8

THE MAC SYSTEM: A PROGRESS REPORT

R. M. Fano

Ford Professor of Engineering and Director of Project MAC
Massachusetts Institute of Technology

INTRODUCTION

The notion of machine-aided cognition implies an intimate collaboration between a human user and a computer in a real-time dialogue on the solution of a problem, in which the two parties contribute their best capabilities. In order for this intimate collaboration to be possible, a computer system is needed that can serve simultaneously a large number of people and that is easily accessible to them, both physically and intellectually. The present MAC System is a first step towards this goal.

The case for machine-aided cognition with the aid of a multiple access computer system was very eloquently argued by Professor John McCarthy in his 1961 lecture, "Time-Sharing Computer Systems."[1] The views presented in that paper were largely the consensus of an MIT committee which had just completed a comprehensive study of the future computational needs of MIT. They are also embodied in the Compatible Time-Sharing System (CTSS)[2] developed by the MIT Computation Center under the leadership of Professor F. J. Corbató, an early version of which was first demonstrated in November, 1961.[3] The MAC computer system is a direct descendant of CTSS.

The last section of Professor McCarthy's lecture introduced the notion of a community utility capable of supplying computer power to each "customer" where, when, and in the amount needed. Such a utility would be in some way analogous to an electrical distribution system. That is, it could provide each individual with logical tools to aid him in his intellectual work, just as electric tools today aid him in his physical work. One might say, in this regard, that the present state of the computer as a source of logical power is similar to that of the early steam engine as a source of mechanical power. The steam engine could generate much more power than could any man or animal, and therefore it could perform tasks that

Computer Augmentation of Human Reasoning, Sass, M.A. and Wilkinson, W.D. Eds., Sparton Books, 1965, pp. 131-150.

were previously impossible. However, the power generated could not be supplied on an individual basis to aid men in their daily work, until the advent of electrical power distribution.

The analogy between electric power and computer power illustrates only one of the aspects of a computer utility, namely, its ability to provide the equivalent of a private computer whose capacity is adjustable to individual needs. Of much greater importance to the individual customer would be the services that such a utility could make available to him by placing at his fingertips a great variety of services in the form of public procedures, data, and programming aids, and by allowing him to conveniently store and retrieve his own private files of data and programs. Furthermore, a computer utility could provide customers having common interests with convenient means for collaboration. For instance, designers working together on a complex system could check continually the status of the overall design as each of them develops and modifies his own contribution.

The MAC System is an experimental computer utility which, since November, 1963, has been serving a small but varied segment of the MIT community. It is the most extensive of several time-sharing systems in operation.[4-6] In spite of its experimental character and its limited capabilities, it has gained quick acceptance as a daily working tool. The purpose of this paper is to present a brief description of the current system, to report on the experience gained from its operation, and to indicate directions in which future developments are likely to proceed. Its scope is limited to the general organization of the system and to the basic services that it can provide. In particular, no mention will be made of the problem-oriented languages and other special programming facilities which are being added to the system by the system users themselves, thereby increasing its intellectual accessibility. Thus the work reported in this paper is a relatively small part of the overall research effort encompassed by Project MAC.

EQUIPMENT CONFIGURATION

The primary terminals of the MAC System are, at present, 52 Model 35 Teletypes and 56 IBM 1050 Selectric teletypewriters (adaptations of the "golfball" office typewriter), located mostly, but not exclusively, within the MIT campus. Each of these terminals can dial, through the MIT private branch exchange, either the IBM 7094 installation of Project MAC, or the similar installation of the MIT Computation Center. The supervisory programs of the two computer installations may, independently, accept or reject a call, depending on the identity of the caller.

Access to the MAC System can also be gained from any station of the Telex or TWX telegraph networks. Some tests and demonstrations have been conducted from European locations, and experiments are being planned in collaboration with a number of universities to provide further experience with long-distance operation of the system.

While teletypes and other typewriterlike terminals are adequate for many purposes, some applications demand a much more flexible form of graphical communication. The MAC System includes for this purpose the initial model of a multiple-display system developed by the MIT Electronic Systems Laboratory for computer-aided design.[7] The system includes two oscilloscope displays with character generator and light pen, and some special-purpose digital equipment that performs the light-pen tracking, and simplifies the task of the computer in maintaining the display, and in performing common operations such as translating and rotating the display. The two oscilloscopes can be operated independently of each other; communication with the computer can be achieved by means of the light pen, and also through a variety of other devices such as knobs, push-buttons, toggle switches, a and typewriter. The meaning of a signal from one of these input devices is entirely under program control. The whole display system communicates with the IBM 7094 of the MAC installation through the direct-data channel, and the display data are stored in the central memory of the 7094. Thus, the display must be located in a room adjacent to the computer installation. Remote operation would require the addition of a memory and some processing capacity for local maintenance of the display.

A separate, very flexible display terminal is provided by a DEC PDP-1 computer which can communicate from a remote location with the MAC computer installation through a 1200-bit-per-second telephone connection. The PDP-1 can also be used as a buffer between the MAC computer and the display system described above, thereby permitting simulation and study of remote operation of the latter.

