MAC-TR-17

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

TIME-SHARING ON A MULTICONSOLE COMPUTER

by Arthur L. Samuel

Thomas J. Watson Research Center, IBM Yorktown New York

March 1965

ABSTRACT: After a brief historical review and a description of the three basic types of time-sharing systems, the general purpose time-sharing system as exemplified by the M.I.T. CTSS system is described in general terms, with particular attention to the way the system looks to the user.

This is an article to be published in the NEW SCIENTIST. Dr. Samuel is a visiting Professor at M.I.T., and is closely connected with Project MAC.

"Work reported herein was supported in part by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under the Office of Naval Research Contract Nonr-4102 (01). Reproduction in whole or in part is permitted for any purpose of the United States Government."

TIME-SHARING ON A MULTICONSOLE COMPUTER

Why all of this sudden interest in time-sharing? Computers are usually shared by many users. In fact, one of the significant trends of the last several years has been the development of elaborate supervisory programs, the so-called operating systems, which manage the computer installation and schedule the work for various users with a minimum of human intervention. This scheduling takes various forms, but the more common form involves stacking of jobs according to their expected operating times. Small jobs are given priority during the normal working day and each job in turn is run to completion if this is at all possible.

Recently, an earlier form of operating system known as "Time-Sharing" has come back into favor and is receiving a great deal of attention, particularly in the United States. It may be of some interest to survey the current state of the art in this old, but now very new way of using computers, and perhaps make a few predictions as to the future trends.

First, a word or two by way of definition. The term "Time-Sharing" is used to identify a computing system with a number of independent, concurrently-usable consoles, or operating panels. These consoles are serviced by a supervisory program, which provides each user with the illusion that he has a direct line to the computer and that he can make use of this computer more or less as if he were the sole user. This illusion is achieved either by limiting the size or type of job that the system will accept, or by commutating the services of the computer between the various users. Actual systems differ with respect to the frequency with which each user is serviced but, in general, an attempt is made to limit the maximum delay experienced by the user to a time commensurate with his reaction time. Given the proper balance between computing speed and the total number of users allowed on the system, it may still be possible to do enough computing for each individual user to meet his

needs or to match the input and output speed capabilities of his console. He may, therefore, be quite unaware of the quasi-simultaneous use of the computer by the other users. Restricting the term "time-sharing" to describe this particular form of computer operation is, of course, somewhat inexact (multiple-console, on-line computing would be preferable), but this is what "time-sharing" has come to mean.

Two, or perhaps three, distinct types of time-sharing installations are already discernible. We can avoid misunderstandings if we draw a sharp distinction between these different kinds of systems. The first type, and the one meant more often than not when the term "Time-Sharing" is used, is the "General-Purpose System." A "General-Purpose, Time-Sharing System" attempts to provide each user with the full range of capabilities (except perhaps for speed) which he would have if he were the sole user of a general-purpose computer. The user should be able (1) to use all components of the computer system, (2) to work in any desired programming language, (3) to use programs and sub-routines which were written, originally, for other (presumably non-time-sharing) systems, with little or no change, (4) to wait for the results or to have the computation continue either in his absence or while he turns his attention to a quite different task using the same console, (5) to communicate freely with the computer both via a conventional typewriter and through some graphical device (a cathode ray tube with a light pen attachment, or the equivalent), (6) to store large amounts of data and program material within the system to which he can gain access as desired on a moment's notice, and finally, (7) to have the use of a large library of service routines and "debugging" aids (What an inelegant term!) (Note 1). There are in existence no systems which actually provide all of these services (the S.D.C. time-sharing system comes very close, Fig. 1), but this is the goal.

