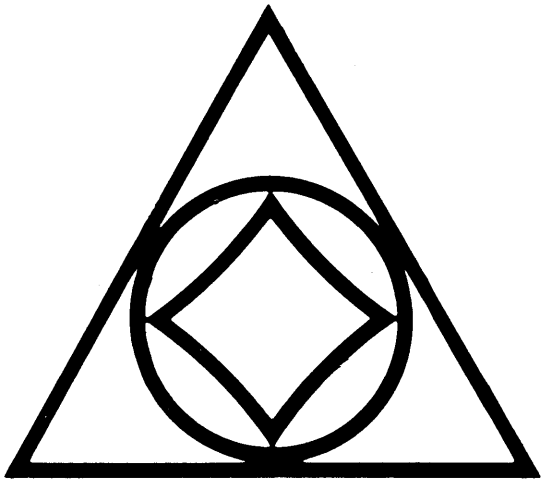


AFIPS

CONFERENCE
PROCEEDINGS

Computers:

THEIR IMPACT ON SOCIETY

(Volume 27, Part 2
1965 Fall Joint
Computer Conference)

1967

THOMPSON BOOK COMPANY
Washington, D.C.

ACADEMIC PRESS, INC.
London

The ideas and opinions expressed herein are solely those of the authors and are not necessarily representative of or endorsed by the 1965 Fall Joint Computer Conference Committee or the American Federation of Information Processing Societies.

Library of Congress Catalog Card Number 55-44701
Thompson Book Company
383 National Press Building
Washington, D. C.

Sole distributors throughout the world, with the exception of North and South America:

Academic Press, Inc.
Berkeley Square
London, W 1, England

Contents

Foreword	v	
Preface	vii	
Conference Committees	ix	
Harry Goode Memorial Award	xi	
The mighty man-computer team	1	*I. Rhodes*
The computer and our changing society	5	*S. Ramo*
Computers and education	11	*R. Gerard*
Computers: The physical sciences and medicine	17	*J. Maloney, Jr.*
Impact of computers on retailing	21	*C. McBrier*
The application of computers to domestic and international trade	27	*W. Merkin and R. Long*
The role of computers in space exploration	33	*C. Gates and W. Pickering*
The impact of computers on the government	37	*J. Ward*
Communications, computers and people	45	*P. Baran*
The impact of computers on urban transportation	51	*K. Schlager*
PANEL DISCUSSION RECORDS		
The computer industry in the buyer's market	59	*J. Ricca, J. Eckert, W. Gallagher, R. Hubner* and *R. Schmidt*
The overseas computer market	63	*M. Mapes, Jr., D. Orr, J. Miles, N. Ream* and *T. Thau*
The future of electromechanical mass storage	77	*W. Farrand, R. Franklin, N. Hardy, R. Graham, M. Eyster, W. Broderick, F. Lohan, D. Sampson, A. Shugart, I. Wieselman*
Promising avenues for computer research	85	*R. Rice, K. Uncapher, T. Steel* and *L. Hobbs*
LATE PAPERS NOT APPPEARING IN VOLUME 27, PART I		
A high speed thin-film memory: Its design and development	103	*Q. Simkins*
Efficiency and management of a computing system	107	*H. Huskey*
Use of digital computers in basic mathematic courses	111	*W. Marsland, Jr.*
ERRATA: Final Version of Papers Appearing in Preliminary Draft Form in Part 1, Volume 27		
MATHLAB: A program for on-line machine assistance in symbolic computations	117	*C. Engelman*
A time-and memory-sharing executive program for quick response, on-line applications	127	*J. Forgie*
Interactive machine-language programming	141	*B. Lampson*
An integrated computer system for engineering problem solving	151	*D. Roos*
ERRATA TO VOLUME 27, PART I	163	

Foreword

This second volume of the Proceedings of the 1965 Fall Joint Computer Conference is an attempt to capture the spirit of innovation that was the hallmark of the Conference held in Las Vegas, Nevada, November 30 - December 1, 1965. Part I of Volume 27, containing the formal papers presented in the Technical Sessions, was distributed at the Conference. The first volume included the traditional papers covering recent advances in hardware and software as well as a number of presentations focusing attention on new applications and problems of management in information processing.

From its inception, the 1965 FJCC attempted to provide the opportunity for professional communication at every level. To this end the conference included a number of panel sessions organized to explore new areas of activities in the field of information processing. In a more radical departure, the structure of the traditional program of the conference was altered to focus the attention of the entire conference on the profound influence that computing is exerting on all facets of American society. In place of the usual keynote speaker, the Conference Committee invited nine outstanding speakers from disciplines outside of the traditional definition of data processing to explore the question of how computers are affecting the world around us. The speakers from government, education, and industry who honored our platform on the day-long session that closed the Conference, were not casual observers remote from the changing world of computer technologies, but rather men of stature that knew and used computers in their diverse fields.

This volume brings together the excitement of these phases of the conference, the edited results of significant panel sessions, and the thoughtful remarks of the invited speakers. Also honored in the record are two pioneers in the field of computers and information processing, George Robert Stibitz and Konrad Zuse, recipients of the second Harry Goode Memorial Award. It is fitting that AFIPS should record in a permanent record the tributes and responses given in connection with this award. Although Miss Ida Rhodes was personally unable to attend the conference due to illness in her immediate family, the complete text of her Luncheon Speech is made a part of this volume.

A post-conference volume also is an opportunity to give public recognition to a dedicated Conference Committee. The Committee was responsible for a successful conference that set records in attendance and opened new vistas in the information processing field. The names of the members of the several committees are here recorded in recognition of a job well done.

Special recognition goes to Samuel Nissim and Sei Shohara, who together made this second volume possible.

ROBERT W. RECTOR
General Chairman
1965 *Fall Joint Computer Conference*

Preface

The structure of the Technical Program for the 1965 FJCC emanated from a conference theme that emphasized the role of computers in modern society. The technical program aimed at satisfying two goals: (a) serve the needs of the entire membership of AFIPS which has constantly expanded to include more diversified disciplines, and (b) focus attention on the role played and the contributions made by computers to American life and its institutions (cultural, economic, etc.).

To implement the former goal, Friday sessions were instituted. No record of these sessions has been kept in either Part 1 or 2 of the Conference Proceedings. The responsibility for organizing the Friday sessions was delegated to member societies, under the supervision and initiative of the Technical Program Committee. This permitted coverage of specialized subjects for which no time slots could be allocated within the main three day technical program planned for much wider membership participation. Sessions were also added to serve the needs of management and marketing oriented membership of AFIPS. These included broad-coverage sessions (tutorial, panel and informal discussions) spread throughout the conference. Records of two such panel sessions—"The Computer Industry In The Buyer's Market" and "The Overseas Computer Market"—are contained in this volume.

In an attempt to implement the latter goal, a Thursday session involving nine "keynote" speakers was introduced to focus attention on a wide gamut of disciplines in which computers play an important role. During the first two days the sessions were designed to cover the more notable achievements of the past, and present, and future trends. These were aimed at answering the questions "Where are we now?" and "Where are we going?". These sessions were then brought into focus in the Thursday session to answer the question "How are we going to affect the world around us?". The addresses by the nine "keynote" speakers are contained in this volume. These papers dominate the second volume of the Conference Proceedings, hence the title adopted for it: "Computers and Their Impact on Society."