All the above terminals can, in principle, operate the MAC computer installation simultaneously and independently by time sharing its central processor. However, the number of terminals active at a given time is limited by the supervisory program to 24 in order to insure prompt response. This number has already grown to 24 from 10, and is expected to grow further and possibly to double in the next few months, although maximization of this number is not a primary objective at this time.

The equipment configuration of the MAC computer installation is illustrated in Fig. 1. The IBM 7094 central processor has been modified to operate with two 32,000-word banks of core memory and to provide facilities for memory protection and relocation. These features, together

with an interrupt clock and a special operating mode (in which input-output operations and certain other instructions result in traps), were necessary to assure successful operation of independent programs coexisting in core memory. One of the memory banks is available to the users' programs; the other is reserved for the time-sharing system supervisory program. The second bank was added to avoid imposing severe memory restrictions on users because of the large supervisory program, and to permit use of existing utility programs (compilers, etc.), many of which require all or most of a memory bank.

The central processor is equipped with six data channels, two of which are used as interfaces to conventional peripheral equipment such as magnetic tapes, printers, card readers, and card punches. A third data channel provides direct-data connection to terminals that require high-rate transfer of data, such as the special display system described above.

Each of the next two data channels provides communication with a disk file and a drum. The storage capacity of each disk file is 9 million computer words, and the capacity of each drum is 185,000 words. The time required to transfer 32,000 words in or out of core memory is approximately two seconds for the disc file and one second for the drum. The two disk files, with a total capacity of 18,000,000 words, are used to store the users' private files of data and programs, as well as public programs, compilers, etc. The two drums are used for temporary storage of active programs, when they have to be moved out of the core memory to make room for other programs. Thus, the two drums act in this respect as an extension of the core memory. Drums with four-times faster transfer rate will be substituted for the present ones in the near future.

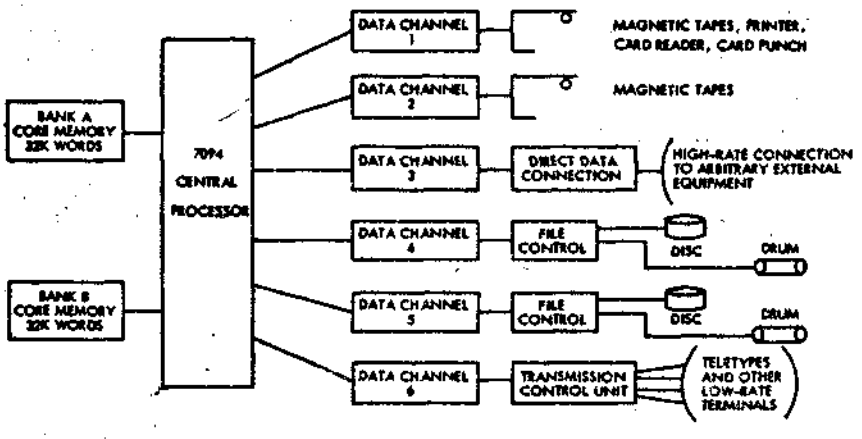

FIG. 1. Equipment Configuration.

The transmission control unit (IBM 7750) is a stored-program computer which serves an an interface between the sixth data channel and up to 112 communication terminals capable of telegraph-rate operation (approximately 100 bits per second). An appropriate number of these terminals are connected by trunk lines to the MIT private branch exchange and to the TWX and TELEX networks. Higher rate terminals can be readily substituted for groups of these low-rate terminals; for instance, on the present MAC System, three 1200-bit-per-second terminals are installed, one of which provides the communication channel to the PDP-1 computer. All such terminals are compatible with Bell System Data-Phone data sets. Part of the core memory of the transmission control unit is used as an output buffer, because the supervisory program and its necessary buffer space have grown in size to the point of occupying the whole of the A bank of core memory. The design intent was and still is to provide sufficient input-output buffer space in the main memory to eliminate unnecessary core-to-core transfers; the present mode of operation is a makeshift made necessary by equipment limitations.

The Digital Equipment Corporation's PDP–1 computer is not an integral part of the MAC Time-Sharing System, except when connected to it as indicated above. It was acquired to permit early experimentation with light-pen interaction with a display, and for other very high speed interaction work. It includes a 16,000-word core memory, Microtapes, a high-speed channel, and a scope display with character generator and light pen. It will be replaced by a PDP–6 computer in the near future.

OPERATING PROGRAM

The operating program of the MAC Computer System is a direct descendant of the Compatible Time-Sharing System (CTSS) developed by the MIT Computation Center, and described in the manual[2] published in July 1963. Many parts of the operating program have since been rewritten to facilitate system maintenance, and various facilities have been added that had been described in the manual but which had not then been implemented. Furthermore, it now includes various user-developed features such as the I-O adapter for the display system described in the preceding section, compilers for various new languages, and other programming aids. A detailed description of these newer system features is beyond the scope of this paper; they will be described in separate reports.