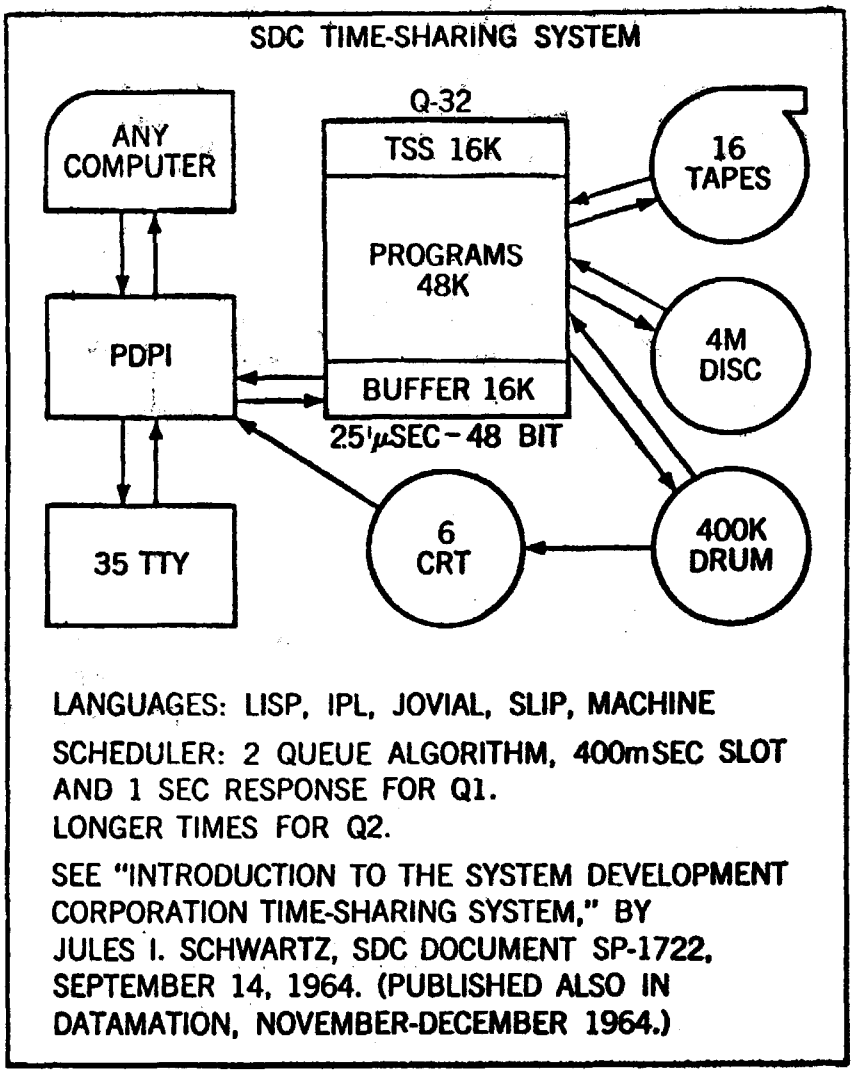
SDC TIME-SHARING SYSTEM
ANY COMPUTER
Q-32
TSS 16K
PROGRAMS 48K
BUFFER 16K
25 μSEC - 48 BIT
16 TAPES
4M DISC
400K DRUM
PDPI
35 TTY
6 CRT
LANGUAGES: LISP, IPL, JOVIAL, SLIP, MACHINE
SCHEDULER: 2 QUEUE ALGORITHM, 400mSEC SLOT AND 1 SEC RESPONSE FOR Q1.
LONGER TIMES FOR Q2.
SEE "INTRODUCTION TO THE SYSTEM DEVELOPMENT CORPORATION TIME-SHARING SYSTEM," BY JULES I. SCHWARTZ, SDC DOCUMENT SP-1722, SEPTEMBER 14, 1964. (PUBLISHED ALSO IN DATAMATION, NOVEMBER-DECEMBER 1964.)

Figure 1

A second type of system sometimes goes by the name of a "Dedicated-System," but a better name would be a "One-Language System." Here, no attempt is made to allow the user a variety of languages, and indeed he is constrained to use but a single language which may, for example, be simplified FORTRAN (as in IBM's QUIKTRAN, Fig. 2), or an even simpler language (JOSS, as used by the Rand Corporation, Fig. 3). In effect, the user is presented with a special computer which is literate in but a single language.

