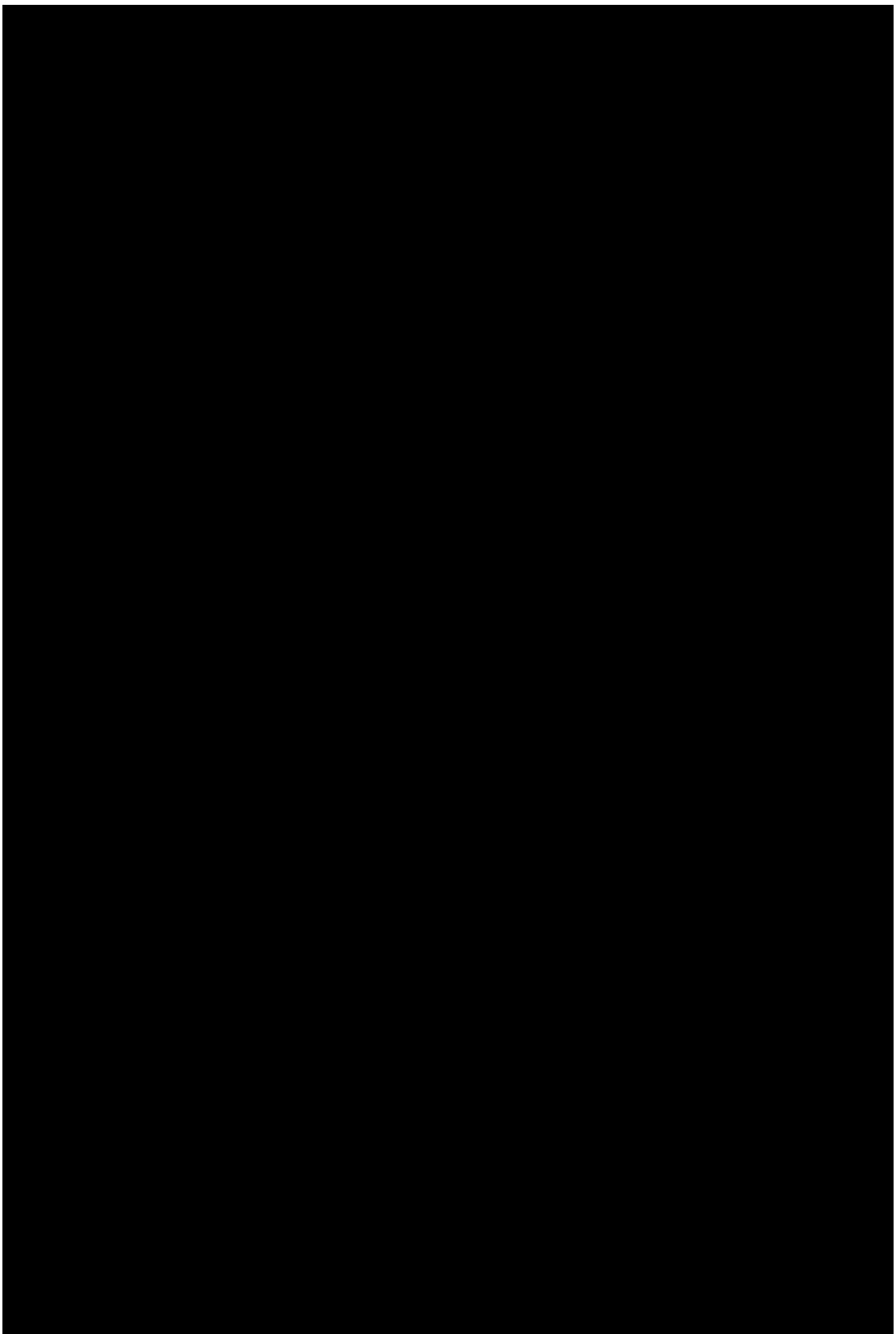


This blank page was inserted to preserve pagination.

MAC-TR-24

USE OF CTSS IN A TEACHING ENVIRONMENT

by

Daniel Roos

Department of Civil Engineering

Massachusetts Institute of Technology

November 1964

This empty page was substituted for a blank page in the original document.

ABSTRACT

Computer time-sharing offers many interesting possibilities for use in teaching computer technology. It might be expected that with proper hardware and software, students using time-sharing as a teaching machine could acquire proficiency in the fundamentals of programming more easily than using batch-processing.

To test this hypothesis, the M.I.T. Department of Civil Engineering divided a freshman programming class, so that half the students used batch-processing methods, and half used the Project MAC time-sharing system to do the same work.

This paper describes the experiment and its tentative results.

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

Massachusetts Institute of Technology

Project MAC

545 Technology Square

Cambridge, Massachusetts

02139

TABLE OF CONTENTS

SECTION		PAGE
	LIST OF TABLES	v
I.	INTRODUCTION	
	1.1 Description of 1.15	1
	1.2 Course Aims	2
	1.3 Computer Operations	2
	1.4 Course Meetings	2
	1.5 Student Enrollment	2
	1.6 Course 1.155	3
II.	TIME-SHARING EXPERIMENT	
	2.1 Mode of Operation	5
	2.2 Group Division	5
	2.3 Homework Assignments	5
III.	TIME-SHARING OPERATION	
	3.1 Remote Console Operation	7
	3.2 Time-Sharing Instruction	7
	3.3 Time Allocation	8
IV.	TIME-SHARING EXPERIMENTAL ANALYSIS	
	4.1 Overall Reaction	9
	4.2 Time-Sharing Orientation	9
	4.3 Console Reliability	10
	4.4 System Reliability	11
V.	STUDENT CONSOLE PERFORMANCE	
	5.1 General Observations	13
	5.2 INPUT Operation	13
	5.3 Checking for Errors before Compiling	14
	5.4 Compilation	14
	5.5 Debugging	15
	5.6 Consultation	17
	5.7 Student Maturity	17

TABLE OF CONTENTS (continued)

SECTION		PAGE
5.8	Allocation of Console Time	18
5.8.1	Assigned Hours	18
5.8.2	Signup Hours	18
5.8.3	Open Hours	18
5.9	Length of Sessions	19
5.10	Students per Console	19
5.11	Automatic Logouts	19
5.12	Response Time	19
5.13	Memory Requirements	20
5.14	Term Projects	20
5.15	Computer Time Requirements	20
5.16	Time Restrictions	25
5.17	Use of Time-Sharing in Lectures	26
VI.	CONCLUSIONS	27

LIST OF TABLES

Table		Page
1	Time Used Per Problem	21
2	Characteristics of Time-Sharing Group	22
3	Characteristics of 1620 Group	24
4	Comparison of 1.15 Final Grades	28

This empty page was substituted for a blank page in the original document.

SECTION I

INTRODUCTION

An experiment involving the Compatible Time-Sharing System (CTSS) was performed by the M.I.T. Department of Civil Engineering in conjunction with one spring term of Course 1.15, Computer Approaches to Engineering Problems. One half of the class performed all their work using an IBM 1620, while the other half used CTSS from a remote console. The purpose of the experiment was to evaluate the educational effectiveness and efficiency of using time-sharing, as compared to an IBM 1620 computer that is fully devoted to student use.

