of the model. The jobs created may be used as the only job stream input or in conjunction with externally created job streams which are fed into the Job Accounting and Job Queue Control segment. If TST4A is set, no jobs are created.

The transactions are created at a rate set by the value of XH$SPRED, which is interpreted in milliseconds. As they are created, each transaction is assigned an individual job number in parameter 1, and a random core size is assigned to parameter 7. At this point, a check is made to see if the TST3 option has been selected. If it has, no further parameters need be given values, and the transaction branches to the Job Accounting and Job Queue Control segment. If it has not been selected, the parameters associated with the I/O device identity, cylinder number on the disk, and number of I/O requests (parameters 4, 5, and 12) are given values. A random transfer is then made (controlled by FN$JTYP) to create processing time intervals in parameters 8 (overlapped processing) and 9 (unoverlapped processing), which approximate the three types of jobs, namely CPU, MIX, and I/O.

The transaction then checks for option 4 selected. If it has not been selected, each transaction represents one complete job, and its formulation is complete. Therefore, it is sent to the Job Accounting and Job Queue Control segment. If option 4 has been selected, a portion (established by FN$JBILD) of the transactions are split to create multitransaction jobs. The jobs created are then processed in the Job Accounting and Job Queue Control segment. All evaluations are computed with random number 8, which is not used elsewhere in the model. This insures that the same job stream can be created regardless of system alterations.

3.3 Job Accounting and Job Queue Control

The Job Accounting portion of this segment is almost exclusively used by job streams which may have more than one transaction per job, i.e., during option 4 selection. The input transactions are directed by a JOBTAPE card to TEST4 which is the beginning position of the accounting portion. At TEST4, the transaction is checked to see if it is a part of a job which is to be dumped (the user may specify that a certain number of jobs are removed from the front of the input job stream). If the job of this transaction is not listed as one of the jobs already determined to be dumped, a check is made to see if enough jobs have been selected to be dumped. (The first n jobs are dumped, where n equals the value of X$DUMP.) If more jobs need to be dumped, the job identity number (parameter 1) is made a member of the group of job numbers to be dumped so that all following portions of the job are automatically deleted as soon as they arrive.

Once the proper number of jobs have been dumped, new jobs are entered into the system until the number contained in X$IJOBS (number of jobs to be considered) is equalled. The remainder of the jobs are then routed to a dump location.

If a job is to be processed by the system, assignments are made to parameters 2 and 3 (I/O type and control unit). The first transaction of the job enters the Job Queue Control portion and is entered into contention for initiation in the CPU after its core size (parameter 7) is altered to fit the system parameters. This is also the point where the internally generated single transaction jobs enter the flow.

A check is made to see if the number of jobs waiting is less than the length of the queue to be searched for the next job to be initiated (X$JSPAN). If it is less, the transaction then checks for a free initiator and sufficient free core in the CPU. Upon finding both of these, the transaction proceeds to cause initiation of the job in the CPU. If either of the above is not present, the transaction enters the job queue INQUE to wait selection for initiation.

If the number of jobs awaiting initiation is greater than or equal to X$JSPAN, the priority of the new job is compared with the priority of the last job in INQUE. If the new job's priority is greater than that of the other, the new job tries for initiation as above, and the older job is sent to the front of the unscanned portion of the queue. If the new job is not of a higher priority, it is sent to be linked to the unscanned portion of the queue, in order of priority. This insures that high priority jobs are considered ahead of lower priority jobs.

The rest of the transactions of a job are routed dependent on the position of the first transaction, i.e., the status of the new job. If the job has not been initiated, the transactions are immediately linked in a wait queue (WTPRO). If the job has been initiated, a check is made to determine whether all previous transactions of the job have been completed. If not, the transaction is linked to await its turn (in INPRO). Otherwise, the transaction proceeds immediately to be processed.

3.4 Job Selection and Initiation (Including Core Management)

The Job Selection and Initiation segment of the program performs functions of both the support system (ASP, LASP, or HASP) and the Operating System during the operation of the model. A major portion of it is the core management procedure, which is used during program option 2. The procedure monitors the size of the largest block of contiguous core which is free, and it bases job selection on both core size and the job being in INQUE, i.e., within J-Span length of the top of the queue.

