Design, thru simulation, of a multiple-access information system*

by LOUIS R. GLINKA,
R. MICHAEL BRUSH** and ALAN J. UNGAR
IBM Corporation,
Gaithersburg, Maryland

INTRODUCTION

The popular concept of a multiple-access processing system as a common facility for many users (at remote terminals) with varied processing requirements, has over the past few years received considerable attention.[1-3] The research for proper hardware and software design solutions has been primarily directed at installations which would support a large and diverse group of users with varied and sometimes complex applications, thus implying the requirement for a powerful computer and large memory storage.[4,5] The usefulness of these time-sharing systems stems from their capability to provide computing power to any user when, where and in the amount needed.

Within this broad classification, there is a type of system which can be characterized by its dedication to support a limited number of applications, and a relatively small number of users whose characteristics are considerably more uniform. This special purpose type of time-sharing system entails the simultaneous usage of multiple terminals on-line with a central information processor by a homogeneous group of system users performing the functions of information storage and retrieval. One example of such an installation is in the military, where non-data processing oriented operations personnel operate and maintain a central depository of information in a field-based tactical intelligence operation.

The characteristics of this system, which is considered to be typical of its kind, imply certain design requirements. The task of formulating an appropriate system design in response to these requirements is greatly facilitated by a modeling and simulation process. The broad objectives of this activity were to represent in a model the characteristics of, and potential designs for, such an information handling system, and to evaluate the system trade-offs between data processing equipment (as characterized by certain parameters), program design and system performance.

This paper discusses the previously cited military system in terms of the job functions to be performed, the man-machine interaction, and the implied requirements of an appropriate hardware and software design It further defines the measures of system performance, the representation of the system, and the presentation of quantitative results. Finally certain conclusions are drawn which center on the effect of the central processor's main memory size and hence the processing strategies as constrained by available memory capacity, on system performance.

The system

The operational role of this system is to operate and maintain a central depository of information in support of an adjoining environment through the maintenance, retrieval and dissemination of data.

Data concerning a broad spectrum of subjects is received and handled in the system in varied forms and from many sources. In general, the system operators (i.e., users) categorize this data by subject and add it to existing collections of similar information. There is a requirement of the system to allow a large number of different data sets; i.e., dissimilar in structure and content.

The integrity and currency of the system data base is maintained by the users through either the adding of data, or the deleting or modifying of existing data. These update actions may affect one or more entries in one or more data sets. In addition to creating and

*This work was supported in part by the Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson AFB, under Contract No. F33(657)-67-C-0158.

**Currently with Electronic Data Systems Federal Corporation, Washington, D.C.

maintaining the data base, the operators use this collection of stored information by effecting retrievals of selected data. These usually are in the form of either highly selective (i.e., discriminating) inquiries, or requests for summarizations of large amounts of stored data. In any case, the user generally requires the output presentation in a particular format on a selected device.

Thus the system is dedicated to supporting a constrained set of applications (viz., information storage and retrieval functions) while on the other hand requiring considerable latitude and generality in the type and amount of data to be handled, as well as in the flexibility of data retrievals and forms of output data presentation. This implies the requirement for a generalized data handling program solution which is independent of the actual structure and contents of any data set in the system.

The system users interact directly with the system rather than through some data processing service group, and usually are operations personnel who are skilled in areas other than data processing. Thus there is the requirement that the operator training requirements and the likelihood of operator errors in system use be minimized. A hypothesized design responsive to both requirements is a sequential process of problem input whereby each operator action results in processor acknowledgment of the action and a processor request to the operator for additional inputs, or processor output of the requested data. This bilateral man-machine dialogue continues through a sequence of information exchanges upon which decisions are made and available options are exercised until the desired end result is achieved. The system user is in effect compiling a job statement by stepping through a pre-determined network of problem-definition nodes. This scheme entails a hierarchical structure in which the user chooses a problem definition path, as a function of job requirements and personal experience or proficiency in system usage, and furthermore is able to discard all non-appropriate steps. At each node (or step) as depicted in Figure 1, the user responds with an appropriate information input according to an instruction cue presented by the processor. The problem definition network has the added attribute of facilitating subsequent modifications and additions and can be arbitrarily large and complex.

The simultaneous usage of multiple terminals online with the system processor constitutes an environment in which each terminal's request for processing is competing with all other requests for use of the processor and other system resources. Furthermore, there is a requirement for rapid access to and response from the system such that at each terminal, the system will appear to be completely captivated by that user. Thus a system supervisor program is required which will schedule the work to be done in the system and the use of system resources according to some priority scheme.