Part 1 contains formal papers delivered during the first two days. This part also contains additional material presented the first two days. Besides the two panel discussion records mentioned earlier, there is a record of panel discussions on "The Future Of Electro Mechanical Mass Storage" and "Promising Avenues For Computer Research." It contains late papers which do not appear in Part 1, and final versions of papers for which the preliminary draft was erroneously printed. It also contains an errata for Volume 1 which readers should carefully note.

A novel experiment at the 1965 FJCC was the Discuss-Only sessions where preprints of the papers were made available in advance of the conference. This permitted devoting the session to a discussion of these papers rather than a formal presentation of them. Despite the fact that the preprints were sent out late, and that a few session chairmen and authors did not really implement the Discuss-Only concept well, most of the conference attendees polled were enthusiastic about the results. It must be emphasized, though, that only the concept of Discuss-Only sessions and its utility have been proven. The many flaws that were discovered in its implementation still require a significant amount of "polishing." It should be noted that all papers in the Preprints are included in Part 1 or 2 of the Conference Proceedings.

Voice prints (on magnetic tapes) have been made of certain sessions not covered by these Proceedings, and for which permission was obtained from participating panel members. (Included are, for example, "Extra-Sensory Perception and Man-Machine Communications," "Implications of Information Storage and Retrieval in the Human Brain," etc.) A copy may be obtained (for a fee) upon request.

We hope that the combination of advanced technical sessions, tutorial sessions, discuss-only sessions, and panel sessions provided a good balance meeting the needs of all attendees active in hardware, software, and management applications. However, as is the case with every conference treating a field as broad as information processing, the 1965 FJCC Technical Program emphasized some areas at the expense of others. We endeavored to direct the emphasis toward topics of current interest and ferment. For example, the increased speed of components has intensified the imbalance between terminal equipment, memories and central processing units with memories forced to bridge the gap. Thus, computer memory technology is emphasized in these Proceedings to direct attention on problems and prospects in this area. The topic of time sharing is also one of current vitality. The two volumes contain several papers related to this area, covering problems of hardware, software and usage. As information processing systems become automated throughout the economy, more attention is required by the problems of applications and management. This, too, is emphasized in these Proceedings.

The vitality and success of the Technical Program reflect the excellent efforts by conference participants, attendees, and planners. We are grateful to all the authors of the papers who made these two volumes possible. Indebtedness is also expressed to the chairmen of the various sessions, including panel discussions, evening discussion sessions, workshops and epilogue session for the excellent results. So many did an outstanding job that it would be unfair to try to single out a few who did particularly outstanding jobs.

Sincere appreciation is due to our collegues in the Conference Committee, in particular the Technical Program Committee, for all the careful planning and hard work that went into structuring the technical program: Robert Gary (Secretary), Ben Ferber (Director, Management & Applications), L .D. Yarbrough (Director, Software), J. R. Bennett (Director, Hardware), D. A. Meier (Director, Member Societies).

The dedicated efforts of Messrs. Gordon Mitchell, Wayne Cotten, Robert V. Davies, Robert S. Kiel, James Mihalik, John F. Piontek, and David L. Yetter at NSA made possible publication of preprints for Discuss-Only sessions on schedule. We are indeed also proud of the paper reviewers who applied themselves so diligently to assure high standards for the papers contained in these two volumes.

The assistance extended so generously by Gordon Mitchell (NSA), Morton H. Lewin (RCA), Harold Judge (Edgerton, Germeshausen, and Grier), and John McCrummen (Bunker-Ramo Corporation) is gratefully acknowledged.

S. Nissim, The Bunker Ramo Corporation
T. B. Steel, Jr., System Development Corporation
Co-Chairmen, Technical Program Committee

FJCC 1965
Conference Committees

GENERAL CHAIRMAN	Robert W. Rector	*Informatics, Inc*
VICE CHAIRMAN	Linder C. Hobbs	*Hobbs Associates, Inc.*
ADMINISTRATIVE COORDINATOR	Marvin Howard	*Informatics, Inc.*
TECHNICAL COORDINATOR	Sei Shohara	*Scientific Data Systems*
EXHIBITS	Richard B. Blue, Sr., Coordinator	*TRW Systems*
	Samuel F. Needham	*TRW Systems*
	Emmett R. Quady	*UNIVAC Division of Sperry Rand*
PUBLIC RELATIONS	William D. Orr, Coordinator	*SF Associates*
	Richard Condon	*Fox-Mathis*
	Al Erickson	*Librascope Group, General Precision, Inc.*
	James Kehoe	*IBM Corporation*
	Dale Lewis	*CEIR, Inc.*
	Irwin Schorr	*System Development Corporation*
	J. Bradley Stroup	*General Electric Co., Computer Department*
	Dawn Walker	*Electronic Memories, Inc.*
SECRETARY-TREASURER	Mark G. Singleton	*North American Aviation, Inc.*
TECHNICAL PROGRAM	Samuel Nissim, Co-Chairman	*The Bunker-Ramo Corporation*
	Thomas B. Steel, Jr., Co-Chairman	*System Development Corporation*
	Robert K. Gray	*System Development Corporation*
	J. Russell Bennett	*Burroughs Corporation*
	Ben Ferber	*Consultant*
	John R. McCrummen	*The Bunker-Ramo Corporation*
	Donal A. Meier	*National Cash Register*
	Howard L. Parks	*The Bunker-Ramo Corporation*
	Lynn D. Yarbrough	*Harvard Computation Center*
LOCAL ARRANGEMENTS	Jerry Koory, Chairman	*Planning Research Corporation*
	Al Bongarzone	*Computer Sciences Corporation*
	Al Deutsch	*Associated Aero Science Laboratories, Inc.*
	Angeline Jacobs	*American Institute for Research*
	Jay KleinBard	*North American Aviation, Inc.*
	Frank Meek	*Control Data Corporation*
	Frank O'Neill	*Stardust Hotel*
	Arthur Rosenberg	*Scientific Data Systems*
	Al Rosenthal	*The RAND Corporation*
REGISTRATION	Eugene S. Gordon, Chairman	*System Development Corporation*
	Weldon C. Dennis	*System Development Corporation*
	George L. Jones	*United States Air Force*
	Robert J. McGill	*System Development Corporation*
PRINTING AND MAILING	J. E. McAteer	*Hughes Aircraft Company*
EDUCATIONAL ACTIVITIES	Marjorie F. Hill, Chairman	*Control Data Corporation*
	Robert L. Albrecht	*Consultant*
SPECIAL ACTIVITIES	Gloria M. Silvern, Chairman	*North American Aviation, Inc.*
	Carl N. Brooks	*Norair Division, Northrop Corporation*
	George F. Forbes	*Litton Industries*
	Irene E. Matthews	*TRW Systems*
WOMEN'S ACTIVITIES	Shirley L. Marks, Chairman	*The RAND Corporation*
	Doris Uncapher	
	Fred J. Gruenberger	*The RAND Corporation*
FINANCIAL ADVISER	Ted Gatto	*The RAND Corporation*
EXHIBIT MANAGEMENT	H. G. Asmus	*AFIPS*

THE AMERICAN FEDERATION OF INFORMATION PROCESSING SOCIETIES

takes pride in presenting the

1965 HARRY GOODE MEMORIAL AWARD

jointly to

George Robert Stibitz

for his contributions to, and pioneering efforts in,

- automatic computing,
- for independently proposing the use of the binary system, floating point arithmetic, memory indexing, and operation from a remote console, and
- for designing the first operating program-controlled computer.

and

Konrad Zuse

for his contributions to, and pioneering efforts in,

- automatic computing,
- for independently proposing the use of the binary system and floating point arithmetic, and
- for designing the first program-controlled computer in Germany—one of the earliest in the world.