Finally, we might distinguish a third class of systems in which the language of the computer is still further restricted and specialized to deal with a restricted set of problems, a common library facility, or a common data base. The SAGE air defense system was an early system of this type. The SABRE air-line reservation system is a more recent example, as are the many information retrieval systems that are being talked about. Computer-based-instruction systems also properly fall into this third category, although there is a tendency to think of them as being quite unique. In the common-data-base system the user can pose only a restricted class of questions about the data base, solve only certain types of problems, for which the solution methods are already available within the system, or provide data in the form of answers to questions which the system itself may ask.

We will devote most of our attention to the general-purpose, time-sharing system. Here, by far, the best known example is the M.I.T. compatible time-sharing system (CTSS), particularly as now used in Project MAC (Note 2), and we will, accordingly, refer to it when we wish to be specific (Fig. 4).

A bit of history may be in order. The Bell Telephone Laboratories operated some of their early relay computers

Quiktran

One-language system with up to 40 terminals on an IBM 7040/7044. Designed for engineering and scientific problems in the desk calculator to small computer range. Available for customer installations and also as leased service through terminals installed on customer premises.

All man-machine communication in the source language (compatible Fortran subset) and in a conversational mode. Incremental translator with powerful debugging features but with substantial degradation in execution efficiency.

See Remote Computing - An Experimental System, Part 1, T.M. Dunn and J.H. Morrissey; Part 2, J.M. Keller, E.C. Strum and G.H. Yang; Proceedings Spring Joint Computer Conference, 1964,pp 413-443.

Figure 2

JOSS (Johnniac Open-Shop System)

JOSS serves up to eight users simultaneously at The Rand Corp. on the 1951-53 built Johnniac, responding typically in a fraction of a second to all but extended computing requests. The user may give direct commands or indirect instructions in a stored program. The system features rapid interaction, exact input, decimal arithmetic, exact output, report quality formatted results, and happy users.

EXAMPLE

[User's request; typed in green on system]

1.1 Type x, sqrt(x), log(x), exp(x), (x+.25)/x in form 1.
Form 1:

Do step 1.1 for x=1(1) 4.

[Machine reply; types in black on system]

1	1.00000000	.00000	2.7183	1.25000
2	1.41421356	.69315	7.3891	1.12500
3	1.73205081	1.09861	20.0855	1.08333
4	2.00000000	1.38629	54.5982	1.06250

See Shaw, J.C., JOSS: A Designer's View ..., Proceedings Fall Joint Computer Conference, 1964

Figure 3

```
login t104 samuel
w1407.9
PASSWORD
 T0104  2484 LOGGED IN  01/28/65  1408.1

 CTSS BEING USED IS     MAC112
SHIFT        MINUTES
      ALLOTTED     USED SINCE 01/28/65 1335.2
 1        100          0.5
 2        100          0.0
 3        100          0.0
 4        100          0.0
LAST LOGOUT WAS 01/28/65  1354.3
TRACK QUOTA=       P,     1500 Q.  0023 TRACKS USED.
R 8.483+.666

resume mon04 5
W 1410.4

 CTSS UP AT 1332.9 01/28/65.

  NUSERS= 35    TIME= 1410.5
   1.8   1.8    BACKGROUND,
  70.5  70.5    FOREGROUND,
  16.9  16.9    SWAP TIME,
  10.8  10.8    LOAD TIME,
  30.7  30.7    USER WAIT,
  13.3  13.3    SWAP WAIT,
   7.5   7.5    LOAD WAIT.

  NUSERS= 35    TIME= 1415.6
   0.    1.5    BACKGROUND,
  68.6  70.3    FOREGROUND,
  17.8  17.0    SWAP TIME,
  13.6  11.1    LOAD TIME,
  30.4  30.7    USER WAIT,
  13.9  13.4    SWAP WAIT,
   8.3   7.6    LOAD WAIT.
 QUIT,
R .350+1.016

logout
W 1416.1
 T0104  2484 LOGGED OUT  01/28 /65   1416.1
 TOTAL TIME USED=   00.4 MIN.
```