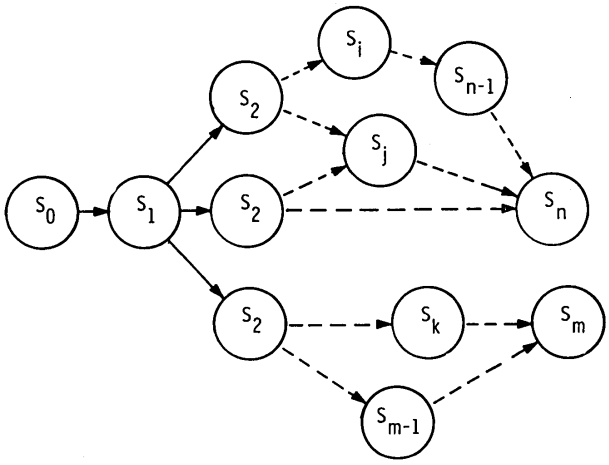

Figure 1 – Job definition network

Since only a small fraction of the total information (including the data base) contained in the system can reside permanently in core storage there is a requirement that the system control program provide for the overlapping of input/output operations with other processing, within the constraints of available core space.

In the total time period of turning around a job, the interaction between man and machine varies such that during the job definition phase the user is active, while during the job processing phase the user is passive or in the background. Similarly the processing requirements vary during this total time period, in that the servicing of job definition steps through a pre-established network entails considerably less processing than does the execution of the compiled job statement. Thus there is the requirement for multiprogramming support within the system control program.

Due to the restricted nature of the job types (i.e., data storage and retrieval) to be processed, and the availability of a generalized data handling program to process any of these jobs, there exists then a certain amount of predictability and commonality in the system operation similar to that normally associated with batch processing applications. In this regard, the processing of a job once initiated to completion appears *a priori* to offer advantages in terms of system thruput performance. This can be rationalized on the basis of the reduction in overhead time achieved

with overlapped program swaps and stems from the predictability of processing requirements. A further extension of this theme towards batch processing operations is based on the commonality of processing requirements among such similar jobs and thus a consideration of the merits in grouping jobs for execution through the processing sequence. This type of processing strategy is termed here as "multi-job" processing.

The servicing of job definition steps through the system control program requires application programs which maintain the status (or position in the job definition sequence) of each terminal, interpret (and/or translate) user inputs, and determine and provide for the subsequent instruction (cue) to the operator.

The previously stated requirement for a generalized data handling program implies the existence in the system of a program to cover the possible eventualities of job statement inputs. However partitioned into modules, this application (or problem) program is, in effect, a group of common routines. In order to provide the required flexibility and adaptability in their intended usage these routines must be serially reusable (if not re-entrant) and relocatable; i.e., following the execution of a routine relative to one job, the same (main memory) copy of that routine can be used again in regard to another job, and furthermore the routine is capable of execution in any physical part of the main memory. Thus there is a requirement that each active job in process have dedicated work space.

Figure 2 depicts a grouping of data processing equipment units which is considered representative of the hardware requirements of systems of this type. The use of specific parameter values to characterize a particular equipment configuration for the cited military system is treated in a subsequent section.

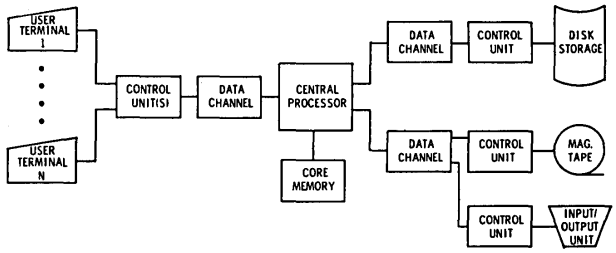

Figure 2 – Equipment configuration

Performance measures

From the system user's point of view, there are at least two important quantitative measures of system performance; viz., job turn-around-time and job processing time.

The *average job turn-around-time* is defined as,

$$\frac{\sum_{i=1}^{N_c}(T_{j\ out} - T_{j\ start})_i}{N_c}$$

and the *average job processing time* is defined as,

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

where,

- $T_{j\ start}$ is the time at which the job definition sequence began.
- $T_{j\ in}$ is the time at which the last job definition step for the job was completed.
- $T_{j\ out}$ is the time at which the output of the job results to the terminal CRT display is completed.
- N_c is the total number of jobs processed to completion during the run.

The job turn-around-time encompasses the total involvement time of the user during the course of accomplishing one job. The job processing time is the time when the user is, in effect, passive and awaiting final output results.

There is at least one additional measure of system performance which is relevant to the system user and is frequently discussed in regard to the broad class of multiple-access systems; i.e., the average system response time per user action (or input). However for this special purpose type of system, where the processing requirements during the job definition phase and the job processing phase differ significantly, the cumulative result would be biased and hence was not considered a meaningful measure of comparative performance.