The only transactions which flow in this segment are those created within the segment at the initial stages of the program to perform the initiation. When the first transaction enters the segment, it resets various savevalues which deal with core size or act as pointers. This is done to insure that the starting conditions are properly initialized after each CLEAR card, and to allow the user to vary the total core size by adding only a STORAGE definition card of the desired size. After this reinitialization, the single transaction is split to create the number of initiators as indicated by X$INITS, and each initiator is assigned an identity number. Only one initiator at a time is allowed to be searching for a job in order to avoid conflicts. This is accomplished by putting all free initiators except one into a user chain, INITS, and releasing one when the first has completed action.

A free initiator enters contention for the CPU before it begins job selection. The initiator has a priority of 100 which insures that it may interrupt any user job in the CPU. Additionally, the priority of 100 insures that system jobs of higher priority can be run without being preempted. Once the CPU has been preempted, used and returned, the initiator proceeds to check for a job in INQUE which may be initiated. The selection of a job is accomplished when the core size required by a job (parameter 7) is less than or equal to the value of XH$FRECR. If option 2 has been selected, XH$FRECR is equal to the size of the largest block of core which is not being used. If option 2 has not been selected, XH$FRECR equals the remaining space in CORE storage (R$CORE).

If a job is found for initiation, the initator's identity number is saved and the initiator waits for completion of the setup of the job in the CPU. This is signalled by the resetting of logic switch INIT. Upon completion of job setup, the initiator checks for selection of option 2. If the option is not selected, the initiator releases the next free initiator from INITS, and joins the chain of initiators in use, INITU. If option 2 has been selected, the initiator gets the pointer to the location in the table of free core which contains the limits of the largest block of free core. Parameters 3 and 4 are assigned the values of the lower and upper limits of core allotted to the current job of this initiator. The core table location is set to indicate the limits of the block of core which is not needed by this job. It is set to zero if no free core remains from the block.

The initiator then proceeds to find the largest remaining block of free core. This is accomplished by comparing all entries in the table of free core and saving the value of the largest block of free core in XH$FRECR and the table location of this entry in XH$POINT. The initiator then proceeds to release the next initiator and join the other initiators in use in INITU.

If a job is not found for initiation, the initiator links to INITW. The presence of an initiator in this chain is used to alter the flow of new jobs in the Job Accounting and Job Queue Control segment. If a new job will fit into the core, the initiator is released from INITW and proceeds as above.

When a job completes processing it releases its initiator for terminal processing. If option 2 is not selected, the initiator releases an initiator to try to select a job for initiation. If option 2 is selected, the initiator tries to merge the newly freed core with the blocks of free core entered in the free core table. A block can be merged if its lower limit is equal to the upper limit of the freed core as given in parameter 4 of the initiator or if its upper limit is equal to the lower limit of the freed core as given in parameter 3. If merger can be accomplished, the entries in the core table are adjusted to indicate the new limits, and any extra table entries are zeroed. If no merger can be accomplished, a free table level is found by either finding a zeroed entry or by incrementing the pointer to the top level of the table (X$CRTOP). A test is then made to determine if the newly freed block of core is larger than the previous largest block as contained in XH$FRECR. If it is, the new value and new pointer are saved. The initiator then proceeds to unlink the next initiator.

3.5 Processor Logic and Final Job Accounting

When a job is selected for initiation, its first and possibly only transaction is transferred from the Job Accounting and Job Queue Control to the Processor Logic and Final Job Accounting segment by the Job Initiation segment. The transaction waits until the initiator's identity number has been saved. It then obtains this number which is used extensively in this segment as a job identifier since only one job at a time is associated with any one initiator. This number is placed in parameter 6. The transaction next moves one job from the unscanned to the scanned portion of the job queue (if the unscanned portion contains any jobs). This maintains the scanned portion at the proper length and insures the proper sequencing of jobs. The transaction next occupies the required core as indicated in parameter 7 and signals its initiator that it may proceed as soon as the job transaction reaches a stopping point. If option 3 is selected, the transaction branches to an advance block which delays it as directed by the initial values of X$ADVAN and X$SSPRD. The transaction then proceeds to terminal processing. If option 3 is not selected, the dispatching priority of the transaction is multiplied by 10 to expand the range of possible priorities and allow a more accurate recalculation if option 1 has beeen chosen.