The heart of the MAC System is the supervisory program which resides at all times in the A bank of core memory. This program handles the communication with all the terminals, schedules the time sharing of the

central processor on the part of active programs, moves these programs in and out of core memory, and performs a variety of bookkeeping functions necessary to protect users' files and to maintain detailed accounting of the system usage. The services that the system can provide are organized in the form of commands, that is, instructions that system users can give to the system. The programs corresponding to the commands are permanently stored in the disc files; when a command is issued, a copy of the corresponding program is made, loaded, and executed just as if it were a user's program. The language facilities available in the system include FAP, MAD, MADTRAN (a translator of FORTRAN into MAD), COMIT, LISP, SNOBOL, a limited version of ALGOL, on-line, extended versions of SLIP and GPSS, and two problem-oriented languages for Civil Engineering named COGO and STRESS.

The system is rapidly evolving through the addition of new language facilities and other utility programs and programming aids. The operating program itself is now being modified by system programmers working on-line, and modifications are occasionally introduced without even interrupting the operation of the system. The entire system, exclusive of users' files, includes approximately a half-million words of code, of which a little more than 50,000 were specifically written for the system, while the rest are adaptations and modifications of compilers and other programming aids developed at MIT and elsewhere.

Some of the basic services available from the system are illustrated in Figs. 2*a* through 2*f*, which are a complete record of a demonstration session held on June 9, 1964 between 10:36 and 11:39 a.m. A total of 1.3 minutes

```
login t193 fano
W 1036.4
PASSWORD
 PARTY LINE BUSY, STANDBY LINE HAS BEEN ASSIGNED
 T0193  2859 LOGGED IN  06/09 /64   1036.7
 CTSS BEING USED IS      MAC05K
SHIFT          MINUTES
      ALLOTTED    USED SINCE 06/09/64 1036.7
 1        100          0.0
 2        100          0.0
 3        100          0.0
 4        100          0.0
LAST LOGOUT WAS 06/09/64  1036.7
TRACK QUOTA=        P,       500 Q. 0041 TRACKS USED.
R 5.550+1.000
```

FIG. 2-a. Print Out of Demonstration Session—Part a.

of computer time was used. The author's typing appears in lower case letters, and the computer replies are in capital letters. All digits represent computer replies except those intermixed with lower case text, and the two lines following "TYPE RANGE N1. N2. etc. . ." in Fig. 2*d*. Each command is immediately acknowledged by the computer with a character *W*, followed by the time of the day in which the first two digits are hours, the next two are minutes, and the one following the period indicates tenths of minute. The end of an interaction is indicated by the letter *R*, followed by the sum of the two numbers indicating the total number of seconds of computer time used during the whole interaction. The first of the two numbers indicates the processing time, and the second one the time wasted in swapping programs back and forth between core memory and drum.

```
input
W 1038.3
00010                print format text,$ $
00020                print format text, $type range n1. to n2. on 2 lines $
00030                read"d format b,fn1
00040                read format b, fn2
00050                n1=fn1
00060                n2=fn2
00070                print format text,$primes are$
00080                through loopb, for n=n1,1,n.g.n2
00090                whenaba n.1.3?whenev?          whenever n.1.3, transfer to print
00100                throught loopa,for i=2,1,i.g.(n/i)
00110                through loopa,for i=2,1,i.g.(n/i)
00120                "loopa    whenever (n-(n/i)*i).e.0,transfer to loopb
00130                print format a,n
00140 loopb          continue
00150                print format text,$range finished$
00160                execute exit
00170                execute dormant"""nt.
00180                vector values a=$(18)$
00190                vector values b=$(f20.8)$
00200                integer n,i,?          vector values text=$(12a6)$
00210                integer n,i,n1,n2
00220                end of program
00230
 MAN. delete,100
 MAN. delete,150
 MAN. file  prime mad
W 1057.6
R .600+3.616

mad prime
W 1058.4

 THE FOLLOWING NAMES HAVE OCCURRED ONLY ONCE IN THIS PROGRAM AND ARE ALL
 ASSIGNED TO THE SAME LOCATION. COMPILATION WILL CONTINUE.

PRINT

ERROR 02051 IN PART3. CARD NO.   00090
INVALID MODE FOR SOME OPERAND IN PRECEDING STATEMENT
 NO TRANSLATION.
R .600+.833

edit prime mad
W 1059.2
00230
 MAN. 130 print  , print format a a"""a,n
 MAN. file prime mad
W 1101.1
R 1.066+1.200
```

FIG. 2-b. Print Out of Demonstration Session—Part b.

The session was started by issuing the *login* command, followed by the author's problem number and name. The computer then asked for his password, which is not recorded because the printing mechanism of the typewriter is automatically disconnected while the password is typed. All the lines assigned to the author's user group were already in use, a stand-by line was assigned because the total number of lines in use was less than 24; this made the author liable to an automatic logout. The rest of Fig. 2*a* contains various information, including the allotment of computer time in minutes which had been made that very morning, and of which none had yet been used.