Las Vegas, Nevada, December 1, 1965

Chairman, AFIPS Board of Governors: EDWIN L. HARDER

Chairman, AFIPS Awards Committee: SAMUEL LEVINE

Harry Goode Memorial Award Committee: JERRE D. NOE, Chairman, SAMUEL N. ALEXANDER ALSTON HOUSEOLDER, JON C. MCPERSON, WILLIS WARE

Biography—George R. Stibitz

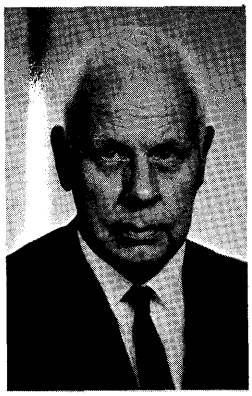
On September 11, 1940, George R. Stibitz demonstrated for the first time the remote operation of a relay-type computer from a console at Dartmouth College in Hanover, New Hampshire, over telegraph lines to the main frame in New York. This Model I Relay Computer climaxed George Stibitz' intensive investigation into computer principles, theory, and existing technology. The first model was followed by five succeedingly more advanced relay computers, built during and after the Second World War.

George R. Stibitz received a Ph.B. degree from Denison University in 1926, a M.S. degree from Union College in 1927 and a Ph.D. in 1930 from Cornell University. From 1930 to 1941 he was a research mathematician at Bell Telephone Laboratories; from 1941 to 1945, a technical aide with the National Defense Research Council of the Office of Scientific Research and Development. Since then, George Stibitz has rendered valuable services as a consultant to many clients in a wide variety of scientific endeavors.

George Stibitz virtually designed the first two relay computers and made major contributions to later machines of this series, two of which had most of the features of modern computers such as binary arithmetic, indexing, self-checking, and floating point arithmetic. Among the "Stibitz computers," as they are sometimes called, are the Relay Ballistic Computer for NDRC (1943/44), whose original model now is at Fort Bliss, Texas, and large relay computers now at NASA's Langley Research Center and the Army's Aberdeen Proving Ground.

George Stibitz' activities have, over the years, encompassed a multitude of scientific subjects, from propeller vibration computations and theory of errors in counting computers, to guided missile simulators, information theory, and learning machines. His investigations have found expression in many published works, in unpublished reports and memoranda, and in talks and lectures before professional societies and institutes of higher learning. Among the earliest of his many patents, George Stibitz' patent for a Binary Computer dates back to 1943.

At present, he is associated with the Department of Physiology, Dartmouth Medical School, where he is principally concerned with applications of mathematics and computers to biomedical areas and computer program models of the nephron and cilia.



ACCEPTANCE SPEECH OF GEORGE R. STIBITZ

I need not tell you how greatly I appreciate the honor of your recognition, and I thank you and the Award Committee for granting me the Harry Goode Award.

Naturally, since the award has been made for pioneer work in the later 1930's and early 1940's, my thoughts turn to that period, and I have looked back over a few records of those days.

I recall the first binary adder I know of. It was built on our kitchen table in 1937. The arithmetic unit consisted of two telephone relays. The storage—with a capacity of two bits—was two more. The input mechanism was a pair of keys made of sheet metal and the output was presented visually by two flashlight bulbs.

From this model to a binary computer for complex arithmetic was a fairly rapid step, and in a few months I had designs for such a computer, using the excess-3 code, as well as plans for a more general algebraic tape-and-keyboard controlled computer.

The Bell Telephone Laboratories had need for the complex computer and it was built and put to use in 1939, but the astronomical cost—close to $20,000—was judged excessive for a computer, and the more elaborate plans were not used until the war.

I am thinking, too, of the contributions of Sam Williams and of Andy Andrews to the early developments, especially to the series of relay computers built during the war years.

It was interesting to glance back at some of the opinions of experts and of lay people in regard to the first of the binary computers. Here are a couple of comments from news releases about a remote control demonstration pre-dataphone, from Hanover, New Hampshire, to New York City in September 1940.

In the *New York Sun,* September 9, 1940, I find: The Bell Laboratories "have not the remotest idea that with their machine they ever will be able to perform a service for any outside industry and they do not foresee the time when Johnny, aged 12, will be able to pick up his telephone receiver and ask the operator for the answer to seven times nine."

But I find a more optimistic view of the same demonstration expressed by a reporter who signed himself "L.W." in the *Newark* (New Jersey) *News,* September 9, 1940. I quote: "And if this may seem a restricted use, just remember there was a time when few people thought the world would have much use for flying machines."

Contrasting the views of these two sources, I feel there is a moral of some kind to be drawn, but can't quite make up my mind what.

In my present occupation of applying mathematics to medical and physiological problems at the Dartmouth Medical School, I have had constant contact with the computer there, and stand in awe of the accomplishments of you people and your colleagues in matters of speed, capacity, and convenience of the computers now available.

Again, I thank you and join in remembrance of Harry Goode in whose name the award is given.

Biography—Konrad Zuse

KONRAD ZUSE is the German pioneer in automatic computing. A building engineer by training, he soon found the numerous computations for statics a tiring and laborious undertaking, which finally led to his decision to construct a mechanical aid. His early work on these computing "mechanisms" covered the period from 1934 to 1936, during which time he laid the foundations for the Models Z-1 and Z-2 Relay Computers built between 1936 and 1940. In 1941 he sought a patent for his "*Rechenvorrichtung*"; the patent was finally granted in 1952.

A native of Berlin, Germany, Konrad Zuse received his technical education in his home town at the Technische Hochschule (Berlin University of Technology), from which he later received an honorary Doctor of Engineering degree (*Dr. Ing honoris causa*) in 1957. Dr. Zuse also holds several other important patents in the field of computing science. Independently of others, he proposed the binary system and the floating decimal point concepts as applied to computer design and built the first program-controlled computer in Germany, the Model Z-3. This machine had 2,600 relays, and a 64-word memory of 22-bit words. The computer was built between 1939 and 1941.