As the job is put into contention for the CPU, the entries in row 1 through 7 of the column of the DISP matrix (which acts as a task control block) are initialized. There is one column for each possible initiator. The rows represent:

Row

1. Initial priority of the job
2. Total time the CPU has been used by this job (this is initialized to zero)
3. Absolute GPSS time that job entered core
4. Total time this job has been in a voluntary wait state
5. Core size of this job
6. Number of I/O requests since last priority recalculation
7. Job ID number (parameter 1)

The information in rows 1 through 6 is used primarily during priority recalculation. Row 7 is used to establish the initiator number for transactions of the job which might arrive in the system after the job has been initiated and to restore the job identity number to parameter 1 when a transaction completes its use of the CPU. All other transactions which are part of this job are released from WTPRO and linked to a chain of transactions whose jobs are in process (INPRO).

From this point on, the processing of the first and all subsequent transactions of a job is the same until completion of CPU processing. Each transaction has its own priority set to the final priority of the previously completed transaction. (The first transaction maintains its original priority because of the initialization of DISP.) The transaction now joins a GROUP referenced by the initiator number so that its priority can be changed if option 1 is selected. Next, the time that the CPU is to be used for overlapped (concurrent with processing of I/O requests) and unoverlapped (after completion of I/O request) processing is established. This time, in milliseconds, is calculated by using the fixed values in parameters 8 and 11 as upper limits and evaluating a random number, modulo the upper limit. This allows for limited random changes. Both values are calculated now because the flow of the I/O request is affected if the amount of unoverlapped processing is zero. An I/O request is now created by a SPLIT block, and sent to the I/O service routine.

The transaction now enters contention for the CPU on a priority basis. Upon gaining control of the CPU, the transaction delays in an ADVANCE block for the calculated overlapped processing time. If an interrupt occurs, the time remaining is saved, and the transaction reenters contention. Upon completion of overlapped processing and associated accounting, a test is made for unoverlapped processing. If there is some to be performed, the transaction awaits completion of the I/O request, and then enters contention for the CPU as in overlapped processing. It then loops to recalculate times if parameter 12 (number of I/O requests to be completed) is greater than one. If there is no unoverlapped processing to be performed, the transaction branches to the loop.

When all loops have been completed (all I/O requests processed), the job identity number is restored to parameter 1. If option 4 has not been selected, the transaction, which represents a complete job, proceeds to free its core and initiator, perform accounting entries, and indicate that one job has been completed. If option 4 has been selected, parameter 13 is checked to see if this is the last transaction of a job (parameter 13 = 1). If it is not, the transaction frees the next transaction of the associated job from INPRO. It then checks to see if it is the first transaction of its job. If not, the transaction is destroyed. If it is the first transaction, it must be saved because GPSS requires that the transaction which enters a queue must be the one to depart it. Since the first transaction is the one which is recorded in the accounting queues CPU, INQUE, and a core size queue, it is retained in the holding chain, BIN. When the final transaction of a job is sensed (parameter 13 = 1), it is gated to unlink the first transaction of its job from BIN and send it to terminal processing. If there is no transaction for this job in BIN, the current transaction is also the first transaction (i.e., this is a one transaction job), and the transaction is sent to terminal processing.

3.6 Priority Recalculation

The Priority Recalculation segment is used only when option 1 has been selected. If it is not selected, the single transaction which is to loop in this segment is destroyed. No other transactions ever enter this segment. If option 1 is selected, the lapse time between priority recalculations is the value of X$RECLC which has a default value of 5000 milliseconds. At proper intervals, the transaction preempts the CPU and recalculates the priority of all jobs in core. The recalculation is accomplished on the basis of the variable DISPZ, which may be changed by the user. Additionally, the count of I/O requests is reset to zero so that the count is the total number of I/O requests between priority recalculations.