Figures 2*b* and 2*c* illustrate some of the facilities for writing, editing, filing, and compiling programs and for retrieving and printing private files. The *input* command causes the computer to type out successive line numbers as each statement of the program is written. The program in Fig. 2*b*, a corrected version of which is printed out in Fig. 2*c*, can be used to compute prime numbers as illustrated in Fig. 2*d*. The program is written in the MAD language. Some of the typing errors in Fig. 2*b* were intentional, others accidental. The quotation mask erases the preceding character and itself; thus, two successive quotation marks erase the preceding two characters. The question mark erases the entire line up to that point. Each line is terminated by giving a carriage return. A carriage

```
mad prime
W 1105.4
LENGTH 00205, T.V. SIZE 00005, ENTRY 00043
R 2.400+1.000

printf prime mad
W 1106.0
00010              PRINT FORMAT TEXT,$ $
00020              PRINT FORMAT TEXT, $TYPE RANGE N1, TO N2, ON 2 LINES $
00030              READ FORMAT B,FN1
00040              READ FORMAT B, FN2
00050              N1=FN1
00060              N2=FN2
00070              PRINT FORMAT TEXT,$PRIMES ARE$
00080              THROUGH LOOPB, FOR N=N1,1,N.G.N2
00090              WHENEVER N.L.3, TRANSFER TO PRINT
00110              THROUGH LOOPA,FOR I=2,1,I.G.(N/I)
00120 LOOPA        WHENEVER (N-(N/I)*I).E.0,TRANSFER TO LOOPB
00130 PRINT        PRINT FORMAT A,N
00140 LOOPB        CONTINUE
00150              PRINT FORMAT TEXT,$RANGE FINISHED$
00170              EXECUTE DORMNT.
00180              VECTOR VALUES A=$(I8)$
00190              VECTOR VALUES B=$(F20.8)$
00200              VECTOR VALUES TEXT=$(12A6)$
00210              INTEGER N,I,N1,N2
00220              END OF PROGRAM
R .416+1.016
```

FIG. 2-c. Print Out of Demonstration Session—Part c.

return (i.e., entering a null line) is also used to enter the manual mode of input, as illustrated in line 230. In the manual mode, preceding lines can be deleted or corrected by issuing appropriate commands. After deleting two superfluous lines the program was filed under the name PRIME MAD as part of the author's private file.

Next, the *mad* command was issued to cause the program to be compiled by the MAD translator. The first attempt at compilation failed because of an error for which a diagnostic was printed. The error consisted of the

```
load prime
W 1111.1
R 3.100+1.200

save prime
W 1111.4
R .866+.600

listf prime saved
W 1111.7
 6/09/64      PRIME    SAVED     P          17
R .650+1.416

resume prime
W 1112.0
#EXECUTION.

TYPE RANGE N1. TO N2. ON 2 LINES
1000000.
1001000.
PRIMES ARE
 1000003
 1000033
 1000037
 1000039
 1000081
 1000099
 1000117
 1000121
 1000133
 1000151
 1000159
 1000171
 1000183
 1000187
 1000193
 1000199
 1000211
 1000- QUIT,
R 8.400+2.216
```

FIG. 2-d. Print Out of Demonstration Session—Part d.

omission of the word *print* in line 130. The *edit* command was then used to reopen the program file, the error was corrected in the manual mode, and the program was refiled under the same name. The second attempt to compile the program was successful, as indicated in Fig. 2*c*, and the corrected program was then printed out using the *printf* command.

The binary version of the PRIME program was then loaded (together with the necessary library subroutines) using the *load* command, and the resulting core image and machine state was recorded using the *save* command. The latter command created a new file named PRIME SAVED, as indicated by the system reply to the command *listf prime saved*. The program was then started by issuing the command *resume prime*. It could also have been started by issuing the command *start prime* instead of *save prime*, or by loading and starting the program in one operation by means of the command *loadgo prime*.

The PRIME program asked for the numbers N1 and N2 defining the desired range of prime numbers; the numbers 1,000,000 and 1,001,000

```
listf
W 1118.5
    10 FILES     61 TRACKS USED
DATE              NAME          MODE    NO. TRACKS
 6/09/64      PRIME    SAVED     P          17
 6/09/64      PRIME      BSS     P           1
 6/09/64      PRIME   MADTAB     P           1
 6/09/64      PRIME      MAD     P           1
 6/08/64      MON04    SAVED     P          12
 6/08/64     PRIMES      MAD     P           1
 6/05/64     FILTER    SAVED     P          19
 5/12/64      GSUBA      BSS     P           3
 5/12/64      SCANA      BSS     P           3
 5/08/64      SCANT      BSS     P           2
R .616+.800


delete prime saved prime bss prime madtab primes mad mon04 saved
W 1122.9
R 3.200+.400

delete *bss"""" bss
W 1123.7
R 1.666+.200

listf
W 1123.9
     2 FILES     21 TRACKS USED
DATE              NAME          MODE    NO. TRACKS
 6/09/64      PRIME      MAD     P           1
 6/05/64     FILTER    SAVED     P          19
R .616+.616
```

FIG. 2-c. Print Out of Demonstration Session—Part c.

were given. The typing out of the prime numbers was interrupted by pressing twice the break button on the Teletype, which resulted in the system replying with the word QUIT.