Following these achievements, Konrad Zuse constructed two special-purpose models of this computer: the Model S-1 for aircraft-wing measurements, and the Model S-2. This latter model can be called the first process control computer because the results of about 100 measurement points were continuously sampled and transmitted to the machine. During this time he also began work on the Model Z-4, the only computer he was able to save; all of his other computers were destroyed during the Second World War.

Undaunted, he continued his work on relay computers which, in later years, gave way to vacuum-tube and solid-state machines. As early as 1945/46 he developed the "plan calculus," a forerunner of modern programming languages.

Starting his own company in what is now West Germany, the Konrad Zuse Kommanditgesellschaft, in 1949, he became an industrial pioneer, with computers gaining in importance in the German market. While he published relatively little, his thinking strongly influenced the development of the computing sciences in German-speaking countries.

ACCEPTANCE SPEECH OF KONRAD ZUSE

I was very happy to receive the invitation of the American Federation of Information Processing Societies, to come to Las Vegas, first because of the honor attached to this invitation, but also because this provided me with the opportunity to come to Las Vegas without my wife being able to object.

I regard it as an exceptional honor to receive the Harry Goode Memorial Award. I do not only consider this as a personal distinction, but I am also highly gratified to see that an American organization should thus appreciate the achievements of European pioneers.

For these reasons, I think it appropriate to tell you some experiences about the history of European computer development.

Let's start with some general considerations on the calculating machines. In contradiction with the general opinion that Pascal and Leibniz were the first ones to have developed calculating devices, it was found by recent historical research that the German Schickard was the forerunner for some 30 years.

Most of you will have heard the name of Babbage. He is the real father of the program-controlled computers. More than a century ago he had made the plans and started the construction of a digital computer. Simply because the technical means available at that time were not adequate, he failed to achieve his goal.

Only a few people realize that he already knew the conditional order.

As forerunners of the present computer development, some other Europeans can be mentioned. For instance, the English mathematician Turing who, from the purely mathematical logical point of view, used the concept of a universal calculating machine. He used this concept, now known as the Turing machine, when analyzing the computability of mathematical functions.

The Frenchman Valtat had already back in 1936 applied for patents on the idea of binary calculating machines. Another Frenchman named Coufignal had developed in his doctor's thesis, only recently publicly known, certain ideas concerning the program control, logical operations and binary system for application in computing machines.

Specific development of computers as we understand them now took place between 1935 and 1945. Quite independently the work proceeded both in the United States and in Germany. Very remarkably, both approaches were not made by specialists in the field of calculating machines, but by outsiders. Since both developments were carried out practically during the same period, it may be useless to try to determine which machine was the "first" one.

I am convinced that you are better informed about the development in this country than I could be, and therefore I would like to give some more information about the development going on in Germany until 1945.

In the early thirties, I was a construction engineering student at the Technical University of Berlin. You will probably know that, in this field, extensive calculations have to be carried out, especially for indeterminate systems. This led me to think about the possibility of designing computers. My goal was to be able to carry through, fully automatically, complete calculation sequences. I did not know anything about computers, nor had I ever heard about the early work of Charles Babbage. Thus—unprejudiced—I could go new ways. In order to illustrate the opinion of the manufacturers of calculating machines at that time, I would like to mention a telephone conversation which I had in 1937 with one of those manufacturers. He told me that it was indeed wonderful that I as a young man had dedicated some time and efforts to the development of new ideas, and that he wished me all the best for possible other inventions, but stated that in the techniques of calculating machines all feasible solutions were already exhausted. Therefore, it would be absolutely hopeless to come up with any new ideas. In addition, he asked me whether my machine was based on the "sequential addition principle" or on the "one times one table." To this I replied that for my machine this was of no importance whatsoever. Here you should know that at that time the specialists of calculating machines were divided in two schools of thought, each applying either principle. According to the opinion prevailing at that time, only a lunatic could make a statement that this difference was irrelevant for his design. Nevertheless, the manufacturer mentioned

came to my workshop and I was finally able to convince him that, in a machine operating on the binary principle, this was indeed irrelevant.

Due to the lack of interest encountered I started on a private basis, supported financially by a few personal friends on a very limited scale. Later I received some backing from research institutes and from the German aircraft industry. To begin with, I built various all-mechanical models. However, I soon realized the limits of mechanical design and, therefore, adopted the electromechanical concept. This led eventually to the sequence-controlled relay computer Z3, which was completed in 1941.

The main characteristics of this machine are: sequence control, binary system, floating-point and electromagnetic relay technique with wide application of switching theory.

This machine was complete and ready to operate. Mainly it was used for test computations. The programs served to determine the eigenvalues of complex matrices; this was a problem of particular interest to certain aerodynamics specialists.

Apart from this one, another special purpose prototype was built, which had a fixed program store wired into rotary switches. It served the task of evaluating wing measurements of a missile assembly line, and operated 16 out of 24 hours for 2 years. The problem was to multiply about 100 gauge values obtained on missiles with certain characteristic numbers in accordance with an aerodynamical program, in order to get adjustment values for the positioning of the wings.

In a more advanced version, the gauges were read automatically via rotary switches, which transferred their positions into the computer. This device may well have been the first computer application for process control.

Proceeding from the universal computer Z3, an improved model Z4 was built, which was essentially completed and could operate on simple programs, when the war came to an end. Alone among the whole series, this machine, Z4, could be rescued. After some enlargement, it was later installed at the Swiss Technical University in Zurich. The historical model Z3, which had been destroyed during the war, was later reconstructed. It is now in the German Museum of Science and Technology in Munich.

Due to the situation in postwar Germany, any further work on the subject was rather difficult at first. Also, the task of designing new computers became teamwork. As to myself, I am afraid I was largely occupied by other problems due to my function of building up and running the Zuse Company. I could, however, hold some essential personal share in the development of a general-purpose automatic digital drawing table, which went into production in 1959, together with other special devices, for geodetics purposes.

I would like to mention the fact that, in 1938, my friend, Dr. Schreyer, set himself the task of redesigning my relay computer by means of electronic components. He developed a set of basic circuitry for solving the fundamental logical operations and built a prototype binary arithmetic unit for 10 digits. During the war, we submitted the design drafts for an electronic computer with 2000 tubes to German research authorities, but their reaction was negative.

I would now like to take the opportunity to cite the name of Dr. Dircks. He had, at a very early stage, produced essential ideas in the field of magnetic storage devices. During World War II, he constructed a prototype of a small magnetic disc file.

Apart from this work on hardware, I was able to devote some of my time to software philosophy as well. The relay technique induced me to do some work in the field of switching theory. From this I gained the fundamental knowledge that any information may be dissolved into true-false values (today called bits). I introduced the general concept of computing as follows:

To compute means to derive new information from given information, according to a prescription or better said algorithm.

At the outset, I developed some drafts for logical computers; for one of these we built a simple prototype. From this resulted the necessity to develop first a general formula language which might be used to formulate even the most general and complicated computing sequences. This work was finished in 1945 with the creation of the so-called *Plankalkül* (Programming Calculus). Essential features of this were:

Consequent adherence to a systematic representation of all information structures.
Extension of the function concept.
Introduction of the sign "results in" (=>).
Incorporation of logical operations and functions (calculus of propositions, calculus of relations etc.).