If option 7 is also selected, the recalculation is accomplished in reverse order during alternate intervals. This insures that if two jobs vying for the CPU have the same priority assigned, their ordering for access to the CPU will alternate upon successive priority recalculations.

3.7 Input/Output Channel Service

As the I/O request is created in the Processor Logic segment, it is entered into the Input/Output Channel Service segment which controls the order of processing of the requests to the devices. As the request enters the segment, a transfer is made based on I/O type (parameter 2). The current model has two types implemented. One type is a drum access request (40 percent of all requests) and the other type is a disk access request with a variable cylinder required (60 percent of all requests).

All drum requests are transferred to selector channel one for processing to the drum. Since the drum is the only device attached to this channel, it is modeled as an integral part of the channel. As the request enters channel one, it is chained in a first-in, first-out order if the drum is processing a prior request. When a request completes drum processing, it unlinks a waiting request and proceeds to interrupt processing.

Disk access requests are transferred to the dual channel processing portion since disks may be accessed from either channel two or channel three. As the request enters, tests are made to see if either option 6 (request processing in order of minimum seek time) or option 5 (request processing in order of job priority) is selected. If neither option is selected, the requests are chained in a first-in, first-out order if the required device is busy. If option 6 is chosen, the requests are linked to the device chain in order of cylinder number if the device is busy, with zero the highest priority. If option 5 is chosen and the device is busy, the requests are linked to the device chain by job priority.

If no prior requests are using the device, the request proceeds to vie for a channel. If no prior requests are awaiting a channel and a channel is free, the request identifies itself with the free channel and proceeds to the device simulation. If other requests are awaiting a channel or neither channel is free, the request is linked into a waiting chain, DUAL, to await selection by the channel routine. The channel routine contains two transactions (one identified with each channel) which are gated for processing, dependent upon their respective channel's status. When the channel is not in use, the channel transaction unlinks as many waiting requests as possible. Any request which is not an end-of-seek signal from a device is assigned to the channel associated with the active transaction. (Requests which have completed seeking have already been identified with a channel.) The request is then sent to the device simulation. If no requests can be released, the channel transaction is routed to a wait chain (CHFRE).

As a seek-type device completes its seek, the I/O request tries to seize the channel it is assigned to. If the channel is busy, the request is linked to await selection.

When an I/O request has completed processing, it releases a request awaiting the newly freed device, if one exists. If option 6 is selected, the request released is the first one with a cylinder number greater than or equal to the current head position. If none exist, then the last request in the chain is released. The released transaction then has the next lower cylinder number. The completed transaction then proceeds to interrupt processing.

For interrupt processing a test is made to determine if the job which issued the request is to perform any unoverlapped processing, i.e., parameter 9 $\neq$ 0. If parameter 9 equals zero, the transaction is immediately terminated since the issuing job is not awaiting an indication of the request's completion. If parameter 9 is not equal to zero, the issuing job will enter a wait state in the processor logic segment by entering a MATCH block. If the issuing job is already waiting, the match is completed, and the I/O request is destroyed. If the issuing job has not yet reached the wait state, the I/O request waits in the MATCH block.

3.8 Input/Output Devices

The drum simulation is discussed in the Input/Output Channel Service segment.

The disk is a seek type device. That is, it is a device which may have to perform physical movement of some kind in order to complete commands issued to it. As such, it may receive and implement its seek commands without using the channel for more than a few instruction cycles. Since the time required by the channel to issue the seek command is considerably below the one millisecond time interval of this model, it is ignored. The request, however, must be routed through the channel as described in the previous segment description. The disk simulation is implemented in such a way that it is reentrant. All disks are controlled by the same program portion, and all references to a specific disk number are indirect and dependent on the disk identity established in parameter 4 of the request transaction. Zero to sixteen disks may be modeled, and the number of the last cylinder referred to on each disk is kept in an associated savevalue location.

As an I/O request transaction enters the disk simulation portion it seizes the disk identified by parameter 4. It then retrieves the previous cylinder number (head position) for its disk and calculates a seek delay which is dependent on the amount of head movement required to reach the new cylinder as indicated in parameter 5 of the request transaction. After the delay, the new head position is saved and the transaction is transferred to the Input/Output Channel Service segment to indicate that the seek phase is completed.