In addition to the above measures of system performance, the behavior of internal system queues and equipment component utilization statistics are common indicators for analyzing the operation of a system and evaluating the performance of a design solution.

The model

The system model entails certain operational assumptions which reflect the envisioned operating environment and a postulated software and hardware design. Since the modeling activity constituted just

part of an over-all effort directed at establishing an appropriate design, then by necessity only gross estimates of the functions and dimensions of an envisioned design could be made at this time.

In principle, the desirability of using parameters rather than constants in a modeling activity is well recognized. The use of constants in this specific activity, however, can be rationalized on the basis of its limited initial objectives.

Equipment

The operating characteristics of the equipment components in the model were taken from descriptions of available hardware. While the modeled units are not all of the same manufacturer, they are compatible in the modeled configuration. The following equipment was modeled:

- The central processing unit (cpu) has a 32 bit word length and a 2.5 μ second memory cycle. The processing speed was represented by an instruction execution rate of 100K instruction per second. The core memory options available with this cpu ranged from 16K to 64K words, in increments of 8K words.
- A random access disk storage device, combining the control and four drives in one unit, was configured on a separate high speed data channel. The unit has a total capacity of 200 million bits and a data transfer rate of 720K bits per second. The modeled unit reflects the capability to seek and read (or write) simultaneously on separate drives. The following additional operating characteristics were incorporated into the model: 20 ms average rotational delay time, and mechanical access times distributed over a range of 59 to 272 milliseconds.
- A magnetic tape unit was configured on a separate data channel. The data transfer rate was 16.9K six-bit characters per second and the start/stop time of the unit was modeled at 6 milliseconds.
- Eight (8) operator terminals with self-contained buffers, an output-only plotter, and a printer were configured on one data channel. The operator terminals were assumed to include a CRT console with a 900 character display image and an alphanumeric keyboard. The X-Y plotter had a 150 eight-bit characters per second data acceptance rate and a 200 millisecond start/stop time. The printer was modeled with a 300 eight-bit characters per second data acceptance rate and a 6 millisecond start/stop time.

Data base

The system was assumed to contain 16 data sets, ranging in size from 100 records to 13,900 records. Each data record had a fixed length of 400 characters. The total data base assumed was approximately 15.5 million eight-bit characters.

Data set descriptor tables, which specify in detail the structure and contents of each data set and permit interaction between the data and a generalized data handling program, were included in the modeled design. A primary index and additional indexes, as a function of data set size, were included for each data set.

System jobs

All jobs were entered into the system via the user terminals. The selection of a job at each terminal was done on a random basis. The following types and their percentages of occurrence were modeled.

- single record retrieval (44%)—which resulted in an output to the terminal CRT display.
- multi-record retrieval (22%)—which required the sequential processing of n data records, where n was arbitrarily set at 8, and resulted in an output to the CRT display, printer and X-Y plotter.
- multi-data set retrieval (12%)—which required the sequential retrieval of m data records, one from each of m data sets, where m was arbitrarily set at 2, and resulted in an output to the CRT display and printer.
- single-record updata (22_)—which in addition to the data maintenance processing, resulted in a logging of the record on magnetic tape and a confirmation-of-update message output at the CRT display

All outputs to the CRT display, printer and magnetic tape were assumed to be 900 characters. Output to the plotter was 400 characters.

For single record retrievals and updates, data sets were selected with equal probability from the 16 available sets. For multi-record retrievals the data set was selected with equal probability from the seven largest sets. The identical procedure was used for the selection of the first set in a multi-data set retrieval. For the latter, the second data set was randomly selected from the nine remaining sets. The number of data set indexes to be processed in each job was a function of the job type and the size of the associated data set; in general, the number of indexes processed ranged from two to five.

Man-machine procedure

The formulation of each job statement was achieved through a sequence of man-machine data exchanges. Each job would be defined by an operator in s distinct steps; based on the procedural complexity of the job mix modeled, a constant value of $s = 10$ was considered to be representative of the average job. At each step, the user viewed a pre-recorded procedural message sent to him by the processor relative to the data set(s) of interest, he decided what input was required, and made the input via either terminal control key or input keyboard action. For each step in the job definition sequence the operator response time was assumed to be $t = 4$ seconds. On completion of the last step, input was complete, the terminal was locked-out from further user actions, the input job entered the job queue and was processed in terms of the requested retrieval or update of data.

To provide the maximum load on the modeled system each operator initiated the job definition sequence for a subsequent job immediately upon receiving output of the prior job.