This report contains a description of Course 1.15, the experiment performed, and an analysis of experimental results. All opinions expressed are those of the author and do not necessarily reflect those of either the Department of Civil Engineering or of Project MAC.

1.1 DESCRIPTION OF 1.15

Course 1.15, Computer Approaches to Engineering Problems, is an introductory course offered by the Department of Civil Engineering. It is designed to prepare engineering students to use computers effectively in their educational programs.

The following subject areas are stressed in 1.15 :

1. The computer as an information-processing machine — hardware and software components — the composite machine;
2. Communicating with the computer — different types of programming languages are discussed, and while primary emphasis is on FORTRAN, machine language, assembly languages, and problem-oriented languages are also covered;
3. Use of computers in solving engineering problems — topics such as numerical methods, simulation, analog computers, and large scale computer systems are covered.

The specific topics presented vary from term to term, and are considerably influenced by the academic level of the students (freshmen, undergraduate, or graduate) and the interests of the instructor teaching the course.

1.2 COURSE AIMS

Although lecture material presented in 1.15 has varied, the intent of the course has always remained the same. It is intended that each student acquire proficiency in computer use by intensive first-hand computer experience. Each student programs from eight to 20 problems during the term. Students are encouraged to solve original problems that interest them or are applicable in other course work

or thesis work. No restrictions are made on the number of computer runs a student may make for each assigned problem. Inefficient and wasteful use of the computer is discouraged and has rarely been observed.

Course 1.15 is an educational experience, rather than a collected fixed body of knowledge which a student is quite likely to forget. The lectures are supplementary to the laboratory, rather than the conventional approach where laboratory sessions are supplementary to the lectures.

1.3 COMPUTER OPERATION

Following a normal 1.15 routine, students debug their assigned programs using the IBM 1620 computer laboratory of the Department of Civil Engineering, and then submit the programs to be run on the IBM 7094 at the M.I.T. Computation Center. The 1620 computer laboratory is open daily (including weekends) for student use from 1:00 to 10:00 P.M. The 1620 operation is described in the Computer Laboratory User Manual, which is available from the Department of Civil Engineering. The laboratory operates on a completely open-shop basis, with a 15-minute time limit imposed from 1:00 to 3:00 P.M., and a three-minute time limit between 3:00 and 10:00 P.M. This scheduling allows a student to make several runs during one computer session. As many as 300 student runs have been processed daily on the 1620 computer.

After a student has debugged his program on the 1620, he submits the program to be run on the 7094. These programs are taken to the M.I.T. Computation Center at 4:30 P.M., Monday through Friday, for overnight batch processing. Results are returned to the students the following day. This form of operation, using two computers, has been quite successful and is representative of many engineering organizations, where programs are developed and debugged on a small computer and then run on a large computer.

1.4 COURSE MEETINGS

The course is scheduled for two lectures, two laboratory-recitations, and two homework hours per week. Each lecture section consists of from 30 to 35 students and each laboratory-recitation section of 15 to 18 students. New material is presented primarily during lectures, while laboratory-recitation hours are devoted to a review of lecture topics, discussions of programming problems and homework assignments, and familiarization with computer equipment (use of Key Punch, 407, etc.). Laboratory-recitation sections are quite informal and are geared to individual interests of the students.

1.5 STUDENT ENROLLMENT

During the Spring Term 1964, 138 students registered for Course 1.15. There were four lecture sections and eight laboratory-recitation sections. Two of these lecture sections were for freshmen, and the other two sections were for

undergraduate and graduate students. One of the two freshmen sections was chosen for the time-sharing experiment.

1.6 COURSE 1.155

During the same term, Course, 1.155, Computer Laboratory, was begun. This course is taken in conjunction with 1.15 by any student wanting to do additional laboratory work. The course was specified in the catalog as having no lectures, five hours of laboratory per week, and no homework. Approximately 35 of the students registered for 1.15 registered for 1.155. Ten of the students in the experimental sections of 1.15 also registered for 1.155.

This empty page was substituted for a blank page in the original document.

SECTION II

TIME-SHARING EXPERIMENT

2.1 MODE OF OPERATION

The freshman section of Course 1.15 was divided into two similar groups of 16 students each. One group used the normal 1.15 mode of operation, where runs were debugged on the 1620 and then submitted to the 7094; the other group used CTSS, exclusively.

Both groups attended the same lectures and received the same programming assignments. Each group attended separate laboratory-recitation sections. No mention was made of specific computers during the lectures, whereas the laboratory-recitation sections were directed toward the particular machine the group was using. Occasionally, the framework was altered as a result of student questions or the topic covered. For example, the machine-language lectures were oriented towards the IBM 1620 machine.

2.2 GROUP DIVISION

To realistically evaluate the relative performance of the groups, it was necessary that the groups be similar. Therefore, division of the class was such that each group was required to:

1. have the same cumulative grade average from the Fall Term 1963,
2. have the same degree of previous computer experience,
3. have the same number of people registered for both 1.15 and 1.155.

Each group had 16 pupils: the 1620 control group had a class average of 3.64 (from 2.1 to 4.8), while the CTSS experimental group had a class average of 3.51 (from 1.2 to 4.8). Each group had five pupils who were also taking Course 1.155, and none of the 32 students had previous experience with computers.

2.3 HOMEWORK ASSIGNMENTS

Students in both sections were given identical homework assignments. Each student in 1.15 was expected to do a minimum of eight assigned problems. These problems were assigned at a rate of approximately one per week, beginning with the third week of the term. Students who were also registered in 1.155 were expected to do four additional problems and a term paper.

In addition, an introductory problem was assigned in the second week of the term. This problem (solution of a quadratic equation) was programmed during class time. The students in the 1620 group had to both punch cards and submit the program. The purpose of the assignment was to acquaint students with

operation of the 1620 computer and related off-line equipment. The time-sharing group used this problem to familiarize themselves with operation of the remote console and use of the time-sharing commands.

Since 1.15 was designed to prepare students for future work, primary emphasis was on the mechanics and application of FORTRAN. This is presently the principal computer language of the scientific and engineering communities. Therefore, the majority of assigned problems involved FORTRAN programming.

Assigned problems were also relatively short. We have found from past experience that students gain more from doing several short programs than from doing one long program.

The students in both groups were requested to keep detailed time statistics of all computer runs.

SECTION III

TIME-SHARING OPERATION

3.1 REMOTE CONSOLE OPERATION

A remote teletypewriter console (teletype number 38) was installed in Room 1-147 at the Department of Civil Engineering by Project MAC for the experiment. This teletype was reserved exclusively for use by students in the time-sharing group. These students were not permitted to use other consoles even if number 38 was inoperative. This restriction was imposed to help estimate the reliability of remote consoles.

3.2 TIME-SHARING INSTRUCTION

The first three weeks of the term were spent introducing the FORTRAN language and the philosophy and mechanics of time-sharing. These topics were presented concurrently, so that students would be able to use CTSS by the fourth week. Time-sharing material was presented only during the laboratory-recitation hours. Approximately three hours of lecture and discussion were needed to introduce the following topics:

1. Philosophy of time-sharing,
2. Basic time-sharing commands,
3. Running procedures and restrictions using CTSS,
4. Demonstration of CTSS.

By the end of the third week of the term, the students were familiar enough with time-sharing and FORTRAN to begin their first problem. Several of the more ambitious students were able to digest the material by the second and third weeks.