Figure 4

with remote consoles, and so deserve the credit for the first "time-sharing" systems (Note 3). This early operating procedure was then largely replaced by the elaborate operating systems to which reference has already been made. An Englishman, Mr. Christopher Strachey, in 1959 was one of the first to advocate the time-sharing idea as applied to the large general-purpose computer. Others in the United States, and in particular Dr. J. McCarthy, Dr. J.C.R. Licklider, and Dr. F. J. Corbato, were prominent about the same time in promoting time-sharing. In retrospect, it is perhaps surprising that this idea should have remained virtually unused for twenty years until its reintroduction by these workers. Undoubtedly many factors were involved, but economic considerations certainly played a part and the development of successful time-sharing systems had to wait for advances in back-up stores, and in large, fast stores.

The argument goes something like this: In the early days, computers were really very expensive, both absolutely and in terms of the cost per operation. The art of programming (preparing instruction lists) was not very advanced and users could, and usually did, require substantial amounts of time between runs to analyse their results and to locate program errors. Furthermore, there was a tendency to apply computers to problems which were repetitive in nature, and problems once programmed were usually run a great many times. Through the years, the computers have gotten very much faster and the cost per operation has decreased precipitously. Programming has become a recognized profession with the insatiable demands for programmers currently outstripping the supply. A job which would take several hours on an early computer now may take only five minutes, programming costs have risen, and computers are now being used for many one-shot problems. This has led to the development of problem-oriented programming languages and the development of larger and better operating systems.

More recently, we have seen an increase in emphasis on "turn-around" time as compared with "through-put" rate. We are discovering that it is no longer wise to let the user wait for service on a computer, and now we can, with equanimity, make some sacrifices in computer efficiencies if we can offer the user immediate access to the computer. Since the conventional operating system does not permit this immediate access, a change in operating procedures seems indicated.

Such a change would not be practical were it not for the recent development of large random-access storage devices, particularly in the form of disc files. Magnetic tapes require substantial amounts of operator time to mount and unmount and they must be read serially. When these were used exclusively for program and data storage and as back-up stores it was impractical to try to interleave several programs at one time. The disc file, storing data measured in millions of words and allowing random access, altered the balance, and one can now store an entire program library for many users within the machine. One can also maintain several programs in a running condition by giving each user a small quantum of time in a round-robin fashion, the only constraint being set by the ratio of "swap time" (Note 4) to running time, which must be kept within tolerable limits.

While we have normally assumed that each user's program will require the entire fast store of the machine, even this assumption is no longer necessary with the availability of larger and larger fast stores. It is now becoming possible to "space-share" the fast store and so greatly reduce the need for "swapping." In fact, it is possible to overlap the execution of one program with the swapping of two other programs so that little or no time is lost. Whether or not there are several users' programs in the high-speed store at the same time, a certain amount of space-sharing is, of course, essential to permit the supervisor program to remain

in control of the system while a user program is being run. The CTSS system, as currently operated in Project MAC, makes but limited use of space-sharing; but at least one current system, the IBM TSM system (Fig. 5), makes rather extensive use of this principle. Still another system, the IBM 44X (Fig. 6), was specifically designed to permit the study of various possible space-sharing techniques. Substantially, all of the new systems now under development will space-share. This, then, is a third factor responsible for the current interest in the time-sharing idea.

Professor Corbato, of M.I.T., emphasizes yet a fourth factor, having to do with the unusually high degree of equipment reliability required for satisfactory on-line service, a degree of reliability that was not always obtainable on some of the earlier computers and that is only now being routinely achieved. Not only is the on-line user more dependent upon continued operation of the system than is the stacked-job user, but maintenance problems are also much more difficult on a time-shared system, since the concatenation of circumstances leading to performance errors and failures can usually be neither determined nor reproduced.