When the transaction returns to this segment, it seizes the channel and is delayed a random access time for reading or writing. The transaction then releases the channel and device and returns to the Input/Output Service segment.

4. SIM/360 USAGE

When using SIM/360 there are two principle ways to modify the model. The first is to change the system parameters to alter operating characteristics, and the second is to request system options which alter the model's configuration. These changes in SIM/360 can be used singly or in combination to provide a representative model of various real system configurations. The standard GPSS output is used.

4.1 System Parameters

Figure 3 contains a list of parameters which may be altered. If no alteration is made for a parameter, the default value is assumed.

4.2 System Options

The options or tests provided to the basic model allow a great flexibility. Seven options are provided. In order to use any of the options, the user must set the associated test switch. For example, to use option 2, set logic switch TST2.

Option 1 recalculates the dispatching priority of initiated jobs at an interval determined by the value of X$RECLC. Priorities are established on the basis of the variable DISPZ, which may be coded by the user, or the default variable:

DISPZ = 100-100 * (CPU use time)/ (Time in core). Data available for recalculation include:

Initial job priority	DISP(1,P2)
Total CPU use time of job	DISP(2,P2)
Absolute time entered core	DISP(3,P2)
Total time in wait state	DISP(4,P2)
Core size of job	DISP(5,P2)
Number of I/O requests since prior recalculation	DISP(6,P2)

Parm	Type	Default	
MINCR	X[a]	60K	Minimum core assigned a job
DFULT	X	115K	Default core assigned to a job
INITS	X	4	Number of initiators in system; maximum of 5
JSPAN	X	50	Length of job queue to be searched for next job to be initiated
SPRED	XH[b]	4000[c]	Fixed interval between the creation of new jobs
JOBS	XH	102	Total number of jobs to be created by SIM/360; JOBS=0 implies unlimited job generation because of a GPSS restriction; setting TST4A destroys all internally created jobs
CORE	STORAGE	466	Available core
RECLC	X	5000[c]	Used with option 1 to specify the interval between the recomputation of dispatching priorities
ADVAN	X	60000[c]	With option 3, establishes the mean delay of jobs
SSPRD	X	6000[c]	With option 3, establishes spread of job delay
IJOBS	X	102	With option 4, total number of externally and internally created jobs to be considered by the system
SPACE	X	1000[c]	With option 4, arbitrary delay between formation of successive transactions of an internally created multitransaction job
DUMP	X	0	With option 4, number of externally created jobs which are to be dumped

[a] fullword savevalue

[b] halfword savevalue

[c] time in milliseconds

Figure 3. Alterable parameters.

Option 2 selects the next job to be done on the basis of the largest block of contiguous core available, instead of the total amount of core free, as in the basic program. Because of the accounting that must be done, this slows the model and should only be used when needed.

Option 3 arbitrarily delays jobs in the CPU instead of simulating the action of the CPU. This is much faster running, and can well be used in conjunction with option 2 for core utilization checks.

Option 4 provides the capability for the user to create his own job stream to run with the model, with or without the internally generated job streams. The input job streams can be composed of single jobs represented by single transactions or it may be constructed of series of transactions which represent the various configurations of the jobs. In order to use the option, the user must assign the parameters the same meaning as the SIM/360 program does (Figure 4), and the input transactions must be in the format of a GPSS Jobtape referenced to block TEST4. If no internally generated jobs are desired, XH$JOBS should be initialized to 1 and logic switch TST4A set. This is done to satisfy GPSS requirements.

Option 5 orders the I/O requests by job priority instead of the default first-in, first-out method.

Option 6 selects the I/O request for processing in order of minimum seek time on the device.

Option 7, used in conjunction with option 1, insures equal access to the CPU by all jobs of equal dispatching priority by altering the order of the transactions within the GPSS priority chains.

Additionally, new variables and functions may be defined to replace any of those given. In this manner, the model can be made to approximate various job streams.