The command *listf* without arguments causes the system to list the contents of the user's own private file directory as illustrated in Fig. 2e. There are four items with PRIME as first name: PRIME MAD, the original program shown in Fig. 2c, PRIME BSS, its translation in machine language, PRIME MADTAB, a table (storage map) useful for debugging purposes, and PRIME SAVED, the complete machine state after loading

```
copy p mon04 saved
W 1125.9
#R 2.616+.400

resume mon04 1
W 1126.1

 CTSS UP AT  902.8 06/09/64.

  NUSERS= 22    TIME= 1126.2
  29.7  29.7    BACKGROUND,
  46.6  46.6    FOREGROUND,
  17.6  17.6    SWAP TIME,
   6.2   6.2    LOAD TIME,
  15.5  15.5    USER WAIT,
   6.6   6.6    SWAP WAIT,
   3.9   3.9    LOAD WAIT.

  NUSERS= 23    TIME= 1127.2
  19.8  29.6    BACKGROUND,
  43.6  46.5    FOREGROUND,
  27.3  17.7    SWAP TIME,
   9.3   6.2    LOAD TIME,
  25.5  15.6    USER WAIT,
  10.3   6.7    SWAP WAIT,
   5.5   3.9    LOAD WAIT.
 QUIT,
R .000+2.616

delete mon04 saved
W 1127.9
R 1.000+.400


logout
W 1138.9
 T0193  2859 LOGGED OUT 06/09 /64   1139.1
 TOTAL TIME USED=   01.3 MIN.
```

FIG. 2-f. Print Out of Demonstration Session—Part f.

the program. The item named PRIMES MAD is a slightly different version of PRIME MAD.

Items are deleted from the file by issuing the *delete* command followed by the names of the items, as shown also in Fig. 2*e*. The command *delete*, with an argument consisting of an asterisk followed by a space followed by a name, causes all items with the stated name as second name to be deleted. The result of issuing the two *delete* commands in Fig. 2*e* was the elimination from the file of all items except PRIME MAD and FILTER SAVED, as indicated in the following printout of the file directory.

A program for monitoring the system operation, by the name of MONO4 SAVED, is available in the public file accessible to all system users. This program must first be copied into one's own private file by means of the *copy p* command, as shown in Fig. 2*f*. Since this program is stored as a record of a previous machine state, it is started by issuing the *resume* command. A second argument in the command (the digit 1) causes the program to be run at 1-minute intervals. At time 1126.2 there were 22 active users; 1 minute later there were 23. The other figures are percentages of the computer time devoted to various operations. FOREGROUND refers to the computer time devoted to running programs requested by on-line users. BACKGROUND refers to computer time available for running normal batch processing, which takes place automatically when no service is requested by on-line users. SWAP TIME is the processor time wasted in moving a program from core memory to drums and vica versa. LOAD TIME is the processor time devoted to loading programs from the disk files into core memory. The first four figures in each table add to 100 per cent (or approximately so). The last three figures refer to the times in which the processor is totally idle while input-output operations are taking place. The USER WAIT is part of the foreground, and it is already included in the FOREGROUND figure. Similarly, SWAP WAIT and LOAD WAIT are already included in SWAP TIME and LOAD TIME. The figures in the second column of each table are percentages evaluated from the time at which the system was last placed in operation (902.8 in Fig. 2*f*). The figures in the first column are percentages over the last minute interval; in the first table they are identical to those in the second column.

After deleting the monitor program from the file, the session was terminated by issuing the *logout* command. The session lasted approximately 1 hour, and was interrupted by two telephone calls lasting for a total of approximately 15 minutes. The total computer time used, swap and load time included, was 1.3 minutes.

The record of the demonstration session illustrates a few of the most basic commands of the MAC System. The total number of commands

available to all users is, at present, 68. This number is continually increasing as new facilities, developed by users as well as by the system programmers, are added to the system. In addition, a variety of programs of lesser general interest are available in the public file, from which they can be copied into private files. This same public file is also used to facilitate the transfer of programs and data between system users. Special facilities are also available for the operation of the display system. This includes facilities for displaying text, for drawing on the screen, for rotating three-dimensional objects, etc. The public part of the system consists at present of approximately a half-million words of code. The users' private files barely fit into the two disc storage units whose total capacity is 18,000,000 words.

OPERATING EXPERIENCE

The MAC System has been operating in roughly its present form since the middle of November, 1963. It is now in operation 24 hours a day, seven days a week. Maintenance, disk dumping and loading, and occasional nontime-sharing operation use approximately 4 hours per day. The on-line use of the system has steadily increased since November to April; the total number of computer hours charged to on-line users (the sum of the numbers printed out by the system on completion of each command) was 311 in April and 297 in May. In other words, the computer time devoted to serving on-line users amounted to approximately 42 per cent of total clock hours. The background use is not included in these figures. The total number of user-hours between logins and logouts turns out to be approximately 17 times the number of computer hours used.

The system is usually fully loaded (24 on-line users) during the day, and almost fully loaded in the evening until midnight and sometimes later. The system is very seldom idle even in the early morning hours.