There is one additional hardware requirement which must be met before one can build a truly successful time-sharing system. This is the requirement for a satisfactory storage-protection scheme. Each user, and each user's program, must be restricted so that he and it can never access (read, write, or execute) unauthorized portions of the high-speed store, or of the auxiliary store. This is necessary (1) for privacy reasons, (2) to prevent a defective program from damaging the supervisor or another user's program, and (3) to make the operation of a defective program independent of the state of the rest of store. This latter reason is often overlooked by the uninitiated, but debugging a program on a time-sharing system without storage

IBM TSS SYSTEM

Experimental, general purpose system.
20 users on a 7090 via 1050 consoles.
Multiqueue scheduling with space sharing.
Basic quantum 1/4 sec., few sec. response.
Also console initiated background.
Languages: FAP, FORTRAN, GPSS, PAT.
See The Time-Sharing Monitor System by H. A. Kinslow; Proceedings Fall Joint Computer Conference, 1964, pp 443-454.

Figure 5

IBM 44X

Experimental system on a modified IBM 7044 with extended addressing capability (2 million words) and multiple-register relocation facilities for pagination. Variable number of IBM 1050 terminals via IBM 7750. Large capacity core store (1/2 million words @ 8 μs) in addition to usual 32 K 2 μs store.

Designed specifically for experimental study of various time-sharing, space-sharing and multi-programming algorithms.

Figure 6

protection can be a most frustrating experience. Storage protection can, of course, be obtained through programming means, but not, however, without some loss in operating efficiency. It is now generally conceded that storage protection can best be handled by special hardware in the computer.

Storage-protection hardware already comes in a variety of forms, and new forms are still appearing. The IBM 7094, as used in the M.I.T. system, contains a relocation register and upper and lower bound registers whose contents are alterable only by the supervisory program. This is, perhaps, an essential minimum. Access to the disc store and to all peripheral equipment is usually controlled by the supervisor through the medium of a directory, and each user refers to his storage regions by file name rather than by location. Many time-sharing systems do not assign fixed storage locations to each user, but make contiguous blocks of storage available on demand up to previously assigned quotas by means of a list-structuring procedure. More elaborate systems make the auxiliary store appear to the user as a simple extension of the fast store (the one-level store idea) with the mechanics of transferring program and data from the auxiliary store to the high-speed store handled automatically (Note 5).

We will be able to discuss but one more aspect of a typical time-sharing system -- this having to do with the so-called "scheduling algorithm." The scheduling algorithm is that set of rules, or more strictly its embodiment in a program, which govern the frequency with which each user is serviced and the length of time each program is permitted to run before it is interrupted. We can note two extreme situations, the one in which each user program is run to completion, and the second in which the time slot is so short that many periods are required before even the shortest job can be completed. Some time-sharing systems

actually operate in accordance with the first scheme. These differ from the conventional stacked-job system only in their use of a number of independent consoles. The illusion of immediate access can still be maintained, by setting stringent limits on the maximum permissible size of programs, in terms of running time. M.I.T.'s CTSS is perhaps more typical in that it maintains a series of queues and runs each program for only a short period of time. The correct form for a scheduling algorithm is one of the most hotly discussed aspects of a time-sharing system, and a great deal of experimental work will still have to be done before the question can be settled.

It will be apparent to even the casual observer that it requires much more than the bare hardware to make a successful time-sharing system. In the first place there must be a supervisor program if the user is to be able to control his apparent computer through the medium of a remote console. Besides the supervisor itself, there must be a large library of service routines which can be called by name and which will do for the user many of the things which he might ordinarily handle by his own programming efforts. Associates from whom he might borrow routines are no longer close at hand and he is more or less forced to limit his input and output to an amount consistent with the speed of his console. To illustrate, the library of service routines accessible to each user on the Project MAC computer is in excess of 500,000 words. Establishing a usable general-purpose time-sharing system is a horrendous job (Note 6), and most systems are evolutionary in concept.