Control program

The system control program was assumed to provide the following functions which were reflected in the model.

- Work scheduling in the system queues such that entries in the job queue were serviced on a first-in first-out basis.
- Interrupt control and handling in the following order: disk, magnetic tape, user terminals, plotter and printer, supervisory functions, application program; such that, an interrupt on I/O-complete would be honored during the execution of an application program.
- Program loader capable of relocating programs during transfer from secondary storage.
- Input/output control which maintains the queue of I/O requests and provides for overlapping of I/O operations with other processing, within the constraints of designated available space.
- Operator-terminal control which sequences the interpretive programs required to validate the operator's job definition input and establish the appropriate subsequent response.
- Multi-programming support of the servicing of job definition steps with the processing of defined jobs. A higher processing priority was assigned to the former so as to minimize the job definition sequence time. Note that in the job definition sequence, subsequent user inputs in compiling a job statement are by definition based on prior outputs to the user.

  Multi-job processing support which establishes the group of defined jobs to be batch processed through the required processing sequence. The size and contents of the group was dependent on the designated available work space. In the model, multi-job processing was defined such that, the selection of the Q oldest jobs in the job queue, to be processed (program by program) to completion in a batch, was performed so as to,

$$\text{minimize } [S - \sum_{i=1}^{Q} J_i]$$

  where S is the work space available for processing these jobs, and J is the work space required to process the Q_i^{th} job.

Application programs

The model reflects the assumed design of a generalized set of data storage and retrieval programs for the processing of all system jobs. The envisioned program was partitioned according to the following functions:

- (P1) job examiner—selects job from job queue and routes to appropriate checking routine.
- (P2) initial job processing—checks statement structure and coded elements.
- (P3) table build—compiles job statement using data set descriptor tables.
- (P4) macro-generation—creates macro instructions from tables built by P3.
- (P5) job execution—executes macros built by P4.
- (P6) retrieval processing—fetches data from disk by a direct access method.
- (P7) update processing—creates and modifies data records, stores records on disk unit, and updates indexes.
- (P8) output message processing—formats the resultant data for output presentation to the appropriate device(s).

The processing programs were assumed to range in size from 500 instructions to 4.9K instructions; with a total of 13K instructions for all component processing programs. The basis for this characterization of the processing programs is largely empirical in nature. Similarly the amount and nature of executed code in each program module was established in light of the system job types. The representation of processing times was established on the basis of these factors as applied to the modeled processor's operating characteristics.

Core storage allocation

The allocation of available core space was based on the following assumptions of space requirements.

- control program 11.5K words
- buffer for 8 operator terminals 4 K
- application programs (maximum) 13 K
  (minimum requirement—4.9K)
- work space per active job 3 K

Since the postulated control program was not sufficiently defined to allow a dimensional partitioning, it was assumed to reside at all times in 11.5K words of core. Collecting other assumed minimum requirements for core space resulted in the gross determination that 24K words of core would be the minimum size modeled. In pursuit of the core size vs. system performance question, three core sizes were modeled: 24K, 32K, and 40K words. For each core size, used to the maximum extent possible considering the size of each of the job processing programs, the number of programs which could simultaneously reside in core storage was established. As depicted in Figure 3, three basic designs resulted; viz.,

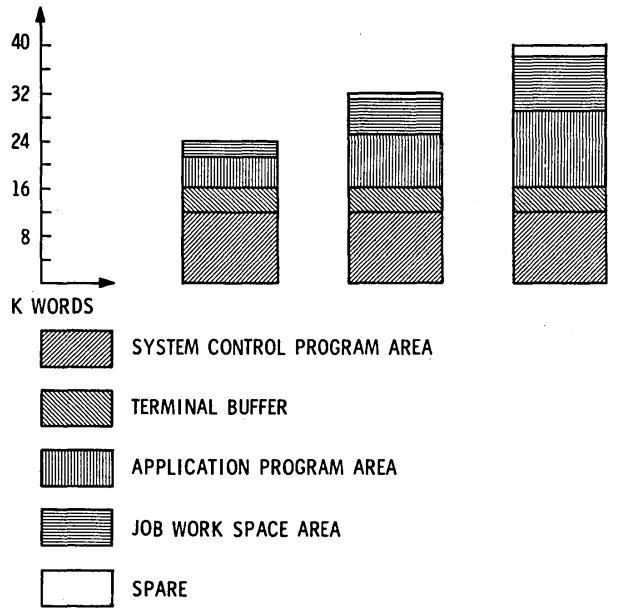

Figure 3—Designated core memory space

- 24K core size—a predominantly program overlay type of design, and allowed for only one job to be processed at a time.
- 32K core size—a design featuring program-read overlap with processing, and provided for the grouping of two jobs through the processing sequence; i.e., multi-job processing limit of two.
- 40K core size—an essentially resident type of design, with a multi-job processing limit of three.