Parameter	Use
1	job ID number. This must be the same for all transactions which are part of the same job. Each job must have a different different number.
2-3	assigned by SIM/360 as a function of the device type
4	device identifier. Sixteen disks are available. Their identity numbers range from 21 to 36.
5	cylinder number on disk. May range from zero to 199.
6	assigned by SIM/360. Indicates which initiator is assigned to each job when it enters the CPU.
7	core size. This may range over any number since the model alters it to fit the system parameters.
8	upper limit on the number of milliseconds the transaction may be in overlapped processing.
9-10	used by model to indicate specific CPU times for each loop
11	upper limit on the time spent in unoverlapped processing for each loop
12	number of I/O requests (loops in CPU). Number must be equal to or greater than one.
13	indicates end of job if equal to one.

Figure 4. Job parameters.

4.3 SIM/360 Output

The standard GPSS printout is used in the model. Figures 5 and 6 give a description of what each output element represents. All times are represented in milliseconds.

The SAVEVALUES are used either to establish system parameters or for internal communication by the model segments. Only X$CLOCK provides information to the user. Its value is the absolute GPSS time that the last externally generated job was dumped. This allows a more accurate determination of the elapsed time of a run.

Matrix DISP and all GROUPS are used internally by the model.

User Chain	Entries
INPRO	second and subsequent transactions of multitransaction jobs in CPU (option 4)
CHAN2	I/O requests which need to seize channel 2
CHAN3	I/O requests which need to seize channel 3
WTPRO	second and subsequent transactions of multitransaction jobs not yet initiated (option 4)
INQUE	scanned portion of the input queue; maximum length is established by X$JSPAN
INITW	presence of a transaction in this chain indicates that an initiator is free for an incoming job
INWAT	unscanned portion of the input queue
INITS	holding chain for initiators to insure that only one initiator at a time is trying to start a job
INITU	initiators in use by jobs in CPU
BIN	first transaction of multitransaction jobs in CPU (option 4)
DUAL	I/O requests which need access to either channel 2 or 3
CHFRE	holding chain for channel control transactions when no I/O requests require the channel
DRUM	chain of I/O requests awaiting access to the drum
DISKS (chains 21-36)	I/O requests awaiting access to the individual disks

Facility	Entries
CPU	central processor element
CHAN2 & CHAN3	selector channels
DRUM	2301 drum
DISKS (chains 21-36)	2314 disks

Figure 5. Standard GPSS printout--user chains and facilities.

Storage	Entries
CORE	accessible free core of the IBM/360

Queue	Entries
INQUE	entered upon initial entry to job queue and departed upon termination of job
CPU	entered into upon initiation and departed on termination of the job
OLAP	time spent trying to accomplish overlapped processing, i.e., access to CPU and processing time
OLAP2	time spent trying to access CPU for overlapped processing
WAITC	entered into upon entering a voluntary wait status, awaiting completion of I/O request; departed when I/O request is completed
SMALJ	same as INQUE but only for jobs with core requirement less than or equal to 116K
MEDIJ	same as INQUE but only for jobs with core requirements between 117K and 232K
LRGJ	same as INQUE but only for jobs with core requirements between 233K and 348K
XLRGJ	same as INQUE but only for jobs with core requirements between 349K and 466K

Figure 6. Standard GPSS printout--storages and queues.

5. MODEL TEST RESULTS

In order to establish the operating characteristics of the SIM/360 model, two different runs of one hundred jobs each were made with the basic model and with each of the various model options selected. The default values were used for all runs. The statistics of the basic model were used as the basis for all comparisons. For all tests, the internally generated job stream was composed of jobs which had the following average distribution of core size required:

20 percent had a core size of 60K bytes,
50 percent had a core size of 115K bytes,
20 percent had a core size of 230K bytes,
6 percent had a core size of 345K bytes, and
4 percent had a core size of 460K bytes.

These ratios were established by the function CORSZ. Unless otherwise noted, the job stream used was identical for all runs of a series of tests so that the effect of system changes would be the only variable in the model operation. The comparisons were made to determine the specific effect of the individual options.
options.