Facilities for detailed monitoring of system operation have become available only very recently, and therefore no dependable data can be presented at this time. It must be stressed in this regard that it is far from obvious what measurements would provide a useful characterization of the system performance, in view of the many variables involved, and of the complexity of their interactions. Furthermore, the frequency and character of user requests vary with time, they are highly unpredictable, and of course cannot and should not be restricted or controlled in any realistic measurement of system performance.

The performance figure of greatest interest to the user is the response time (the time interval between the issue of a request and the completion

on the part of the computer of the requested task) as a function of the bare running time of the corresponding program. The response time depends on the scheduling algorithm employed by the system, as well as on the number and character of the requests issued by other users.

The scheduling algorithm operates roughly as follows. Each user request is assigned an initial priority which depends only on the size of the program that must be run. The highest priority is assigned to the smallest programs. The highest priority programs are allowed to run for a maximum of 4 seconds before being interrupted, while lower-priority programs are allowed to run for longer intervals that are multiples of 4 seconds. The lower the priority, the longer is the allowed interval. If a program run is not completed within the allowed interval, the program is transferred from core memory to drum (the state of the machine being automatically preserved), and its priority is reduced by 1 unit.

The curves in Fig. 3 illustrate the behavior of the wait time and swap time as a function of program run time for programs 50 words long and 25,000 words long. The swap time is defined for this purpose as the time spent in transferring the program between disk file and drum on the one hand, and core memory on the other; wait time is the time during which the computer is performing tasks that are not pertinent to that program. Obviously, the total response time is the sum of the bare run time (the abscissa in Fig. 3), the swap time, and the wait time. The points

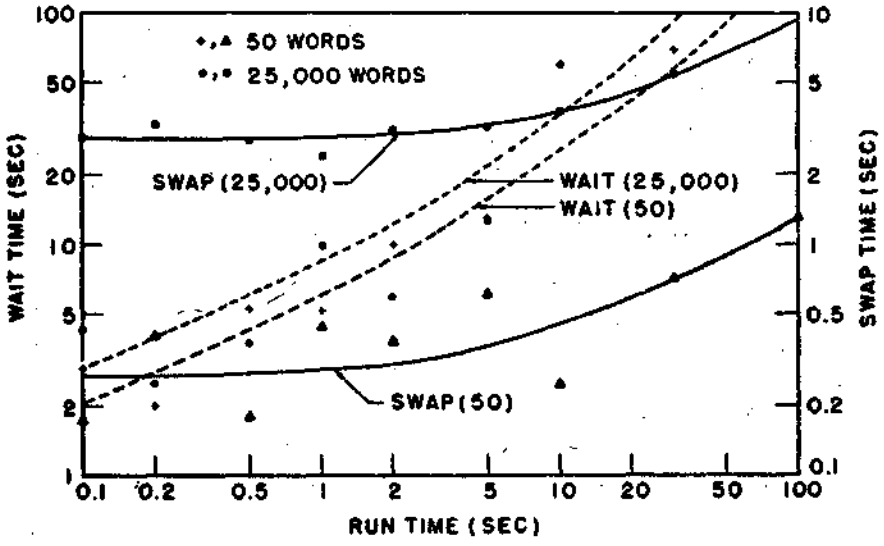

FIG. 3. Behavior of Wait Time and Swap Time as Functions of Run Time.

in Fig. 3 were obtained from measurements performed by M. Jones, and the curves are rough interpolations between the points. Each point represents the average value of 30 consecutive measurements. The number of users during the time that the measurements were being made varied from a low of 13 to a maximum of 21; it was between 17 and 20 most of the time. The scattering of the points in Fig. 3, together with the variability of the number of users while the measurements were being performed, should make clear the kind of difficulties one faces in obtaining precise and accurate measurements of system performance.

It is clear from Fig. 3 that the swap time constitutes a large overhead for short runs. It must be kept in mind in this regard that the swap time includes the initial transfer of the program from disk file to core memory and the final transfer back into the disk file; the two-way transfer of 32,000 words between disk file and core memory takes approximately 4 seconds, and the same transfer between drum and core memory takes approximately 2 seconds. The wait time increases less than linearly with run time, and, as expected, is not greatly affected by the size of the program.

The MAC computer installation has experienced a normal amount of hardware failures. On the other hand, it should be realized that hardware failures are much harder to diagnose in a multiple-access system because of the impossibility of reproducing the machine conditions at the time of failure. Furthermore, the probability distribution of machine states in a multiple-access operation is quite different from that in a batch-processing operation, and therefore hardware troubles become apparent in the former mode of operation that go unnoticed in the latter. Above all, it is often very difficult—much more than usual—to determine whether a particular malfunctioning is the result of a hardware failure, a system-programming error, or even an error in the logical design of the machine. There is no doubt that multiple-access systems present maintenance problems substantially more difficult than conventional systems.

The most delicate aspect of the operation of a multiple-access system of the MAC type is the responsibility assumed by the system managers with respect to the users' programs and data that are permanently stored in the disk files. Elaborate precautions must be taken to protect the contents of the disk files against malfunctioning of the system, as well as against actions of individual users. The supervisory program restricts the access of each user to his own private file, and to public files that he can not modify. A full copy of the contents of the disk file is made twice a day to minimize the loss in case of malfunctioning. While losses of users' programs and data have occurred, their frequency and seriousness have not discouraged users from entrusting their work to the system.