Let us now turn our attention to the user of a time-sharing system. We have been tacitly assuming that this user would be attempting to solve the same class of problems he had previously solved on a stacked-job installation. This certainly is part of what he will want to do, but it is not all. The rapid response time of the time-sharing system

leads to changes in programming methods and, more important, it makes possible a redistribution of work between the human problem solver and the computer.

This redistribution is particularly useful in dealing with problems that are not well structured. It often happens that more time is involved in formulating a problem than in solving it. Frequently, as Dr. Richard Hamming, of the Bell Telephone Laboratories, has said, "The purpose of computing is not to get numbers but to gain understanding." When we use a computer to gain understanding, it still gives us numbers and a major part of our task is to formulate the problems so that the numbers do indeed lead to understanding. When confronted with a long turn-around time and a high fixed-overhead charge per run, one must either wrestle with this aspect for a long time before assigning the problem to the computer or risk the expenditure of needlessly large amounts of computer time and the production of large amounts of useless output.

By way of contrast, when one can have access to the computer on an unscheduled basis, when one can observe the course of a computation and alter it at any time, and when it is no longer uneconomic to solve a problem piecemeal or pose very small problems, one can have numbers available in all stages of the problem formulation. For example, if one has doubts about the convergence of some approximate method, one simply tries it out. Instead of spending several hours checking a program for clerical errors, one lets the "assembly" program list the errors for him. If a program does not seem to be giving the right kind of answers or if it simply takes too long, one stops it. Instead of asking for a memory dump, one asks to see the particular items which are thought to be significant. Instead of wasting time printing out all of the output from a run, one leaves the output stored on the disc and simply prints out the numbers needed at the time. One procedure, now only but dimly

envisioned, has to do with cooperative procedures enabling several people to work on different parts of a single problem, with the coordination of their efforts and the interchange of information between the workers effected by the computer. Mr. Jules Schwartz, of the Systems Development Corporation, is a strong advocate of this usage. Even more drastic changes in our conventional methods of formulating and solving problems are sure to evolve as time-sharing systems are perfected and become more generally available.

Does this mean the demise of batch processing and its replacement by time-sharing? The world is never this simple. Many of the larger computer users have individual jobs which run for hours. It may not be sensible to split such a job into small portions interspersed with other jobs. Some users must have large amounts of outputs. Some tasks recur on a fixed schedule, or run for a fixed length of time. Still others have such overriding priority that they must have the exclusive use of the computer. Conventional systems work well under these conditions. Some of the newer operating systems of the conventional sort offer many of the advantages of so called time-sharing systems and many avant-garde time-sharing systems, after the first blush of novelty has worn off, are developing a tendency to run short jobs to completion. Just as the competition in the past between "scientific" and "commercial" computers has disappeared with the development of universal computer systems, such as the IBM SYSTEM/360, the G.E. compatible series, and the R.C.A. Spectra 70, so we can expect an evolutionary trend toward a new system concept which combines the better features of time-sharing systems.

Let us return for a moment to consider the details of the M.I.T. time-sharing system known as CTSS. While there are actually several different time-sharing installations at M.I.T., two of these use the CTSS and are substantially identical; one is in use at the M.I.T. Computation Center

and the other is at Project MAC. The MAC computer has more users (over 200), serves a few more users simultaneously (30) and provides more hours of time-sharing service a day (approximately 20 hours per day, 7 days a week). Each of these systems uses an IBM 7094 with two 32,000-word core memory banks. Only one of these banks is accessible to the user, the other bank is used by the supervisory program. "Polling" (commutation between terminals) and some terminal-code conversion functions are performed by an IBM 7750. The user consoles, which can be connected to either system, are about equally divided between Teletype machines and IBM typewriters. Some of the latter have both card and paper-tape reading and punching equipment. There are also facilities for connecting to the TWX and TELEX networks and there are a few additional special-purpose consoles in use and under development. While many of these terminals are in the same building with the main computer or in other buildings on the M.I.T. campus, several consoles are in staff members' homes at some distance from the campus and a few people are using the system from quite remote locations ranging from Norway to California.