When option 1 (dispatch priority recalculation) was selected, the simulated elapsed time required to run 100 jobs was decreased by approximately five percent. No alteration was noted in the ratio of delay times in the various size job queues. This indicates that the increased efficiency of the CPU caused by priority recalculation had an approximately equal effect on all job sizes.

Option 2, (contiguous core management) when selected, had less than one percent effect on the average delay time on all jobs as recorded in INQUE and on the time required to accomplish 100 jobs, but it caused the ratio of time delays for various core size requirement jobs to change markedly. A significant increase in the delay time of medium and large jobs was noted. This is an expected change because when option 2 is not selected, two separated blocks of 115K bytes of core would act the same as one contiguous block of 230K bytes.

Therefore, a job which needed 230K bytes of core could be initiated. With option 2 selected, the largest block of free core would be 115K bytes, and the job requiring 230K bytes would be further delayed.

The change caused by selection of option 2 is only significant if information is desired on the relative delay times of various core size jobs. Since option 2 requires increased accounting by the model, it should only be used when needed.

The selection of option 3 (simplified CPU representation) causes a different job stream to be created because all of the parameters of a job transaction are not assigned values, and the random number sequence which is used to assign core size is different during tests with this option. In addition, the default values which determine the delay of each job in the CPU are set higher than the average time required in the CPU during other option runs. This increase was established so that the job queue would increase more rapidly during this option, and allow more flexibility in the testing of various J-Span lengths or other characteristics. The increase in simulated run time was 200 percent, and because of the increased average job queue length, the average turn around time increased by 300 percent. The computer time required to run the model with option 3 selected was decreased by 60 percent.

Option 4 (multiple transaction job creation) also causes a different job stream to be created because each job requires extra evaluations of the random number sequence as it loops in the job creation segment, in order to form the transactions which compose its various portions. Even though entirely different job streams were developed, there was only a four percent decrease in the simulated elapsed time and less than a two percent change in the average turn around time as computed in INQUE. This indicates that the type of job streams developed with and without option 4 selected are similar. The advantage of option 4, besides permitting externally created job streams to be used, is that by varying the ratio of the various types of job segments (I/O, MIX, CPU), the jobs created more nearly approximate the actual jobs of a system. It was also found that dynamic recalculation provides a better improvement with jobs which change type during their operation.

When option 6 (I/O ordering by minimum seek time) was selected, a three percent decrease in elapsed time was noted.

Options 5 (I/O ordering by job priority) and 7 (reversing of order of jobs within one priority class), reduce to the basic model when option 1 is not selected, and no new information was obtained.

6. CONCLUSIONS

The following conclusions and suggestions were formulated from the statistics created by the operation of the SIM/360 model. Unless otherwise noted, the default values as previously listed were used.

6.1 JSPAN Length Tests

In order to determine the effect of changes in the J-Span length on the system efficiency, six job streams of 500 jobs each were run for each J-Span length from 50 jobs to 140 jobs (incremented by 5). Additional runs were made with J-Span lengths from 70 to 100 jobs in order to verify the following conclusions. Optiions 2 and 3 were selected during all of these runs. Option 2 was selected in order to provide the most accurate simulation of core usage and management, and so that the various size jobs were more accurately delayed. Option 3 was selected because it allowed a greater number of jobs to be run at one time since the model operates much faster with this option.

The results of these runs indicate that average turn around time is decreased by 6.5 percent as the J-Span length is increased from 50 to 95 jobs. The increase of the J-Span to 140 jobs yields only 1.5 percent further improvement. As the J-Span is increased, the difference in average delay time of small jobs (60K or 115K) and that of other jobs increases. With a J-Span of 50 jobs, the average delay of medium size (230K) jobs is 1.23 times that of a small job. When the J-Span is 95 jobs the ratio is 1.5. With a J-Span of 140 jobs the ratio is 1.65. The ratio between small and large job delays ranges from 2 at 50 jobs to 3 at 140 jobs.

With the above data, a management decision could be made as to the desired "trade-off" between system efficiency, as measured by overall average turn around time, and the ratio of various size jobs' turn around time. From the test results obtained, a J-Span of 95 jobs is a "good" choice since it gives 80 percent of the maximum increase in system efficiency and 60 percent of the increase in ratio of turn around times.