The following commands were presented, so that students could effectively use CTSS for FORTRAN programming:

LOGIN	LISTF	PRINTF	INPUT	EDIT	MADTRN
LOAD	START	SAVE	RESUME	DELETE	LOGOUT

A special handout was prepared which gave a detailed description of the above commands.

As the term progressed and more advanced material was introduced, new time-sharing commands, pertinent to the material, were presented. Additional handouts were prepared describing these commands. In addition, the CTSS Programmers Guide was available at all times at the remote console. Students were encouraged to experiment with any of the commands described in it, even those not presented in class.

3.3 TIME ALLOCATION

Different forms of time allocation were used to determine which is most effective for student use. These included assigned time, open time, and sign-up time. The console was available for use only during the following hours:

Monday - Friday, 9:30 A.M. to 4:30 P.M., and 8:30 P.M. to 10:00 P.M.

Saturday - Sunday, 1:00 P.M. to 4:30 P.M., and 8:30 P.M. to 10:00 P.M.

During the latter part of the term, it was necessary to relax these restrictions and permit runs after 10:00 P.M. This occurred because many students lost their normal session as a result of computer down time.

Each student was assigned two hours of console time per week, and any student also registered for 1.155 was assigned an additional two hours of console time. The assigned time was divided into sessions, varying from one-half hour to two hours. The length of sessions was purposely varied to determine the optimum length of a session.

In addition to the assigned time, six-and-one-half hours each week were classified as "open hours". The console was available on a first-come-first-served basis during these hours. If a student missed a part of his regular session because the computer was down, he was given first priority during open hours.

SECTION IV

TIME-SHARING EXPERIMENT ANALYSIS

4.1 OVERALL REACTION

At the beginning of the term, many students were skeptical about time-sharing and were somewhat annoyed that they would be unable to use the 1620. By the end of the term, their views had changed and they were extremely enthusiastic about time-sharing. Part of this enthusiasm may be due to the glamour and status of the experiment. Many of the students brought friends and dates to the console room to show off time-sharing. The major cause of the enthusiasm was, however, satisfaction with the time-sharing system. The following comments made by the students convey their overall favorable reaction.*

> "My overall reaction to the use of time-sharing was quite favorable."
>
> "Besides the 'thrill' of being in the experimental section, I enjoyed very much the personal satisfaction I received from having this direct line to a computer — a 7094 at that."
>
> "On the whole, I found the use of the time-sharing system to be very interesting and well worth the many instances when I was sure the purpose of it was principally to ruin the user."

4.2 TIME-SHARING ORIENTATION

The first month was the most difficult time during the experiment, from the standpoint of both the students and the instructors. During this period, many new foundations were built. Use of the console and the mechanics of time-sharing were added worries to students who were already puzzled over concepts such as the workings of a computer and the grammer of FORTRAN. One student said,

> "Besides trying to use a newly-learned language, which he is not very sure of, a student is faced with the problem of trying to utilize the intricacies of a system which often gives responses that he cannot understand. I for one remember losing the major part of a two-hour session simply because I couldn't decipher the 'PLEASE CORRECT IT OR TYPE IGNORE' command. Admittedly this problem could be solved simply by telling the student before he sat down at the console, but the point is that, if it is not this command it will be another, or perhaps an error diagnostic, that he will waste valuable time trying to interpret."

---

* Throughout the remainder of this report, excerpts will be quoted from reports submitted by the students.

Figures obtained from the students at the beginning of the term show that it took approximately 2.7 man hours, 1.6 console hours, and 2.3 machine minutes per student for initial orientation. This time was spent typing in, compiling, and running the pre-programmed problem (quadratic equation). These figures should be used with caution, since each student had a different conception of what "being acquainted with the system" meant, and man hours spent ranged from one to six hours, while console hours ranged from one-half to four hours.

Principal difficulties that students encountered during the first sessions were the following:

1. Use of the INPUT and EDIT commands, particularly use of the tab key. Students would try to type in commands while in the INPUT mode. The "PLEASE CORRECT IT OR TYPE IGNORE" statement was also quite confusing to several students.
2. Error messages were hard to decipher (PROTECTION MODE VIOLATION, etc.).
3. MADTRN diagnostics were non-existent or unclear.
4. Output, following a successful MADTRN compilation was confusing (such as the message THE FOLLOWING NAMES . . . .). Nowhere did the statement COMPILATION SUCCESSFUL appear.
5. Minor teletype difficulties were encountered.
6. Students were using commands they did not fully understand.

The last point deserves some comment. It is unreasonable to expect a student with no previous computer experience to have grasped the concept of loading, compilation, binary files, etc., after the first few weeks of the term. He, therefore, is quite confused by the sequence of commands: FILE, MADTRN, LOAD, and START. (LOADGO was not in operation when the experiment began.) The beginning programming student should be able to type in one command in the INPUT mode, which will perform all of the above operations if the program is correct. Later on, as students learn the internal operations of the machine and write more sophisticated problems, in different languages, they will be ready for the individual commands. During the first few weeks, however, they were only writing simple FORTRAN programs.

One interesting observation was made by the instructor. During the first few weeks, several students admitted they were quite confused; however, at the end of the term, these same students claimed they had no trouble mastering the system.

To summarize, the instructor's observations are: by the second week of time-sharing students were using the system, but not quite comprehending what it was all about; and by about the fourth to sixth week most of the students used CTSS quite proficiently.

4.3 CONSOLE RELIABILITY

Overall performance of the teletype unit was quite good. Analysis of data

furnished by the students shows that the teletype operated properly approximately 70 percent of the time. The two principal malfunctions while the system was operational were double letters and incorrect characters. Jammed keys and motor malfunctions were the principal reason for the teletype being inoperative. The principal complaint was that the typing rate was too slow. Several students expressed interest in IBM 1050 teletypewriter units, which appeared faster. One student was quite disappointed that he did not have access to a CALCOMP plotter, because the 1620 group were able to use the plotter that is connected on-line to the 1620.

One annoying problem was the delay encountered in getting the teletype fixed. This varied from one to several days, with the result that considerable rescheduling was necessary. Despite this, students seemed satisfied with the teletype as a minimum remote console. However, there is some doubt as to how effectively these beginning programmers could have utilized input/output devices which were part of a sophisticated remote console.

4.4 SYSTEM RELIABILITY

Unreliability of the system was a major problem. Student statistics reveal that the system was reliable only about 60 to 65 percent of the time. The two principal problems were system "down time" and "bugs" in CTSS.

The most bothersome "bugs" concerned:

1. MADTRN - Difficulties with the MADTRN command are discussed in the compilation section.
2. Storage of disk files - Several students lost files or received other users' files during the course of the term. Since the history tape procedure was not yet in full operation, it was difficult to retrieve lost files.