Thus the increases in available core space, in 8K word increments, were reflected in the model by providing in each case an increased capability in processing.

Disk storage allocation

The allocation of the disk storage space available on the four mounted disk-pack drives was performed for the following categories of information: programs, pre-formatted output messages (i.e., instruction cues for the user), data set descriptor tables, and data sets. As depicted in Figure 4, information (e.g., processing programs, data set descriptor tables, etc.) which was to be requested with a predictably high frequency was stored near the median cylinder of each disk drive; thus attempting to minimize the arm motion incurred in disk accesses.

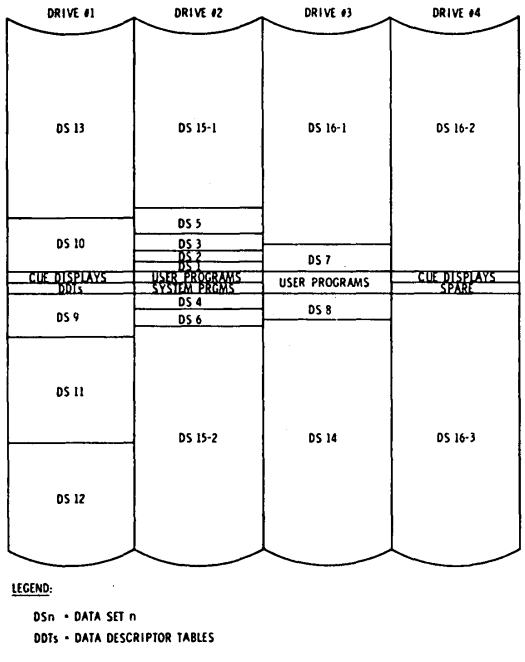

Figure 4—Disk storage map

Operation of the model

The operation of the model is depicted in gross form in Figure 5.

To provide a common basis for comparison of run statistics, the following conditions were maintained for all runs:

- simulated system operation time was arbitrarily set at 15 minutes;
- the system was initialized by setting the status of the processing programs as being initially in or out of core, according to the core size design;
- the disk arms were prepositioned to cylinder #50;
- all user terminals were activated simultaneously at the start of each run.

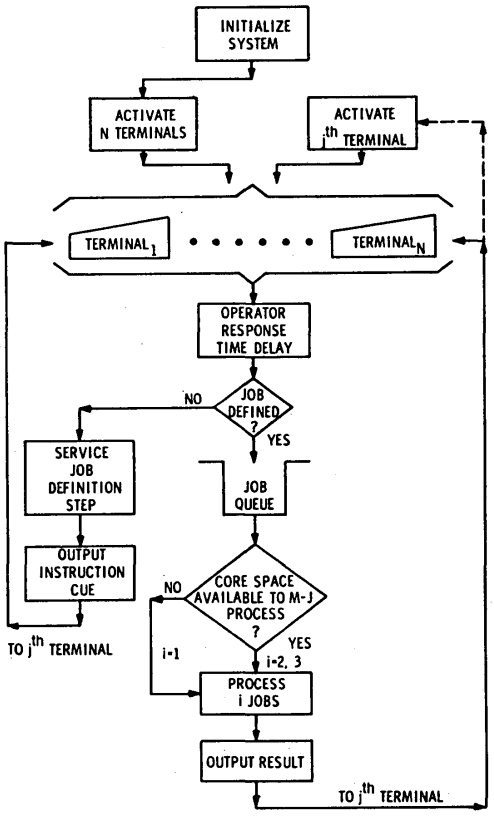

Figure 5 — Operation of the model

Results

The results presented here were selected as the most pertinent of the large volume of statistics resulting from the simulation. They are partitioned into four categories, viz., job queue statistics, job turn-around-time, job processing time, and equipment utilization.

Job queue statistics

The average number of jobs in the job queue and the average time spent by a job in the queue are shown in Table I for five different simulation runs. Results are included for the multi-job processing scheme as described previously (ref., The Model), and furthermore for a single job processing scheme at the 32K and 40K word core sizes. The latter two runs were achieved by inhibiting the multi-job processing capability; i.e., setting the multi-job limit equal to one.*

	MULTI-JOB PROCESSING			SINGLE-JOB PROCESSING	
CORE MEMORY SIZE (WORDS)	24K	32K	40K	32K	40K
AVERAGE NUMBER JOBS IN QUEUE	.12	.09	.08	.11	.11
AVERAGE TIME IN QUEUE (SECONDS)	.50	.34	.29	48	44

Table I — Job queue statistics

The results illustrate some advantage in the multi-job processing scheme. Specifically, in the 32K system with multi-job processing, jobs spent 28% less time in the job queue than for the 32K system with a single-job processing strategy. The 40K multi-job processing scheme showed about a 30% advantage over its single-job processing counterpart. From the user's (macroscopic) view, these differences could be considered operationally insignificant.