6.2 Comparison of Implicit and Dynamically Calculated Dispatching Priorities

In order to evaluate the gains accrued by implementing a dynamic priority recalculation scheme, various formulas for the recalculation were implemented, and the results were compared with the results obtained with no recalculation. Option 4 was used during all tests in order to provide a more realistic and varied job stream. Unless otherwise noted, the default simulated time of 5 seconds was used as the recalculation interval.

The completion of the 100 job runs without recalculation required an average of 708 seconds (simulated elapsed time), and this was used as the base for all of the following comparisons. All ratios are in reference to the average value of these base runs. The same job streams were used for each set of tests.

When the default formula was used with option 1, the average simulation time to complete a run was decreased by seven percent. The average time spent waiting in core for the completion of an I/O request was decreased by 10 percent.

The dispatching priorities of the jobs always tended toward the maximum allowable (100) during these runs. Because of this, a set of runs was made with the following formaula:

DISPZ = 100 - 200*(CPU use time)/(time in core).

This formula is the same as the default, except that the priority decreasing factor is twice as large. This spreads out the dispatch priorities. These runs produced a decrease of 10 percent in the average simulated elapsed time.

Further tests using this formula and decreasing the recalculation interval indicated no further improvement in this figure, and it actually caused a degradation when the interval was decreased to 2.5 seconds.

Another series of runs was made using the formula:

DISPZ = (number of I/O requests between recalculation)/2.

This formula assigns the highest priority to the job which issued the most I/O requests during the previous recalculation interval. This test indicated only a two percent improvement in the average elapsed time of the model.

From the above results, the largest improvement in system operation is accrued when the dispatching priority is recalculated every five seconds according to the formula:

DISPZ = 100 - 200*(CPU use time)/(time in core).

6.3 Comparison of I/O Request Ordering Techniques

Tests were run in order to compare the average simulated elapsed time of the model when the various possible I/O request ordering techniques were used. Option 1 with the default recalculation formula and option 4 were used for all tests.

The default ordering method is first-in, first-out. The average elapsed time for 100 jobs using the default ordering technique was 660 seconds. When the I/O requests were ordered by job priority (option 5), the average elapsed time increased by four percent. The increase was caused by an increase in the average time spent in core by jobs awaiting completion of an I/O request.

Option 6 (I/O request ordering by minimum seek time) was tested with and without dynamic priority recalculation (option 1). In both cases, there was a three percent improvement in the simulated elapsed time. The improvement came about by decreasing the average time required to complete an I/O request, and it was independent of the job dispatching priority formulation.

6.4 Summary

The model test results obtained indicate that the Operating System configuration which provides for the most efficient use of the computer will have a J-Span of 95 jobs or more (depending on management decision on the ratio of comparative turn around times of various job sizes). It will have a procedure to dynamically recalculate the dispatching priority of jobs in the CPU every five seconds according to the formula:

DISPZ = 100 - 200*(CPU use time)/(time in core), and the I/O requests should be ordered by minimum device seek time.

The SIM/360 model can be used to determine the effect of future changes to the IBM Operating System/360.

BIBLIOGRAPHY

Corbet, Larry L. and Jones, D. J. OS/360 Advanced Multiprogramming Performance Analysis Via Simulation, IBM Technical Information Exchange, Z77-6231. IBM Corporation, Technical Publications Department, White Plains, New York, 1966.

Graham, John H. Modeling and Computer Simulation, IBM Technical Information Exchange, Z77-8057. IBM Corporation, Technical Publication Department, White Plains, New York, 1968.

Hillier, F. S. and Lieberman, G. J. Introduction to Operations Research, Holden-Day, Inc., San Francisco, September, 1968.

Marshal, B. S. "Dynamic Calculation of Dispatching Priorities Under OS/360 MVT," Datamation, Volume 15, Number 8, Pages 93-97, August, 1969.

Naylor, T. H. Computer Simulation Techniques, John Wiley and Sons, Inc., New York and London, 1966.