In the hardware which I developed until 1945, I strictly avoided the use of conditional orders. The theoretical analysis made proved to me that a single wire connection feeding back from the arithmetical unit to the program controlled unit would enlarge the logical capabilities of the machine to an extent, the consequences of which were difficult to foresee. I elaborated the various possibilities of such a feedback on a pure theoretical basis and introduced them in the software philosophy. While doing so, I clearly found that if once the step is made towards the conditional order, the door is open for consequent development leading to the present-day and future computers which finally, step by step, take over large tasks of the human

brain. However, I did hesitate to be the one who made this dangerous feedback connection at first.

The Programming Calculus served partly as a starting-point for our modern formula languages, such as Algol and others. Its basic ideas may not, however, have unveiled their full implications until now. As you will agree, we have at present arrived at a point in time where the more specialized formula languages in use today have reached some limits. The hardware which I developed in those days has now become of historical value only. On the other hand, I tend to assume that the software ideas which make up the *Plankalkül,* or Programming Calculus, may be of some importance in our day. I cherish the hope that they will still bear some fruit.

I also devoted some—perhaps interesting—thoughts to the self-reproducing machine. In the United States, this problem has been treated mainly from a mathematical point of view. My own thinking was independent of this and moved more in the field of engineering. It led to the concept of what may be called a "technical germ cell." Again I hope that some day it will gain a certain actuality.

The brief report that I could give you includes mainly the period until 1945.

You all know that after World War II, an upgrowing development took place in the field of digital computers. In Europe this postwar development was initiated mainly in England. However, I don't feel competent to select from the large amount of work done the most important approaches.

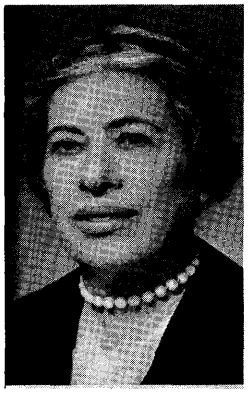
Also in Germany some development was started by various groups. Before the computer industry gained some importance, there were research groups at universities and other institutions who took the initiative. Amongst those who participated in this work, I would like to mention Dr. Billing, who designed at a very early stage a drum memory.

Besides the work done in England and Germany, there were some other computer developments carried out by the following European countries: Austria, Belgium, France, Netherlands, Sweden, and Switzerland. These are the ones known to me, and it is possible that other countries unknown to me have also carried out some original development work.

We Europeans have a considerable admiration for the research done in this field in the United States, as well as for the American computer industry.

On the other hand, I am glad to see your interest in our European approach and I am very grateful that I was given this opportunity to talk to you on this subject.

I do sincerely hope that international cooperation will continue to grow and that in this activity sector ideas coming out of good old Europe may still contribute to our common progress.



The mighty man-computer team

IDA RHODES

National Bureau of Standards
Washington, D.C.

If the Martians, Venusians, or Plutonians ever bother to observe the antics of their neighbors on the Planet Earth, they must be vastly amused by our attitude toward the digital automatic computer, the DAC for short. While professing to be completely baffled by the frenzied rush on the part of the lemmings to be drowned in the sea, we—earthlings—have perversely chosen to denigrate the unsurpassable human brain by affixing to a pile of wires and tubes that ludicrous title "The Thinking Machine." What a churlish way to thank Mother Nature for our bountiful mental endowment which dwarfs into insignificance those fabulous gifts of magic, lavished upon their favorites by the doting fairy godmothers.

Let us take inventory of our divine heritage. The human cranium possesses a storage capacity equivalent to at least a thousand billion binary digits. Supplementing this opulent installation, there are five known input devices—our magnificent sense perceptions—and perhaps additional senses, of whose existence we are only dimly aware. Maintained by a superbly efficient system of physical organs, our mind is enabled to exercise an astounding number of sublime functions. The myriads of impressions, which we receive through our senses, are being constantly shuffled and miraculously combined into *concepts,* whose count exceeds by far the total number of elementary particles in the entire universe. Yet, this colossal aggregate is actually infinitesimal in comparison to the fantastic number of *ideas* which our brain can engender by continuously associating various concepts. We are thus led to the conclusion that should another Merlin arise, capable of constructing a lifeless contraption for simulating human thought, he would run out of all the material available in the Cosmos, long before he succeeds in completing his very first specimen.

The elation, which some of us may feel at such largesse on the part of Providence, rapidly subsides as we reflect on how few members of our species either fully appreciate, or strive to make use of, their prodigious birthright. Regard the humble amoeba. With its single tiny cell, it manages to find its proper environment; to gather, ingest, and digest suitable food; to eliminate its wastes; to grow, to mature, and to produce young. How many of the earth's three billion human denizens care to utilize their stupendous mental powers for higher or nobler aims than the unicellular amoeba?

We trumpet with loud fanfare the blessings of our huge array of devices, invented for the purpose of extending and enhancing our natural prowess, as well as of freeing us from back-breaking, time-consuming tasks. Let us bear in mind, on the other hand, that every tool is a two-edged sword. If its wielder be animated by vicious motives, it can turn—in his hands—into an accursed weapon of wanton destruction. Impressed by man's mighty intellect and the horse's amazing strength, our ancestors were inspired to blend the two into the image of a devastating Centaur. It is now within our power to replace this violent monster by an incomparably more efficacious, yet supremely beneficent, Megataur. Embodying the awsome potentialities of the high-speed computer and guided, at all times, by the

highest dictates of an exalted human conscience, the new image could become a source of radiant hope for our strife-ridden, despair-laden world.

It is gratifying to learn that a group of high-minded medics have united to form "Physicians for Social Responsibility." It is easy to predict, that the members of our profession, who were deeply stirred by the eloquence of John F. Kennedy's inaugural address, or by the loftiness of Pope Paul's plea for peace, or by the earnestness of President Johnson's appeal to join the Great Society, would be eager to endorse the "Megataur for Social Responsibility."

We might start by borrowing a custom of the ancient Hebrew scribes, who underwent an elaborate daily ritual of self-purification, before assuming their sacred task of copying the Scriptures. Before outlining what constitutes, from my point of view, a similar ritual for Automators, I would want to make sure that all visitors' guns were checked at the entrance.

We are told that when the great de Forest—regarded as the father of television—observed the fare being dished out for the benefit of the captive viewers, he exclaimed in despair: "Heavens, what have they done to my child?" I suspect that the inventors of DAC are moved to utter a similar cry of distress, when witnessing the daily abuse and misuse of their illustrious brainchild.