It will be instructive to follow the course of events as viewed by a user at a typewriter. He must, of course, first get a connection established between his console and the 7750. If he is on the campus, he can dial the computer directly; if outside, he may dial M.I.T. through the ordinary telephone system and ask the local telephone operator to connect him with the computer, or he may dial directly on the TWX or TELEX networks. When the connection is established, he hears a 1,000 cycle tone, whereupon he pushes a button to disconnect his telephone and connects the typewriter. He then types LOGIN and gives his problem number and his name. The computer responds by requesting a password and temporarily disconnecting his printing mechanism so that no printed record appears as he responds with his assigned 6-character alphanumeric private identification. If he gives

a valid password and if there are fewer than 30 users at the time, the computer acknowledges that he is logged in and supplies him with a brief summary of the state of his files and of the system. He is then free to use the computer in any way he may desire.

For example, he may wish to resume a program which he had interrupted at an earlier session and saved by typing "SAVE IAN" (IAN being the name he had arbitrarily assigned to the particular program.) If so, he simply types "RESUME IAN" and the program continues from the interruption point. Or he may wish to edit a program. If so, he calls upon one of the several available editing programs by typing its name, followed by the name of the program to be edited. These editing programs allow him many convenient features -- the ability to print out portions of the program, to correct single characters (identified either by location or by context), to add new material, etc. As still another example, he may now be ready to "compile" a program, in which case he types the name of the "compiler" he wishes to use, again followed by the name of the program to be compiled. As a final example, the user may wish to start the operation of an already compiled program and perhaps have it "stop at" some specified point, or he may wish to invoke some one of the many debugging aids which the system provides.

The user has a "quit" button on his console which enables him to interrupt his program at any time. This allows him to communicate with the supervisor program without in any way affecting the interrupted problem program (except, perhaps, for the loss of some output information which was being typed at the time of the interruption). He can type "SAVE IAN" if this is his wish, invoke the debugging aids, or simply restart the interrupted program by typing "START". Finally, service is terminated by typing "LOGOUT." The general appearance of the output as typed on a

1050 terminal is shown in figure 4, which incidentally shows a five-minute run of a monitoring program that gives one-minute-averaged and cumulatively-averaged percentage statistics for the system at a time when there were 35 users on-line.

Other systems differ in detail but offer much that is similar to the system just described. The interested reader may obtain some insight into these other systems by the brief synopses given in the figures, or he may wish to refer to some of the original papers given in the references.

Now for a word or two more about One-Language-Systems. While some people might quarrel with the classification of the IBM QUIKTRAN system as a time-sharing system, since more attention is paid to system through-put and less to time-slicing than is the case with the systems just described, QUIKTRAN is a typical one-language, time-sharing system within the broad definition given earlier. This system has been made publicly available for use on standard IBM equipment and will undoubtedly be widely used. A better known, but not generally available, one-language system is the JOSS system of the Rand Corporation, shown in figure 3. Still another system is being offered in the Boston area by Charles Adams Associates (Fig. 7). The Western Union Corp. has also announced a system which is believed to be of the same general type.

The design considerations behind common-data-base systems are perhaps of less immediate interest to the reader than are the previously described types. Common-data-base systems frequently use specially-designed terminals, adaptable for handling the special class of service which they provide. In IBM's SABRE system, as installed for American Airlines, there are over 1,000 consoles now in use which communicate with the central computer and enable the agents to inquire regarding the availability of space on

ADAMS ASSOCIATES - KEYDATA SYSTEM

On Line packaged commercial data processing, FORTRAN and other engineering packages, on a PDP - 6 with 48K core, million word drum and 33 million character disc.

Initially 16 leased-line teletype terminals with planned expansion to 256. Dynamic 32-word-page core allocations, interpretative processing. Service on first come, first served, basis either to completion or to a needed drum or disc access.

Response time should not exceed 1/4 sec more than 10% of time.