The system users number a little more than 200, with 10 academic departments represented among them. While most users had previous experience in programming, there is a growing group of users who are working entirely with programs developed by others, or through high-level problem-oriented languages that enable them to communicate with the system in an English-like language appropriate to their professional field. A discussion of the work being done with the system, and of the special language facilities[8-11] which are being developed for use in specialized fields is beyond the scope of this paper.

Enthusiasm mixed with a great deal of frustration is the most common reaction to the system on the part of its users. The system was very quickly accepted as a daily working tool, particularly by computer specialists. This quick acceptance, however, was accompanied by the kind of impatience with failures and shortcomings that is characteristic of customers of a public utility. The capacity of the system is limited, and therefore users are often unable to log in because the system is already fully loaded. Furthermore, the system may not be in operation because of equipment or programming failures, just at the time that one was planning to use it. In other words, the system is far from being as reliable and dependable as a utility should be. Yet, the experience since last November has shown that it is perfectly feasible for a computer system to be the object of research and development for some people and, simultaneously, an effective working tool for others. System experimentation and use are not only compatible but mutually beneficial.

It is becoming increasingly evident that the system's ability to provide the equivalent accessibility of a private computer is only a secondary, although necessary, characteristic. What users find most helpful is the fact that the system places literally at their fingertips a great variety of services for writing, debugging and compiling programs, and facilities for working on problems in their own fields through appropriate problem-oriented languages.

The system users themselves are beginning to contribute to the system in a substantial way by "publishing" their work in the form of new commands. As a matter of fact, an editorial board is being established to review such work and formally approve its inclusion in the system. Thus, the system is beginning to become the repository of the procedural and data knowledge of the community that it serves. A corollary of this trend is, of course, the increasing difficulty that users find in ascertaining what facilities of interest to them are included in the system. In other words, the system is beginning to have the undesirable as well as the desirable characteristics of a library.

SYSTEM TRENDS

Looking at the future, the organization of the MAC System appears to be at the threshold between two basically different points of view on computer systems and their utilization. The traditional view is that of a processor serving one user at a time and executing programs in succession, with a negligible amount of interaction during execution with the user himself or any other part of the outside world. A corollary of this view is that the processor, the memory, and the peripheral equipment must be designed to fit the requirements of the "typical user" rather than the average requirements of users as a group. Thus, the system as a whole can be used efficiently only by specifically tailoring programs to its peculiarities.

The present MAC System is still organized in a traditional manner in the sense that programs, whether public or private, are executed in succession as independent and indivisible entities. The fact that one program may be interrupted by a higher-priority program is irrelevant for the purposes of the present discussion. For instance, the execution of a system command generates a copy of the corresponding program (stored in the disk file), which is then run just as if it were a user program. Thus, if several users are simultaneously compiling programs written in the same language, several copies of the same complier are simultaneously and independently shuttling back and forth between core memory and drum. Another consequence is that any interactive program must be present in main memory in its entirety when data or instructions are needed from the user, even though the presence of one or two of its subroutines may be sufficient to accomplish the interaction.

A further and very serious aspect of the present mode of operation is that only one complete, executable user program can reside in core memory at any one time. This implies that the central processor must remain totally idle while a new program is being transferred into core memory or while necessary input-output operations are taking place. The idle time is very substantial in the present MAC System, as indicated by the "wait" percentages in Fig. 2*f*.

These system inadequacies are clearly the result of an organization keyed to the traditional point of view of a central processor executing independent programs one at a time. Furthermore, the characteristics of the present equipment would preclude in practice, if not in principle, any substantially different system organization. In order to overcome these limitations, one must approach the system design problem from a point of view considerably different from the traditional one.[12,13] We observe in this

regard even the term "time-sharing" is inappropriate and somewhat misleading because it emphasizes the necessary but secondary goal of providing the equivalent of a private computer. The term "multiple access" is also misleading when applied to the central processor. The emphasis should instead be on the system ability to provide convenient and flexible multiple access to an ever-changing structure of procedures and data capable of interacting as parts of distinct processes. In other words, it is the software structure which is important, and the hardware assumes the secondary role of providing convenient means for access to it.

If the system goal is to provide convenience of access to such a software structure, one is naturally led to view the system itself as memory centered rather than processor centered. Furthermore, the memory that forms the heart of the system would be not in the main core memory, but the bulk memory consisting of disk files or similar devices in which the procedures and data are normally stored. The main memory would play, instead, the role of a giant buffer matching the relatively low transfer rate of the bulk memory to the fast processing rate of processors and input-output channels. When the system is looked at from this point of view, it assumes the appearance more of a message-store-and-forward communication system, than of a traditional computer system. Indeed, the smooth flow of messages is of paramount importance to efficient operation, and the major function of the supervisory program is to organize the transfer of messages (procedures data, as well as input-output messages) in such a way as to avoid unnecessarily long queues, and to insure efficient utilization of the available equipment.