It can hardly be a source of pleasure for them to be told that a prospective user of their wondrous prodigy requires only a brief exposure to mumbo jumbo in order to match the performance of an Isaac Stern on a Stradivarius. We are put in mind of the visitor to a European Museum. Glimpsing the piano of the composer Liszt, he decided to display his own virtuosity as a pianist; whereupon he voiced his conjecture that previous visitors must have been inspired to similar action. "I would not say so, sir," remarked the attendant. "The last person who came through here was Paderewski. He claimed he was not worthy of touching the keys of this instrument." Now I submit, ladies and gentlemen, that in order to become an accomplished automator, one must go through as much preparation and training, as—say—a skilled physician, or competent lawyer. Members of these and numerous other crafts have wisely established rigid criteria for licensing their applicants. From my point of view, it will be a red letter day in the annals of electronic computation, when means will be found to keep the rank amateurs away from our multimillion-dollar DACs. Those perennial bunglers tie up the machines with their error-laden routines which soon assume the visage of a ferocious hydra, since each time they undertake to correct one of their goofs, they manage to introduce seven brand new ones.

My next suggestion for a sacrificial offering will raise quite a few hackles. But remember, no shooting! The exceedingly simple and uniquely suitable language with which each inventor endows his DAC is calculated to secure maximum economy and expediency of performance. Yet it is consistently being discarded in favor of a spate of grotesquely time-wasting jargons. Disregarding the fact that superbly qualified teams of mathematical geniuses have not yet succeeded—after millenia of intensive effort—in creating an unambiguous universally accepted terminology for their science, the high priests of mumbo jumbo insist that their quest for artificial languages must go on, because of the great diversity of machine-code structures and the massive infiltration of the aforementioned breeders of hydras. I shall not take the time today to refute the fallacy of their contentions; instead, I should like to sponsor the alternate proposal of standardizing the machine codes. When Artur Rubinstein is invited to perform in Tokyo or in Moscow, it is not necessary for him to tote his Steinway to those distant places. Musicians have learned long ago the wisdom of regulating the basic notes of the scale, as well as the forms of the instruments that produce them. Although we have not yet reached the stage where our fraternity can undertake to compile a roster of instructions that will endure for all time, we have amassed enough experience and authority to agree on an excellent list which would serve quite well for at least a decade. I venture to predict that our demand for a universal adoption of this set would be greeted with a sigh of relief by all makers of DAC, since they would be liberated, at last, from the murderous treadmill of software construction and would be able to invest the considerable amount of money thus saved on truly effective data handling attachments.

If I still have any friends left in this audience, I hope they will bear with me, as I mention still another method of self-purification. The Russian people have an apt saying: "The house is still in the process of completion, and the rightful owners have not yet moved in, but the cockroaches are already in full possession of the premises." Our vital and honorable profession is in grave danger of being invaded by a horde of self-ordained geniuses who, under the guise of earnest seekers of scientific truth, may succeed in foisting their spurious wares upon generous but gullible sponsors. As an example of a field, highly susceptible to this sort of infestation, I might mention Information Storage and Retrieval, including Machine Translation. The only hope of attaining even a modicum of success in this and similar endeavors lies in the complete and cordial cooperation of all interested parties, not merely on the

national, but also on the international, level. No isolated, uncoordinated group of workers can ever come up with an acceptable answer to the heartbreakingly complex problems involved in those fields, just as no set of isolated, local meteorological observations could possibly yield reliable long-range weather forecasts. Let us make quite sure, therefore, that the ironic tale of the "The Emperor's New Clothes" is not reenacted right in our midst, lest we acquire the unsavory reputation of unscrupulous schemers and heartless mulcters of the public till.

The precious machine time now being scandalously wasted on the sprawling monsters of the milksop coders; on the gibberishes, broadcast from the Tower of Babel; on the devious machinations of the Emperor's tailors—might be almost sufficient to compute the location of that elusive fulcrum sought by Archimedes in order to move the world from its orbit.

As soon as we shall have succeeded in putting our own house in order, we can start setting up the Megataur for the vital business of cleaning out the Augean stables of human misery. Each generation is prone to put the blame for the hideous state of those stables upon the bigotry, hatred, and greed of its predecessors and to insist that an army of Herculeses would not suffice to cope with the accumulated filth. In fairness to our forebears, it should be pointed out that they did not possess DACs to help them solve the staggering social and economic problems which have been plaguing humanity for countless centuries. Lacking such equipment, even their noblest efforts had to resemble—perforce—the awkward performance of a tyro mathematician as, pencil in hand, he struggles to calculate a solution by the method of relaxation. Unable to assess properly the impact of all the conditions existing in the pertinent domain, he pounces upon some troublesome spot and labors assiduously to relieve the tension in its immediate neighborhood. Unfortunately, he only succeeds in raising a far greater protuberance in another, hitherto calm, region. He rushes to the new center of disturbance to apply the same panaceas, and this time causes a huge geyser to erupt in still another portion of the domain. The purity of his intentions is unquestionable, the diligence of his efforts is admirable, but the quality of his results is deplorable. If we now expand our vision to embrace the entire gamut of human activity and watch, in dismay, the vast mass of mankind, writhing and seething under the lash of unremittent hunger, pain, and indignity, we realize that any attempt to calm down permanently this restless agitation is quite futile without the aid of titanic implements that can help us analyze and remove its underlying causes.

The present models of the sadly miscalled "Giant Electronic Brain," though quite adept at smoothing out the bumpy domain which had confronted our tyro mathematician, is pitifully inadequate for tackling the formidable complex of matrices representing the all but hopeless tangle of human affairs. However, anticipating the advent of its far more overpowering successors, we may now dare to aspire to the solution of some of the most pressing problems facing our fellowmen. For the present, we can rely only upon the penetrating force of an exalted dream. I shall not presume to spell out for this audience what the nature of that dream ought to be. In the words of President Kennedy: "For this, every man must search his own soul." I would like, however, to relay an anecdote about Abraham Lincoln, which throws an interesting light upon the topic of self-interest.

While riding with a friend along a country lane, Lincoln was engaged in a discussion relating to the various motivations that underlie human conduct. The friend maintained that only altruism may be regarded as an acceptable guide for man's behavior, whereas Lincoln argued that the dominant impulse necessarily resides in selfishness. Suddenly the air was rent by the piercing squeal of a pig trying to extricate its head from the slats of a farmer's fence. As Lincoln jumped out of the buggy and ran to deliver the animal from its plight, his friend taunted him good-naturedly: "I suppose you are going to tell me that freeing this pig was an act of pure selfishness on your part." "It most certainly was" asserted Lincoln. "If I had failed to come to the aid of that poor creature, I would not have been able to sleep all night."

I think all of us will agree that Lincoln's interpretation of selfishness may serve as an excellent definition for "social responsibility." Over a period of countless generations, the term "reaching for the moon" was used as a synonym for an absolutely unattainable ideal. Yet recent events have deprived it of such connotation. One need not be a Walter Mitty to imagine the dizzying heights which may be reached by humanity, if each of us resolves to adhere to the principles of social responsibility at all times and under all circumstances. Such unswerving dedication is bound to instill in our subconscious minds an abiding habit that would automatically impel us to seize upon every opportunity to elevate the stature of the Megataur.

Let us take for our motto, then, a paraphrase of the immortal words, uttered by our beloved leader, martyred two years ago: "Ask not what the machine can do for you. Ask what you, in conjunction with the machine, can do for your country and for the world."