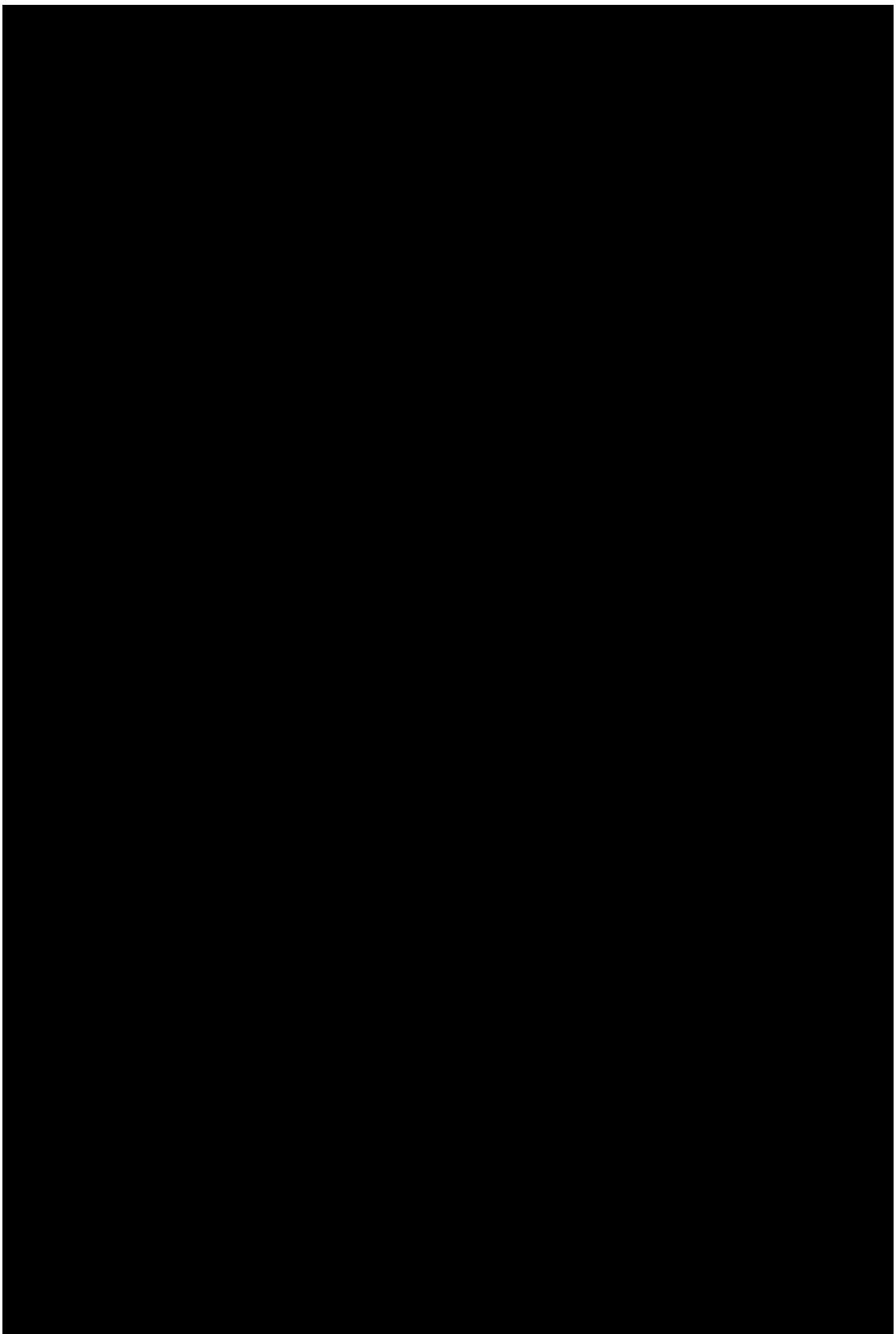
Computer down time caused major inconvenience for both students and instructor. Students would arrive for their time-sharing session only to find the machine down. Since there was rarely any estimate of when the machine would be up, a student would sit and wait for the hour or two he was scheduled. Generally, very little productive work could be accomplished while waiting. The net result was several wasted hours and much frustration. It must be remembered that the problems introduced by down time are quite different for a student than they are for a staff member. A student does not have a console in his office or close to his office, so that if the machine is down he cannot immediately return to other work. Unless consoles are installed in dormitories or fraternities, time-sharing does not offer students the same physical accessibility to the computer that it does to other users.

Several students suggested that messages be added to the comment CTSS NOT IN OPERATION, informing users when the machine would be back in operation. Those students who called the computer room requesting information about computer trouble always received very courteous service.

Computer down time introduced a major rescheduling problem. Homework due dates could be extended or makeup sessions assigned during the "open hours." Twice, homework assignments were delayed one week, with the result that two problems which were originally to be assigned (using FAP) were eliminated. A homework assignment was postponed only if a majority of the class had lost their sessions.

Reliability was fairly good until the middle of April. Two or three students lost time each week, but they were easily rescheduled. However, when a new time-sharing system was introduced, in April 1964, major problems resulted. There was one period when the system was down continuously for almost two weeks. Open hours were no longer sufficient to accommodate all the students needing makeup hours. To counteract the problem, time was assigned during the hours of 10:00 P.M. to midnight. As the situation worsened, time was assigned to 2:00 A.M. and even in one case until 4:00 A.M. These hours were only given to students who wished them and did not have morning classes the following day. No students were asked to work at an unreasonable hour. Several of the students preferred time-sharing after midnight because of the improved service, but in general we feel it unreasonable to expect students to work after midnight.

The experiment showed that sufficient open hours must be reserved in the event of machine down time. The instructor or an assistant must be willing to devote a considerable amount of time to rescheduling problems that continuously occur. Hopefully, all problems caused by the unreliability of the system will be eliminated as we gain more experience with time-sharing and design new systems.



5.3 CHECKING FOR ERRORS BEFORE COMPILING

Because of the accessibility of the computer, many students were quite sloppy about inputting their programs, and then performed no error checking before attempting compilation. Since they could compile immediately, they let the computer detect all their errors. One student said,

> "The system tends to make one try to run a program before it has been properly checked. This sometimes lets errors get by which will cause faulty results at the expense of computer time."

Almost no students used the PRINTF command before compilation to see if their program was entered correctly.

Of course, one could argue that the increased computer time brought about by eliminating human error checking is justified, because user work time is reduced. We will not try to answer this question here, but merely state that some proper balance should be obtained between work done by the student and work done by the computer. We will have more to say about this latter when the subject of running-time restrictions is discussed.

5.4 COMPILATION

The students wrote their programs in FORTRAN, and compilation was obtained using the MADTRN command. Any number of compilations and executions were permitted for each problem. The average number of compilations and executions per problem was between eight and fifteen, though some students had over 30 compilations for a single problem. Program compilations were attempted at the rate of three to four per hour, and the program execution rate was also about three to four per hour.

The principal reasons for the large number of compilations were: time-sharing accessibility offered, an absence of restrictions on the number of runs, and problems arising from the use of the MADTRN command. This command had the following serious drawbacks:

1. Diagnostic messages were difficult to interpret. It is essential that beginning programming students be provided with informative and correct diagnostics.
2. Certain errors were not detected by the translation. This was particularly true for errors in FORMAT statements. Since students commit many format errors, the absence of diagnostics was quite unfortunate.
3. Correct FORTRAN programs were incorrectly translated by MADTRN. The translation of COMPUTED GO TO statements was almost always incorrect. It is quite frustrating to write a correct program only to receive incorrect results because of a bug in the compiler.

Hopefully, these difficulties should be eliminated as soon as a FORTRAN compiler is introduced into CTSS. Until then, MADTRN can be viewed at best

as an only somewhat satisfactory temporary system.