That the average number of jobs in the job queue was less than one job for all sizes suggests that the processor, even for the 24K design, frequently found the queue empty and had to wait for work. Clearly the rate of input was lagging behind the processing rate.

Job turn-around time

The average and maximum job turn-around times for the 24K-40K multi-job processing runs are depicted in Figure 6. Job turn-around time as defined earlier, is the time measured from the start of the job definition sequence to the time of the final job output.

As shown in Figure 6, there were no operationally discernible differences, again from the user's view, in average job turn-around time for the three core sizes; the values are essentially constant at 42 seconds.

The maximum job turn-around times were encountered over relatively few jobs at the beginning of each run as a result of the simultaneous terminal starts. Thereafter, the majority of the jobs fell very close to the average value. Specifically, for the 24K design,

*In this case, the 40K word design required 31.5K words of memory storage.

93% of all the jobs were turned-around in <44 seconds; for the 32K and 40K designs, 95% and 96% were turned-around in <44 seconds, respectively. For each run there were essentially 150 jobs processed in the system. The sharp drop in the curve from 24K to 32K shows again the effect of multi-job processing when queueing is a factor.

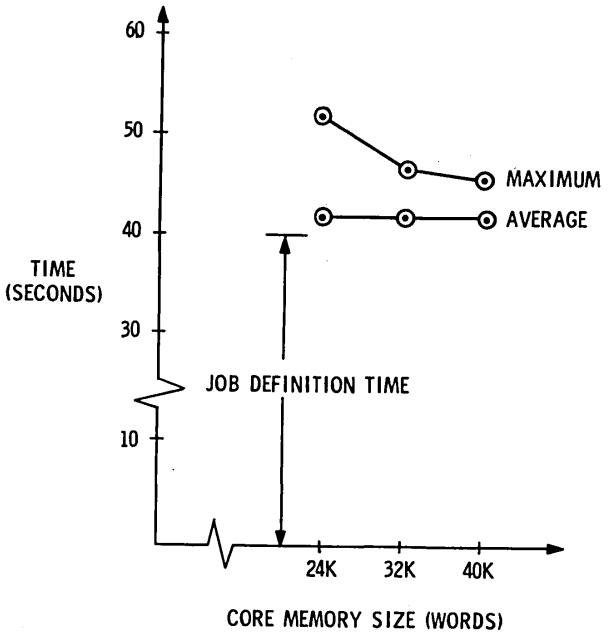

Figure 6—Job turn-around time (multi-job processing)

Recalling that 40 seconds of the average job turn-around time is attributable to user response delays, one can see in Figure 6 that the average rate of processing defined jobs is on the order of two seconds. Here, then, is the explanation of the lack of buildup in the job queue. With each user requiring about 40 seconds to define a job, the combined group of users can input an absolute maximum of one job every five or six seconds, while the processor can handle defined jobs at about three times that rate.

Job processing time

There were differences, though small ones, in the average processing times for each run. Processing time, in review, is defined from the time of input of a defined job at the job queue to final output at the terminal. Figure 7 depicts the average processing time for multi- and single-job processing, as well as for a defined-"effective" multi-job processing time.

Curve A of Figure 7 is drawn from the results for the multi-job processing scheme. In the 32K model, two jobs were taken from the job queue whenever possible and processed together to completion. In