The computer and our changing society

SIMON RAMO

Vice Chairman of the Board
TRW Incorporated
Redondo Beach, California

Those responsible for this convention are to be congratulated for including a whole day's session on the impact of computers on society. Usually when engineers and scientists meet the subject matter is almost wholly technical. In a way this is strange because the common, quick definition of engineering is "the application of science to society's needs." If this definition were really to be taken seriously, it would mean that those who practice engineering would seek to be equally expert in science and in society, since one can hardly be professional about applying science to something he does not understand.

Engineers are only occasionally expert in the problems, needs, and organization of society. Conversely, knowing very little about science does not automatically make someone outside the engineering profession any more competent in social problems than the engineer. This is a great shortcoming of our world as we head for a much more technological civilization. There is, as a matter of fact, a missing profession—the "socio-technologist."

All in all, we may not be as prepared as we would like, but it is particularly urgent at this time to discuss the impact of the computer on our lives. More broadly, we should consider the impact of the new technology which involves all aspects of the handling of intellectual and informational tasks by electronics, and for which we will often use the word "computer" as a short, though inadequate, title. It is especially timely and important to take up the computer's relationship to society, because the computer is rapidly replacing nuclear energy and space as the leading technological item of confusion and fear in the public mind. We cannot today picture with completeness and precision the entire future effect of computers on our civilization. However, we can endeavor to eliminate some common misconceptions which are on their way to becoming well-established, harmful myths.

WILL THE COMPUTER REPLACE MAN?

This may be the greatest fear. It is apparent from the frequent appearance in the public press of statements such as "A computer can't think" or "A computer isn't creative—it can only do what man directs—it has no mind of its own!" These are defensive words arising from the myth that the computer is a competitor to man. It is apparently important to keep reassuring ourselves that we are intellectually superior. Thus, we rejoice when we hear that an automatic device somewhere has failed. When the electric power blackout took place in the Northeastern United States recently, many hoped to hear, and were anxious to spread the word, that a computer, to which we presumably entrusted too much responsibility, had goofed. "Please, God, don't let it turn out to be the result of a human error," was the prayer many intoned.

But that man and the computer are competitors is a misconception. Electronic information handling technology is presenting new ways to us to acquire, store, process, disseminate, and utilize the information that makes the world go round. It is making possible improved systems of production, banking, transportation, and education. The true concept is change—

change simultaneously offering two things: potential benefits and potential dislocations. A mature society will work at minimizing the negative reaction so as to emphasize the benefits. Properly handled, the computer in replacing man represents a small effect, and the potential benefits, a large effect. At any rate, fear and misconception will certainly work against realizing the benefits.

What the computer makes possible is not primarily a replacement of man or competition with man, but a new man-machine partnership. It enables an extension of man's intellectual and information handling capabilities and hence enhances man. The combination, exceeding the capabilities of unaided man, can better meet society's needs and can attain higher achievements. As to man's specific occupations that might be affected by his partnership, the result will be a new and broader spectrum of jobs, new professions, and greater satisfactions.

The X-ray machine did not compete with or replace the physician. It broadened the practice of medicine. It made it a better profession, a more intellectual one, a more useful one to society. It brought a requirement for technicians and apparatus and facilities not previously existing, all economically justified by the new technology.

The book is a machine having to do with extending man's intellect. When the concepts of printing and books were born perhaps some considered books as competitors to teachers. In a sense, they did replace something that the human educator previously had to accomplish alone, without books. But they succeeded in broadening the dissemination of knowledge, increasing communications of thoughts between people, leading to a much broader and larger educational system. Books made possible the employment of even more teachers, who brought more benefits to society than were economically achievable without the machine—in this instance, the book.

ARE EDUCATIONAL REQUIREMENTS FOR THE NEW TECHNOLOGICAL SOCIETY UNATTAINABLE?

There is a growing alarm that the educational requirements for the decades ahead simply cannot be met. The technological life toward which we are headed requires increasingly broader education of every young person to prepare him to make a living and to be a good citizen. A larger fraction of adults must now continue education to keep up with the rapidly expanding order of change and remain productive. Meanwhile it is seemingly becoming more difficult to provide this education. The cost of facilities and the demands for educators are escalating, and shortages seem certain to grow. This educational crisis is often blamed on technological advance. The technological society generates, it is claimed, impossible demands for education and hence guarantees its own failure and collapse.

A breakthrough in educational concepts is clearly needed—something comparable with printing and the book. Advanced electronic technology offers precisely the kind of new concepts needed to revolutionize the educational process. Human educators can be assisted by networks of electronic facilities that are backed by an educational industry that does not even exist today. Educational experts will be able to plan present, test, and analyze with great enhancement of their informational and intellectual powers, rising to a new plateau of accomplishment. Here the man-machine partnership can provide a match between society's needs for all kinds of education, and society's ability to supply that need.

Embryo teaching machines of today are as far away from the full use of technology in education as the first stone tablets are from today's television network. Matured computer and electronic networks, applied under the skillful direction of educators and engineers working together, can provide new forms of education in the home, in schools and in industry. Educational communication satellite systems can provide to a special network in the homes of the nation a choice of hundreds of different courses of study chosen simply by pushing the right buttons on the home educational TV set. Carefully presented programs available when called for can involve student identification and participation, the answering of questions by pushbutton, the monitoring of answers by "live" experts, and a record of results.

In schools, material can be presented not only by the human educator in person, but through audio-visual devices which can automatically speed up, slow down, or switch to a completely different presentation, all in automatic response to the student's apparent ability to follow the material. That is, the presentation can periodically include questions to the student, the answers to which can influence further presentations to suit the student's pace. Computer systems can keep track of the progress of millions of students. These systems can compare progress against estimates and can make possible statistical analyses and a type of creative planning not now remotely practical. At the same time, an individual student can have synthesized for him presentations or tests completely unique to his particular requirement. These can be determined by the virtually instantaneous availability of a full record of

progress and a comparison of that record against alternate courses for him to take in the future.

Electronics does not separate the student from the human teacher any more than the electrocardiograph keeps the physician from having direct contact with and interest in his patient. On the contrary it enables him to "listen" to his patient's heart with greater skill. Comparably, the human educator will be able to consider an enormously larger number of facts about both student and course with greater accuracy and confidence. He will be able to make available to the student more material with greater efficiency and with a much broader selection. He will be able to reach the eyes, the ears, and the minds of students, children and adults alike, everywhere. He will be able to propose new concepts in education and to check his plan to discover how it worked, and then alter and improve it.

In the coming technological society, education can become the greatest occupation of man and his greatest preoccupation as is required to meet the challenges of the age. The educational profession will expand into a larger number of specialties, and it will be equipped to do more research to achieve important generalizations to guide its members in that educational acceleration. Educators will have the support of a huge industry that provides systems engineering, communication networks, information dissemination, storage and retrieval, and analysis techniques capable of supporting the higher educational plateau.

It is a myth to think that rapidly advancing technology is creating an impossible dilemma in education. It is rather that the crises that have been developing for years in education may at last be attacked with a scope equal to the task by the utilization of advanced technology.

WILL THERE BE NO NEED FOR UNSKILLED LABOR?