FORTRAN has two principal drawbacks when used in a real-time system:

1. Diagnostic messages are not received until the entire program has been typed in and compilation requested.
2. Results are not obtained until the program has successfully compiled and execution has been requested.

For student use it would be desirable to take advantage of time-sharing and introduce editing routines and incremental compilers so that students would obtain immediate response. An editing program could be used in conjunction with the INPUT command, so that after a line is typed, the program would check for grammatical and some logical errors and print them out so the student could correct them immediately. A. L. Samuel's INPUT command is a step in the right direction. The attributes of incremental compilers have already been widely discussed, and experiments with COGO and J. Weizenbaum's language demonstrate the significant value of this type of approach. As one student suggested,

> "The constant man-machine communication possible on the console makes the console into a virtual teaching machine with instant reward and punishment."

New compilers should be developed to be used in conjunction with time-sharing which would increase this interaction.

5.5 DEBUGGING

The debugging operation consists of two phases; removing grammatical errors, and removing logical errors. Time-sharing was used quite effectively by students for the first type of operation. The grammatical errors were spotted quite readily and then immediately corrected. The students were unanimous in their enthusiasm for being immediately able to correct errors and rerun programs. Some students became a bit spoiled by the interaction time-sharing offered, and as a result, they had no real understanding of the usual turn-around-time problem.

The methods employed and the success obtained in removing logical errors from programs varied considerably among the different students. A student performed from 10 to 17 debugging runs per problem and the runs were made at the rate of about three to four per hour. The students in the 1620 group made only about three runs per problem. The greatest difference in the performance of the two groups was the number of runs per problem that each made.

The principal debugging techniques employed by the time-sharing group were:

1. Intermediate printouts;

2. Variation of input data and observation of effect on answers;
3. Examination of MADTAB files, to see if variables were properly defined and to obtain the machine locations of variables for use with the PM command;
4. Examination of MAD file, to check out intermediate MAD programs;
5. Use of the PM command, for a post-mortem printout of the program's condition in core.

Methods 1 and 2 were used almost exclusively by the poorer students, whereas all five methods were used by the better students. Debugging methods improved during the second half of the term when machine language, assembly programs, and internal machine structure was stressed. Many students missed the availability at the console of a FAP listing of their FORTRAN programs.

In general, the debugging techniques used by the students were considered mediocre. One student commented,

> "Assuming CTSS in operation, the major disadvantage is the difficulty in finding a logical error while sitting at the console."

This statement is ironical, considering that one of the goals of time-sharing is to permit users to discover errors using man-machine interaction. The student, instead of viewing time-sharing as a debugging aid, looks upon it as a disadvantage. The inability of many of the students to adequately use time-sharing for debugging raises many important questions. Can the relatively inexperienced programmer adequately use the somewhat sophisticated debugging techniques that time-sharing offers? Many of the students even had trouble with the PM command. They did not know how to interpret the output they received. It seems apparent that we need numerous simple and non-machine-oriented debugging aids for the novice programmer to use with time-sharing.

The principal difficulties students encountered related to input/output. Format statements caused many problems. Therefore, it would be desirable to provide the beginning programmer with processors that have very simple input/out conventions. Format-free input would be especially desirable.

Because many students constantly got bogged down with a problem in the middle of a session, the usual procedure was to work on several problems in the same session. As soon as a student could go no further with one problem, he would begin the second one. This is a far superior plan of attack than to blindly type in commands in an attempt to discover errors. The latter approach, unfortunately, was employed by several students.

The question of too much accessibility to the machine is again raised with respect to debugging. One student commented,

> "The main problem with debugging is that the machine is your servant for two hours a week - it is difficult to convince oneself that taking a hard look at the program may be more fruitful than giving command after command, in hopes that the error will make itself obvious."

Although debugging performance was only fair during the term, by the end of the term all students were fairly proficient at debugging. Since all students in the other 1.15 sections did not have the same proficiency, it seems reasonable to conclude that time-sharing had helped the student gain greater insight into the computer. It is possible that the unproductive sessions experienced by many of the students during the early months were a necessary evil to reach the final objective.

It was also obvious, from observing debugging performance at the console, that the value of time-sharing varies with the capabilities and experience of each student.

5.6 CONSULTATION

Two lab instructors were available throughout the day for consultation. Although the instructors' offices were less than one minute from the console, students never consulted the instructors during their time-sharing sessions. If, however, the instructor was in the time-sharing room, he was flooded with questions, many of which could have been answered by the student with a moment's thought. Here is another example of the student's reluctance to surrender the console.

It would appear from this discussion that it would perhaps be unwise to have an instructor on duty in the time-sharing room. On the other hand, there are many times, especially during the first time-sharing sessions, where the student really needs assistance. A brief comment by the instructor as to the meaning of an error message could save him hours of useless work.

A compromise would seem to be the answer. Student or teaching assistants could serve as consultants in the console room, and supply limited information. The student would consult his regular instructor for major debugging problems.

Many students asked questions of staff members, not in any way connected with 1.15, who were using other time-sharing consoles in the same room. These people were more than willing to help, but they were distracted from their own work. It should be decided whether educational consoles and research consoles should be located in the same room.

5.7 STUDENT MATURITY

For the most part, students demonstrated extreme maturity when using the experiment. They followed all rules and regulations and did not try to fool around with CTSS. Two types of situations arose where students deviated from the usual procedure. A student used the system from 8:00 P.M. Friday to 4:00 A.M. Saturday, even though he was only scheduled for a two-hour session from 8:00 P.M. to 10:00 P.M. This same student and his roommate, who was also taking the course, constantly tried to use CTSS during unauthorized hours. It is rather difficult to be too critical of such overly optimistic and eager students.

Two other students used slightly off-color expressions when naming their disk files and for printouts during their runs. The students used these comments even though they were well aware that their output would be carefully examined.

Both of these situations, although somewhat immature, should not overshadow the overall mature approach of the students. These students could be trusted with the time-sharing system.

5.8 ALLOCATION OF CONSOLE TIME

Console hours were allocated on the basis of assigned hours, signup hours, and open hours. The first two systems worked well, but the third did not. Operation under each system is described below.

5.8.1 Assigned Hours

This was the principal means of allocating time. Students planned their work beforehand, so they would have sufficient material to keep them busy for their one- or two-hour session. The only undesirable effect noted was pressure on the students to use the console constantly during their assigned hours, even if they were not sure what to do.

5.8.2 Signup Hours

With the signup hours, a student could vary time-sharing sessions from week to week, to fit into his overall schedule. In most cases, however, students did not feel there was any real advantage of signup hours over assigned hours. During the term, most students assume a fairly stable schedule, so that devoting the same two hours a week to time-sharing is not a real hardship. This method of scheduling has the major disadvantage that a signup schedule must be maintained from week to week.

5.8.3 Open Hours

With only one console available, the open-hours system was a failure. Many students would travel from 10 to 45 minutes to get a time-sharing session during an open hour, only to find that the console was already in use by some other student. If sufficient consoles were available, this situation would not have arisen. The open-hours system has the desirable advantage that a student can time-share when he needs to; he does not have to commit himself beforehand for a fixed amount of time, and he is not under pressure to produce something of value at the console during each fixed time period.