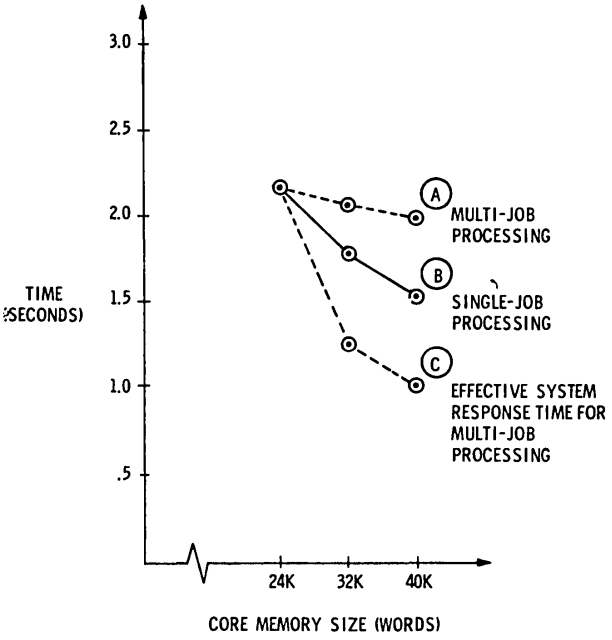

Figure 7—Average job processing time

40K, up to three jobs were processed concurrently when they were available in the job queue. This approach moves more jobs through the system in a given amount of time than does the single-job processing scheme, by avoiding, among other overhead functions, repetitious program reading. For example, if two jobs processed individually take two seconds each, the two processed together might take a total of 3.9 seconds for a time savings of 0.1 second. Note that for the jobs processed individually, the average is two seconds, while the two processed together, by virtue of waiting for each other, average 3.9 seconds each in the processing area. On occasion, the user might in fact sense this real delay as his job, the first in the job queue, waits for other jobs in the batch to be completed. Nevertheless, the system would experience an over-all improvement in output rate. Curve A shows the average job processing time, perhaps more accurately called average time in the processing phase, as seen by the terminal user.

Curve C represents an alternate viewpoint of the multi-job processing capability by showing the "effective" system response time. The effective system response time for a job in a multi-job processing system is defined here as the quotient of the processing time for a batch of jobs processed together divided by the number of jobs in the batch. Thus, from our prior example, two jobs processed together in 3.9 seconds would be said to have an effective system response time of 1.95 seconds each. But this view of system

performance although useful to the system designer is more relevant to the system manager than the system user.

Since there were no significant buildups in the job queue, the multi-job processing solution had little opportunity to produce any important processing gains relative to the single-job processing scheme. A comparison of Curve B, drawn from the results of the single-job processing runs (multi-job processing inhibited in 32K and 40K) with Curve A demonstrates that this was in fact the case.

Equipment statistics

The cpu utilization, disk channel, disk arm and magnetic tape channel utilization statistics for each run are listed in Table II.

	MULTI-JOB PROCESSING			SINGLE-JOB PROCESSING	
CORE MEMORY SIZE (WORDS)	24K	32K	40K	32K	40K
CPU UTILIZATION (%)	9.0	6.6	5.8	9.0	8.9
DISK CHANNEL UTILIZATION (%)	24.2	19.8	11.9	24.2	12.3
DISK ARM UTILIZATION (%)					
• ARM #1	6.9	7.5	5.6	8.9	5.7
• ARM #2	18.4	11.3	3.8	15.4	4.4
• ARM #3	11.1	10.2	10.1	11.4	11.2
• ARM #4	2.5	3.2	2.7	2.7	2.6
MAGNETIC TAPE CHANNEL UTILIZATION (%)	.2	.2	.2	.2	.2

Table II – Equipment utilization statistics

For all core size designs, the simulated cpu was performing useful work; that is, not idling, for less than 10% of the simulated run time. This apparently low percentage should be viewed in light of the fact that for approximately 66% of the time the cpu is forced to idle because there are no jobs in the system, and therefore the maximum possible cpu utilization in the type of operating environment modeled is approximately 33%. The decreasing cpu utilization with increasing core size is attributed to the reduced requirement for executing program loader and initialization functions.

The disk channel utilization, which represents the percentage of the simulated run time during which commands or data were being transferred via the disk channel, shows a decrease at the increasing core size designs. This reflects the impact of not reading processing programs from the disk at the higher core sizes. Thus, it can be interpreted that for the 24K core size design approximately one-half of the disk channel utilization time is spent in transferring processing programs from disk to core.

The utilization of the magnetic tape channel, which was modeled as a path distinct from the disk channel, was extremely low and constant for all core size designs. The constant 0.2% value is attributable to the fact that of the job types modeled, only the file update job was defined as requiring data transaction recording on the tape. Both the frequency of occurrence of this job type as well as the volume of data transferred was low.

Disk arm utilization, per arm, is relatively constant for all core size designs with the multi-job processing capability. The reading of processing programs from disk arm #2 accounts for the higher utilization at the lower core size designs. Similarly, the disk arm #2 utilization for the 32K core size design with single-job processing is higher than the 32K design with multi-job processing because more program reads were executed to handle the jobs singly than in groups. Disk arm utilization is a valid criterion for evaluating the merits of a disk map relative to the reliability of disk operations. That is for any given core size design, a constant utilization across all four disk arms reflects an evenly distributed work load and minimizes the relative vulnerability of any particular disk arm to failure.