It is also clear that a computer system intended to serve simulaneously a large number of people must be organized in such a way as to avoid any unnecessary duplication of information in either main or bulk memory and unnecessary transfers between the two. This implies that procedures and data must be executable as common parts of processes simultaneously and independently initiated by different users. It also implies the possibility of executing processes involving several subroutines in such a manner that only the necessary subroutines are in core memory at any given time. The program segmentation scheme advocated by J. B. Dennis is keyed to these objectives.[13]

A corollary to this view of a computer system is the functional subdivision of the hardware into pools of equipment serving the same function, with each piece of equipment being duplicated to meet the average system demand. The point here is that, if the objective of the system is to provide convenient access to the procedures and data stored in the bulk memory, enough equipment must be provided to perform the necessary functions, so that a bottleneck in one part of the system does not prevent the full

utilization of other parts. The equipment itself can then be designed to match the average demand of users as a group, rather than the requirements of the "typical users." Furthermore, if each piece of equipment is duplicated within the system, one can envision achieving, through on-line maintenance, a level of reliability and continuity of operation which is unthinkable for the present MAC System.

In conclusion, the experience with the present MAC System suggests a trend toward memory-centered, as opposed to processor-centered, systems, including pools of bulk memories, core memories, central processors, and input-output channels, all communicating with one another, with the core memories acting as buffers. On the software side, the trend seems to be in the direction of executing processes consisting of many subroutines and data structures which are never assembled into a single program, and some of which may be common to other independent processes simultaneously in execution. This view of computer systems is indeed very different from the traditional one. Its implications are far from clear. Their exploration is a major objective in the development of the next MAC System.

ACKNOWLEDGMENTS

This paper reports on the accomplishments and ideas of a great many people associated with Project MAC, far too many to list individually. However, special credit is due to Professor F. J. Corbató, Deputy Director of the MIT Computation Center, and to his staff who developed the Compatible Time-Sharing System, and who are mainly responsible for its evolution into the present MAC System. It is a pleasure to be their spokesman, a pleasure mixed with fear of having failed to do full justice to their work.

The successful operation of a computer utility during its first year of existence is an accomplishment of which all those concerned with the system management and maintenance can be rightly proud. Their devotion to the goal of continuous operation has often extended far above and beyond the call of duty.

The financial support of the Advanced Research Projects Agency of the Department of Defense, and the managerial support of the Office of Naval Research are gratefully acknowledged. Their confidence and interest in Project MAC are a constant source of encouragement. In particular, I wish to express my personal gratitude to Dr. J. C. R. Licklider whose technical vision and contagious enthusiasm as Assistant Director of ARPA were responsible for the initiation of Project MAC.

Finally, I wish to express my gratitude to the I.B.M. Corporation, and in particular to Mr. Loren Bullock, its long-time representative at MIT, for their helpful cooperation in planning and implementing equipment modifications to meet the special needs of the MAC System. I am similarly grateful to the New England Telephone and Telegraph Company for its help and cooperation in engineering the Teletype network in the MAC System.

REFERENCES

1. McCarthy, J., "Time-Sharing Computer Systems," in M. Greenberger (ed.), *Management and the Computer of the Future* (Cambridge, Mass.: The MIT Press, 1962), pp. 221–256.
2. *The Compatible Time-Sharing System, A Programmer's Guide*, by the MIT Computation Center (Cambridge, Mass.: The MIT Press, 1963).
3. Corbató, F. J., M. M. Daggett, and R. C. Daley, "An Experimental Time-Sharing System," *AFIPS Conference Proceedings, Vol. 21* (1962), pp. 335–344.
4. Boilen, S., E. Fredkin, J. C. R. Licklider, and J. McCarthy, "A Time-Sharing Debugging System for a Small Computer," *AFIPS Conference Proceedings, Vol. 23* (1963), pp. 51–57.
5. Schwartz, J., "A General-Purpose Time-Sharing System," *AFIPS Conference Proceedings, Vol. 25* (1964), pp. 397–411.
6. Dennis, J. B., "A Multiuser Computation Facility for Education and Research," *Communications of the ACM, Vol. 7* (September 1964), pp. 521–529.
7. Stotz, R., "Man-Machine Console Facilities for Computer-Aided Design," *AFIPS Conference Proceedings, Vol. 23* (1963), pp. 323–328.
8. Ross, D. T., and C. G. Feldman, "Verbal and Graphical Language for the AED System: A Progress Report," Technical Report MAC-TR-4, M.I.T.
9. Biggs, J. M., and R. D. Logcher, "STRESS: A Problem-Oriented Language for Structural Engineering," Technical Report MAC-TR-6, M.I.T.
10. Weizenbaum, J., "OPL-1: An Open-Ended Programming System Within CTSS," Technical Report MAC-TR-7, M.I.T.
11. Greenberger, M., "The OPS-1 Manual," Technical Report MAC-TR-8, M.I.T.
12. Corbató, F. J., "System Requirements for Time-Shared Computers," Technical Report MAC-TR-3, M.I.T.
13. Dennis, J. B., "Program Structure in a Multi-Access Computer," Technical Report MAC-TR-11, M.I.T.