The open-hours system does, however, produce an unbalanced load on the computer. During desirable hours, many students would be time-sharing; whereas during other hours, the consoles would be vacant and the computer load

fairly light. This situation currently exists with respect to the key-punch machines in Building 26. With assigned time or signup hours for using CTSS, the computer load could be evenly spread throughout all hours.

Although the open-hours system appears attractive, if there are sufficient consoles, the assigned-time system and the signup system are more realistic and economic.

5.9 LENGTH OF SESSIONS

The students were unanimous in the following views on the length of time-sharing sessions:

1. Little can be accomplished in sessions less than one-half hour;
2. Sessions longer than two hours were generally unproductive after the second hour, unless some larger program, such as a term project, was involved;
3. Two one-hour sessions per week were superior to one two-hour session per week;
4. The optimum session length would be about one hour (45 minutes to 1-1/2 hours).

5.10 STUDENTS PER CONSOLE

One console was more than adequate to fill the demands of 16 students in Course 1.15 and the 6 students in Course 1.155. The console could handle from 25 to 30 students if the following conditions are assumed:

1. Each student is allotted two hours of console time per week;
2. Assigned console time is eight hours for Monday through Friday and six hours for Saturday and Sunday;
3. One to two hours per day are available for makeup sessions.

If sessions as late as midnight are scheduled, and minimal time is reserved for makeup sessions, then as many as 40 students could be accommodated on a single console.

5.11 AUTOMATIC LOGOUTS

During the semester there were a total 66 automatic logouts. Approximately ten of these occurred at 4:30 P.M. Most students found the automatic logouts annoying, but because of their low frequency of occurrence, they were tolerable.

5.12 RESPONSE TIME

Average response time was estimated by the students to be from 10 to 15 seconds. This delay was considered quite reasonable. The majority of students

considered delays greater than 30 seconds unreasonable, and in some cases intolerable. As the students became more familiar with the operations necessary to execute each command, their response-time requirements were adjusted. For example, they expected response time for the MADTRN command to be considerably longer than for LISTF.

Two qualifications should be associated with the figures given above. First, it is extremely difficult for untrained persons to estimate time, so that a ten-second estimate by a student might, in reality, be only five seconds. Second, a student's response requirements are greatly influenced by the average and exceptional performance of the system. Once a student sees that MADTRN can execute in ten seconds, real-time, he expects it to always execute in ten seconds. Therefore, it seems reasonable that students will always be somewhat dissatisfied with response times, unless they are virtually instantaneous.

5.13 MEMORY REQUIREMENTS

Each student was allotted 50 tracks on the disk file. He was expected to delete files when they were no longer needed, and use the space as efficiently as possible. The students agreed that fifty tracks were more than sufficient. The only exceptions were several students who did rather lengthy term projects. Most students felt that they could have performed quite satisfactorily with only 20 to 30 tracks.

5.14 TERM PROJECTS

Time-sharing was used quite effectively by all students who did term projects for Course 1.155. These projects involved rather long, difficult programs that required considerable debugging and testing. Time-sharing was ideal for these operations.

Term projects performed were:

1. Simulation of the 1620 on the 7094 (done by two students),
2. A program to play dominoes,
3. Searching and sorting techniques,
4. Solutions to simultaneous equations,
5. Polynomial Curve Fitting.

It was quite evident that the students enrolled in both 1.15 and 1.155 made far better use of time-sharing than those students enrolled only in 1.15. This seems to support the contention that as a programmer becomes more experienced, he makes better use of time-sharing.

5.15 COMPUTER TIME REQUIREMENTS

Average console and machine time used for problems is shown in Table 1.

Table 1

Time Used Per Problem

Time-Sharing Section

Problem	Console Time (hours)	Machine Time (min.)	Programming Time (hours)	Total Work (hrs.)	Time Span (days)
a) 1.15 Problems (listed in order assigned.)					
Quadratic Equation	2.6	4.26	1.	3.3	6.3
Triangle	2.4	3.8	2.1	5.2	9.5
Elementary Operations	1.79	2.94	1.49	3.28	6.85
Format	4.75	10.4	4.5	9.9	15.
Change	2.	3.6	1.32	3.53	12.
Matrix Multiplication	2.57	5.1	1.65	4.55	7.1
Information Retrieval	4.25	8.85	1.8	6.9	11.
COGO (No. 1)*	.25	.95	.23	.7	1.
COGO (No. 2)*	.74	.16	.6	1.26	2.8
COGO (No. 3)*	1.25	3.	3.	4.25	7.
Simulation	2.5	2.97	1.2	3.25	4.1
Average	2.62	4.81	1.82	4.66	9.18
b) 1.155 Problems					
Straight Line Fit	3.	4.6	.75	3.75	10.
Azimuth	2.03	3.9	1.25	3.25	6.8
Soil	2.1	4.1	1.9	3.8	22.
Terminal Velocity	.75	2.35	1.9	3.	7.
Inverse Hyperbolic					
SIN	1.05	2.7	.75	2.	22.3
Root	1.13	2.2	.38	1.76	1.3
Average	1.68	3.31	1.155	2.93	11.57

*Students were only required to choose one of three COGO problems. All three COGO problems counted as a single problem in the calculation of average values.

Table 2

Characteristics of Time - Sharing Group

Student	1.55 Participant	February Console	February Machine	March Console	March Machine	April Console	April Machine	May Console	May Machine	Total Hours Console	Total Hours Machine
Carlson	no	.76	.01	9.91	.45	5.97	.26	2.99	.12	19.23	.84
Christiansen	no	1.75	.06	10.70	.59	10.55	.54	5.88	.29	28.88	1.48
Friedman*	yes	.61	.02	21.04	2.33	1.12	.03	.00	.00	22.77	2.38
Gottlieb*	yes	7.48	.53	29.49	1.69	26.40	.95	81.78	3.71	145.15	6.88
Jeffrey	no	.00	.00	4.17	.13	7.62	.39	6.80	.30	18.59	.82
Jones	no	1.39	.02	7.05	.19	10.87	.43	9.51	.46	28.82	1.1
Kern	no	.62	.02	7.37	.27	5.34	.23	6.90	.34	20.23	.86
LaBreche	no	.00	.00	5.58	.17	9.46	.30	10.22	.49	25.26	.96
Lemer	no	.98	.02	4.46	.11	6.15	.18	4.66	.21	16.25	.52
Radzikowski	yes	4.99	.15	16.82	.51	11.30	.35	20.78	.68	53.89	1.69
Reinert	no	1.75	.04	4.17	.12	3.84	.06	7.88	.21	17.64	.43
Simon	no	1.35	.03	2.09	.07	2.73	.15	7.27	.28	13.44	.53
Steinmetz	no	3.97	.12	13.89	.48	10.08	.53	20.58	1.53	48.52	2.66
Stuntz	yes	2.27	.06	10.16	.41	11.94	.47	24.45	1.28	48.82	2.22
Weiner	no	.39	.01	19.48	.78	18.28	1.17	4.65	.19	42.80	2.15
Willis	yes	3.61	.19	27.69	1.11	13.26	.53	11.65	.50	56.21	2.33
Total Hours		31.92	1.28	194.07	9.41	154.91	6.57	226.00	10.59	606.90	27.85
Average		1.99	.08	12.1	.59	9.7	.41	14.1	.66	37.93	1.74

*Since these students worked on the same term project they logged in using one number (Gottlieb) during most of April and all of May.