SUMMARY

The primary goal of this effort was to examine from the system user's point of view, the effect of core memory size on system performance. For a dedicated special-purpose type of multiple-access information system such as the one modeled, it was evident that the advantages to be gained through incremental increases in core storage space could only be reflected in the model, as well as in the actual system, by increasing the system's apparent processing capability. The specific interest here was not in validating a specific software design, particularly since at the time the activity was conducted neither a control nor application program solution existed. Therefore only an envisioned design, deemed to be appropriate and responsive to the implied requirements of the system, was modeled on a broad, or macro, level Starting with a 24K word core memory size which was based on certain assumptions of program, buffer, and work space requirements, the capability of the processing system was improved for additional increments of 8K words by increasing the available user work area. In the modeled system, this allowed for more appli-

cation programs to co-exist in core memory and increased the multi-job processing limit.

The most significant characteristic of the system, as demonstrated by the simulation statistics, was the slow job input rate relative to the fast job processing rate; viz., a ratio of approximately 1:3. Since the user terminals were the sole source of jobs into the system, then the man-machine mechanism for entering job statements was most significant in establishing the input rate. This disparity in rates was such that there was no significant queueing in the system at any core size solution. It is evident that the operational configuration represented in the model could withstand substantial change and not appreciably affect the job turn-around time. With the same job types and operational procedures, doubling the number of active user terminals, for example, would only raise the job input rate to approximately two-thirds of the processing rate. With eight user terminals active, halving the number of steps in the job definition sequence or halving the speed of the modeled cpu would have similar effects.

The cpu represented in the model is characteristic of the processing power usually associated with multiple-access time-sharing systems, and on that basis was *a priori* considered to be an appropriate hardware solution. As the run results indicate, however, the cpu overpowers this system's processing problem. A cpu with one-half to one-third of the execution speed modeled (i.e., an average instruction execution time between 20 μ seconds and 30 μ seconds) could still result in an input to processing rate ratio of less than unity, and thus no significant queueing in the system. The processor's response to user input would be slower, of course, but imperceptibly different to the system user.

Based on an understanding of the operational requirements of this system and some prior association with information handling programs, a level of detail was established in the model which would support the objectives of the simulation. Certain quantitative assumptions established the minimum core size to be modeled at 24K words; this was an arbitrary reference point from which to examine the effect of core memory size on system performance. The run statistics showed that as the core size, and hence the processing capability, increased there was no appreciable change in system performance from the system user's point of view. We conclude that not only is the smallest core size modeled adequate, but that lower core memory sizes (i.e., 16K words or less), which would require considerably more program overlaying in the control and user program areas, would provide an adequate and responsive solution. Implicit in this conclusion, is that an improved processing capability, such as multi-job processing in the modeled system, is of little value in a system where the job input rate is considerably slower than the processing rate.

Admittedly, certain of these results could have been determined through analytical methods. This is largely due, however, to the nature of the chosen example system. In general, the system designer's choice of analysis methods should not be delimited by only the immediate considerations, but should instead be made in light of the entire planned system development activity. A simulation program model, even in the case of the cited example system and however gross in its initial form, provides the designer with an important analysis tool which can be modified and refined part by part as the system design phase progresses so as to evaluate and verify the evolving solution.

A multiple-access, information storage and retrieval system in which the job input rate is considerably slower than the job processing rate has been discussed here, whose design requirements contrast rather sharply with those of other multiple-access computing systems. This special purpose system is representative of a type which can be expected to become widespread as data processing extends to the smaller installations and users. The handling of current account data on credit applicants in a merchandise chain or the maintenance of current warehouse inventories by wholesale distributors are examples of such systems which typically would not require extensive processing power. The information storage and retrieval processing requirements of these types of installations as those of our cited military system, can be satisfied through straight-forward programming solutions, implemented on computers with moderate speeds and with modest core memory capacities.

This activity, furthermore, has served to emphasize the value of a macro level modeling approach to a system design problem; which in the authors' opinion is a fact not commonly enough recognized. When employed as early as is reasonable in the design process, it can yield, with an economy of effort, valuable insight to the system under study.

REFERENCES

1 J McCARTHY
Time sharing computer system
In Computers and the World of the Future M. Greenberger ed. The MIT Press Cambridge Mass pp 221-236 1964

2 J B DENNIS
A multiuser computation facility for education and research
Communications of the ACM vol 7 pp 521-529 September 1964

3 A H TAUB
On time-sharing systems, design and use
Proceedings—IBM Scientific Computing Symposium on Man-

Machine Communication pp 9-16 1965

4 R M FANO
The MAC system: the computer utility approach
IEEE Spectrum pp 56-64 January 1965

5 J I SCHWARTZ et al
A general-purpose time-sharing system
Proceedings—Spring Joint Computer Conference pp 397-411
1964