Figure 7

aircraft and to make and cancel reservations at a rate in excess of 1500 inputs per minute. It is characteristic of most common-data-base systems that the system program remains in control at all times and in effect treats the user's request as input on which the system program operates. The user may, of course, pose a fairly complicated problem but he does not actually write his own program as he does for the general-purpose time-sharing system.

Returning to the subject of general-purpose systems, the question uppermost in many people's minds relates to the maximum number of users who can conveniently be serviced in this way on a specific computer and, hence, the cost per user. Accurate figures are hard to get, and they are unreliable at best, since most of the existing systems are still experimental. One of the earliest systems in operation, the B.B.N. time-sharing system (Fig. 8) served five typewriters on a PDP-1 computer. Predictions based on this experience as to the number of users who could be served on a larger computer, the 7094 for example, have not been borne out in practice. This is not because the predictions were wrong, but simply because niceties of system design have been sacrificed in order to get the existing systems into operation. The "swapping time" problem and the need to space-share the high-speed store were usually not given due consideration and it is only now that people are facing up to these problems. Two other systems that may be of interest are the ATS (Fig. 9) and the STSS (Fig. 10).

Opponents of time-sharing argue that a properly-designed, stacked-job operating system can, by its very nature, always outperform a time-shared system in terms of through-put. The proponents of time-sharing counter by objecting to the choice of raw through-put as a valid measure of system utility, by pointing out that time-sharing offers the user a service that simply is not available with

Bolt, Beranek and Newman
Hospital Computer System

PDP-1D computer, tapes, 500 million bit Fastrand drum and data channel, 18 bit word; 5.3 μs cycle time independent 16K, 4K and 4K stores. User programs on high-speed drum. 30 simultaneous teletypewriter users. On-line MIDAS assembler plus version of J. C. Shaw's Joss. (q.v.) Multi-level queues with smallest time slot of 32 ms. Typical max. response time of 1/2 second.

Supervisor includes extensive common subroutines to enable user programs to be prepared quickly and to run in 4K of user core.

Figure 8

Administrative Terminal System

Time-shared text-editing system available to all users of IBM 1440 or 1460 systems.

Provides basic editing and desk calculator functions, paging, line width adjustment, line justification, multi-font features, character, word, line and section deletion, replacement or insertion, etc.

Up to 40 IBM 1050 terminals can be used simultaneously, working on the same or different projects.

Figure 9

STANFORD TIME SHARING SYSTEM

Computers: PDP-1, 7090.
Number of Stations: 20; 12 CRT + 8 teletype.
Swap Time: 34 m sec PDP-1, 6 sec. 7090.
User Files: IBM 1301 disk.
Languages: MACRO, GOGOL, LISP on PDP-1
BALGOL, FORTRAN, FAP, LISP on 7090.
Uses: Teaching machine laboratory, general computing, on-line data reduction.
5 user system - July 1964, 20 users - April 1965.

Figure 10

the conventional system, and by claiming that this new service is worth all that it costs. They prefer to measure utility in terms of user satisfaction, turn-around time, programming efficiencies, and net system through-put, defined not in "meg-ops" (millions of computer operations) but in terms of the useful output from the total man-machine complex as measured against all of the operating costs. The argument continues, and at the moment time-sharing seems to be winning. Time-sharing has been made to work; it is still not strictly economical, but it promises to be so in the very near future.

NOTES

1. Like it or not, the term "debugging" is now generally used to describe the process of locating debugging errors.

2. MAC is an acronym standing for Multiple-Access Computer or, as some prefer, Machine-Aided Cognition. This project is supported by the Advanced Research Projects Agency of the U.S. Department of Defense through the Office of Naval Research.

3. Your scribe has yet to find a reference to time-sharing in Charles Babbage's published work but he is still looking.

4. Defined as the time required to transfer one user's program from the high-speed store to a back-up store and to load the high-speed store with a second user's program.

5. Many commercial computers offer hardware facilities which simplify this procedure. A well-known British example is the Atlas.

6. Another reason why time-sharing was slow in catching on. Actually, the task is relatively simple if one is willing to settle for something less than complete generality.