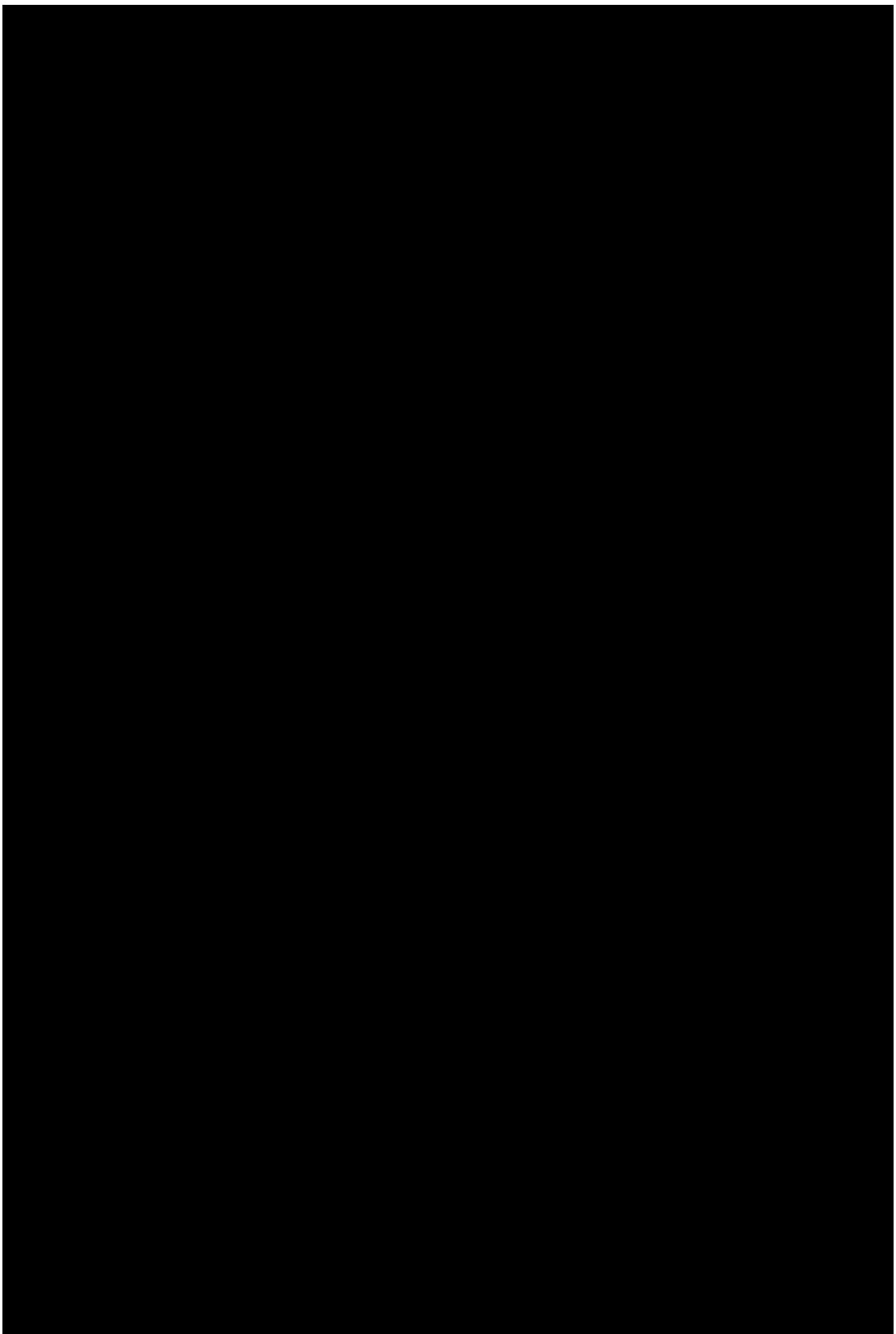


Table 3

Characteristics of 1620 Group

Problem	Number of Runs	Punch Time (min.)	Initial Program Time (hours)	Machine Time (min.)	Wait Time (min.)	Total Work Time (hours)
a) 1.15 Problems						
Triangle	1.5	64.1	1.78	5.9	28.8	3.72
Elementary Op.	3.29	42.4	1.91	7.67	72.6	4.9
Format	3.78	68.7	2.55	9.67	66.3	6.22
Change	4.	33.6	.86	8.85	37.8	2.13
Matrix Mult.	4.6	62.1	2.54	11.9	40.	5.2
Info. Retrieval	2.75	62.	2.8	8.6	32.5	5.72
COGO	2.4	43.	1.6	4.8	15.3	3.27
Simulation	3.4	50.	1.15	11.2	31.4	3.6
Average	3.21	53.2	1.9	9.5	40.6	4.33
b) 1.155 Problems						
St. Line Fit.	1.67	30.	1.18	2.5	3.	1.6
Soil Classif.	2.	27.5	.66	3.6	20.	1.25
Simultaneous Eqs.	6.	67.5	1.75	10.5	50.	3.62
Series Sum.	4.5	48.	1.1	9.	32.5	2.9
Median	3.67	40.3	1.74	6.1	56.1	3.5
Machine Lang.	1.83	21.4	1.33	2.78	42.5	2.88
Average	3.28	39.1	1.26	5.75	34.	2.63

The figure of 1.74 hours of 7094 time per student is too high. If each 1.155 student is given double weight, since he was enrolled for two courses, then the figure becomes 1.33 hours per student. This is still considerably higher than the figure of 0.25 hours per student per term which is used by the M.I.T. Computation Center in allotting 7094 time to courses 6.41 and 6.251. However, the following factors should be considered:

1. 1.15 students programmed twice as many problems as 6.41 and 6.251 students;
2. Much of the time used by the 1.15 students was unproductive memory swap time.

If we assume that only 50 percent of the time used by the students was for productive work, then each student used 0.67 hours. This time does not compare too badly on a per problem basis with the times for 6.41 and 6.251. There is no question, however, that each student in 1.15 did use a considerable amount of computer time.

5.16 TIME RESTRICTIONS

The CTSS section of the class was unanimous in their opposition to any restrictions on the number of runs per problem or computer time per problem. The philosophy of 1.15 has always been to allow the students freedom to use the computer. The course has always used a small-scale computer (650, 1620), so that operations without restrictions with respect to computer time were feasible. It is questionable whether this same philosophy could be used with respect to a 7094-oriented course. Several students used over 30 minutes of 7094 time in the solution of a single problem. This is ridiculously expensive.

Another approach is represented by courses 6.41 and 6.251, where the student is assigned a limited number of problems, graded on each run and allowed a maximum number of runs (usually four) per problem. The students seemed quite opposed to this framework. They claim that a student turns himself into a computer and spends hours checking over his program for minute errors. One student who had already taken 6.41 commented,

> "I found myself (in 1.15) using considerably less time on programming, with time-sharing, than I spent in 6.41; where a person had to spend hours meticulously checking the program for errors, hoping to get a high mark by having it run on the first try."

The whole concept of time-sharing is opposed to the type of operation imposed by restricted runs.

The solution is probably some compromise between the 1.15 approach and the 6.41 - 6.251 approach. To give the student complete freedom with no restrictions would have two undesirable effects:

1. The student would use the system carelessly and waste valuable computer time;

2. The student would be educated unrealistically, since he would assume that he would always have unlimited free machine time, and would not realize the proper balance between computer and human time.

On the other hand, the restrictions should be minimized so that the student can gain full benefit from time-sharing. One approach would be to put a computer time restriction on each student for the entire term, and allow him to divide that time between problems (with guidance) in the best possible way. This would place a high degree of responsibility on the student.

5.17 USE OF TIME-SHARING IN LECTURES

Three lectures were given in which time-sharing was used in conjunction with closed-circuit television. These lectures were presented in the Civil Engineering Experimental Computer Classroom (Room 1-150). The topics and material presented are summarized below:

1. Format statements - as various specifications were discussed, data was typed in by the instructor and the resulting internal representation was typed back by the computer;
2. COGO - The COGO language was presented by typing in a demonstration problem on the remote console;
3. Simulation - the use of simulation was demonstrated using time-sharing.

For Lectures 1 and 3, special notes were prepared for the students, which explained the problems and contained space for them to write down the results of each experiment.

The use of time-sharing during the lectures was overwhelmingly successful. Student interest and participation was high. During the simulation and COGO lectures, students kept suggesting problems they wanted to try. Because of computer accessibility, it was possible to immediately try their problems on the computer. The lecture on format statements was particularly successful, since the use of format has always caused considerable difficulty. Time-sharing gave the lecture new meaning, and allowed the students to see exactly what was being described.

Time-sharing should have tremendous potential in the classroom, using television and a console. The teacher would be able to use the computer whenever necessary to demonstrate salient points. The Department of Civil Engineering is now actively engaged in using time-sharing in many of its courses. This work is being conducted using the Computation Center computer in conjunction with a Ford Foundation research grant.

SECTION VI

CONCLUSIONS

Many respected educators at the Institute have been concerned that the "glamour" of freshmen computer courses has an undesirable effect. The students spend all their time on computer work, while neglecting their important courses, such as 5.01, 8.01, and 18.01. This danger was increased for the time-sharing section, where the student was presented with the additional glamour of a time-sharing console.

The experiment seems to indicate that such fears did not materialize. The student's grades are shown in Table 4. The cumulative average of the time-sharing class was 3.50 during the first term and 3.68 during the second term. In the 1620 group, the average dropped from 3.65 to 3.48.

Many students felt that time-sharing encouraged their work in the other courses. Several students even wrote programs for use in their physics laboratory sessions. And the time-sharing sessions provided a good balance to counteract the usual home work of freshman basic core courses.

The results of the experiment have not conclusively demonstrated that time-sharing is superior to the 1620. In fact, it could be argued that time-sharing was inferior, because of the many runs students in the time-sharing section required and the large amount of machine time they used. The authors, however, feels the experiment did illustrate the great potential of time-sharing in an educational framework.

The experiment was conducted under extremely adverse conditions: The time-sharing system used is centered around a computer that is not designed for time-sharing; the majority of machine time a person uses is for unproductive purposes; the software being used is a carryover from the batch-processing days; and, at the time, the system had many problems and was frequently inoperative. Even with all these disadvantages, students were able to use the system quite effectively. One can imagine the results of the experiment if the students had proper hardware and software to work with.

The use of time-sharing in the lectures was quite encouraging. This concept could be expanded, and time-sharing could be used for lectures in engineering and science courses. Demonstrations could be run on the console, with the results displayed via closed-circuit television.

The following observations were made as a result of the time-sharing experiment:

1. The overall reaction of the students to time-sharing was very favorable.
2. Each student in the time-sharing section used an average of 37.9 console hours and 1.74 machine hours during the course of the term.

Table 4

Comparison of 1.15 Final Grades

a. Time-Sharing Section Grade Data

Name	Term Average Fall 1963	Term Average Spring 1964	1.15 Grade	1.155 Grade
Carlson	4.8	4.5	A	
Christiansen	4.5	4.4	B	
Friedman	4.3	4.2	A	A
Gottlieb	4.3	4.4	A	A
Jeffrey	3.8	3.3	B	
Jones	3.3	4.4	B	
Kern	3.6	3.7	B	
LaBreche	2.9	3.7	A	
Lemer	3.0	3.5	B	
Radzikowski	3.0	2.7	A	A
Reinert	2.2	2.1	D	
Simon	1.2	2.5	C	
Steinmetz	4.8	4.7	A	A
Stuntz	3.2	3.1	B	
Weiner	3.7	3.7	A	
Willis	3.6	3.9	B	A
Total	56.2	58.8		
Average	3.51	3.67		

b. 1620 Section Grade Data

Name	Term Rating Fall 1963	Term Rating Spring 1964	1.15 Grade	1.155 Grade
Ascherman	3.2	3.4	A	A
Baker	2.7	2.0	C	
Desmond	3.9	4.4	A	
Ebert	4.6	4.4	A	
Flanagan	4.8	4.4	A	A
Gips	3.8	3.9	A	
Hart	3.4	2.7	C	
Haussling	4.2	4.5	A	B
Meads	3.8	3.9	B	B
Miller	2.1	0.9	F	
Nainis	4.1	3.6	C	
Robinson	3.5	3.7	D	
Schoenwald	4.0	3.8	B	
Sikes	3.3	2.7	B	D
Taylor	3.9	4.2	A	
Wanek	3.0	3.2	B	
Total	58.3	55.7		
Average	3.64	3.48		

One to three hours of console time, and about four minutes of machine time, were needed for the solution of each problem.

3. In the 1620 group each problem required seven minutes of computer time.
4. The 1620 group required far fewer runs per problem (four vs. nine) than did the time-sharing group.
5. One remote teletype console was able to adequately handle the demands of 16 students.
6. Assigning specific console hours for each student worked quite well. Time-sharing sessions about one hour long are optimum for students.
7. Student users of time-sharing have different requirements than research users. A student runs one-shot programs, spends considerable time inputting and has a high ratio of console to machine time.
8. The effectiveness of time-sharing varied in direct proportion to a student's ability. As a student gained computer and programming experience he used the system more effectively.

It may have been a mistake to perform the experiment so soon after time-sharing was introduced. The time-sharing system has improved so markedly, that if the experiment were to be repeated now the results would probably be quite different. The overall conclusion is that time-sharing should be used by students when adequate software and hardware is available, and when its use can be economically justified.

This empty page was substituted for a blank page in the original document.

CS-TR Scanning Project
Document Control Form

Date : 12/11/95

Report # LCS-TR-14

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 36 (41 - IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): FOLLOWS TITLE, V, 3, 29

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:

IMAGE MAP: (1-36) UNNUM'ED TITLE & BLANK PAGE, iii - V,
UNNUM'ED BLANK, 1-3, UNNUM BLANK, 5-29, UNNUM BLANK.
(37-41) SCANCONTROL, COVER, TRGT'S (3)

Scanning Agent Signoff:

Date Received: 12/11/95 Date Scanned: 1/19/96 Date Returned: 1/26/96

Scanning Agent Signature: Michael N. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 1/19/1996

M.I.T. Libraries
Document Services