The leading "dark horse" candidate for the myth of the century may be this: modern technological society eliminates the need for unskilled labor. Since, no matter how technology is used for the good of society, not everyone will be well educated and competent to work at highly intellectual pursuits, this myth suggests we will permanently have a growing number of unskilled and unemployed citizens. I venture to predict that after some initial dislocations, and within the period of a decade, we shall commence to realize that, far from there being a shortage of jobs for unskilled labor, there may well be a shortage of unskilled labor to fill jobs.

To see why this is so let us move ahead to the year 2000. Let us also assume, in complete consistency with the premise of the myth, that unskilled labor will no longer be needed because we will have reached an essentially fully automated society. The factories will turn out all of our material needs through nearly automatic operation with little intervention by man. There will be moving sidewalks and automatic rapid transit cars in the cities which, like today's automatic elevators, will perform without human operators. None of us will use coins or currency anymore. We will instead assume that when we buy a necktie or a piece of land we simply hold our finger against a little window so our fingerprint can be scanned electronically, then in the record-keeping facilities thousands of miles away something will be taken off our account and put on someone else's account. Let us assume that automobiles, missiles, and houses are designed by automatic computer programs. The tests taken of each of us in the hospitals will be automatically recorded and analyzed, and a treatment prescribed with little intervention by human operators or analysts.

This is an exaggeration, an extreme description of a fully automated age. However, this route enables us quickly to arrive at a series of important points. The world fitting our description is reachable only by a complete redoing, updating if you will, of our entire national physical resources to take advantage of technology to reach the fullest automaticity conceivable. To achieve such a system, even if it were possible or desirable, would require an expenditure per year far exceeding our gross national product. It would mean virtually creating a whole new nation full of new expensive facilities and resources and a newly developed and implemented way of operating them. Our cities, factories, hospitals, schools, and transportation systems would all have to be rebuilt. We do not have the total resources in manpower, skilled and unskilled, to accomplish this transition by the year 2000. Even if it were economically or socially sound, even if there were individual or cooperative incentives to seek to achieve it, the total cost would be too great. We simply don't have what it takes.

But this is another way of saying that, in a basic sense, a major transition of this sort is economically unsound. What really will happen will be much more sensible. To use advancing technology to improve all aspects of our society will involve the proper use of people and things, men and machines. The only practical and reliable plan is to make an optimum selection from these two categories for the tasks to be performed. Now, man can be produced with relatively cheap labor, and can be trained to do an enormous variety of tasks with his brains and senses while having physical mobili-

ty, for a rather reasonable initial and annual maintenance cost. By comparison with a certain class and variety of jobs man can do, a machine designed to do the same tasks becomes absurdly expensive. We can abbreviate this analysis by asking a substantive and symbolic question: *In the highly automated society who will change the light bulbs?* That is, associate each of us with a large number of black electronic boxes that do virtually everything automatically, eliminating the need for unskilled labor, and ask who would interchange the boxes when one of them malfunctions or wears out? Perhaps the ready answer is that this also can be done automatically. Then we must assume the design and building of still more electronic boxes to maintain the first set and these must respond to automatic diagnosis devices, calling out automatic putting in of spares and the automatic transportation of equipment from one place to another. Obviously, the extreme of trying to do away with man entirely is as silly at the level of the unskilled as it is at the skilled level. Nor does it make sense to design a way to operate our society to provide for all of our physical, intellectual, and cultural needs without intervention by man.

A city consisting entirely of trained engineers and scientists might either have to remain dirty, or depend on a means of drafting creative people for an hour or so a day to keep it clean. Or else the skilled workers might have to put their time and resources into designing a city that keeps itself automatically clean, even though this might be so expensive in the use of their time and resources as to prevent their realizing many other more important benefits.

The intelligent means of accomplishing that which man needs and wants done in society is to use an optimum partnership of man and machine at all levels of skill. This most favorable condition may be difficult to reach in the future because the percentage of unskilled workers who will be matched to the duller, mundane, less intellectual task will be fewer than the demand. As society moves forward, as we broaden the spectrum of man's activities as a result of his being able to make use of machine partners on the intellectual-informational front, and as more people become educated, we may find ourselves forced to a greater than optimum reliance upon the machine just because of a shortage of unskilled labor.

IS TOO RAPID A RATE OF CHANGE DISORGANIZING SOCIETY?

It is becoming common for all of us to complain that we can't do anything these days without a huge exercise in frustrating arrangements. Life is becoming too complex and we associate this with its being one of rapid technological acceleration. Because earlier societies were simple and understood, they appeared well organized. The computer age seems to be headed for increasing confusion.

Fortunately, the computer is the foe of disorganization and chaos. It is the tool of all time for carving orderly patterns of control. Indeed, the computer has just arrived in time. Electronic information handling systems are being developed and installed just fast enough to prevent our being completely drowned in a sea of red tape. The problem of keeping track of everything that has to be included to keep the operation of the world running is growing, and the ability of electronic systems to help us keep our heads above water is timely and fortunate. Electronic systems are ideally suited to gathering information, assembling facts, applying logic, and controlling the flow of all needed data and directions. The constraint to smooth physical operations of our civilization today is a bottleneck of paper, of information handling. It has been bad enough in the recent past but, in the future, without modern computer networks, none of us would be able to get our pay checks, keep our insurance policies active, obtain our bank statements, deliver messages, keep track of who owns what, and maintain a semblance of order. Without electronic systems now being developed and implemented we would have, in short, a much lower standard of living and approach the very chaos we fear. The misconception here about the impact of the computer is a sort of "guilt by association." But the computer is the hero and not the culprit, the defender and the hope, not the attacker and the villain, of our fast-paced, increasingly complex society.

WILL THE FUTURE CONSTITUTE A ROBOT CIVILIZATION?

It is becoming increasingly common to believe, with resignation and chagrin, in the certainty that ours will become a robot civilization. Man is envisioned as becoming, in the future, an anonymous cog in a vast interconnection of cables, computers, signals, and moving vehicles. The world is pictured as a place where every action of society to its infinite details will be planned and controlled, with a man a mere number, an apathetic, nonparticipating disinterested bystander in decision making.

Such a structuring of society is inconceivable unless one simultaneously postulates the existence of a pervasive, automatic electronic information system that

senses all of the data needed to control our daily operations; one that is busily engaged on a mass scale in communicating information, to which machines and men all respond. Now, if such a system were to exist at some future date, it would make possible a mode of social operation which would be quite the opposite of robotism. A ubiquitous electronic information network, as effective and efficient as this, would make possible the disseminating to every citizen of all items of information worth evaluation for the selection of goals by the people. This network could gather opinions and use this polling in the making of decisions.

The technological advances that one must assume to be concerned over a future robot society would also make possible individual participation in our homes of a form of "instantaneous democracy." The same system that can tell millions of people exactly what to do can just as well ask them to choose what to do from a group of well-presented alternatives. The citizens of the future, so far as technological potentials are concerned would be able to tune in on the highest level discussion of the big issues and take part by expressing their opinions electronically from their homes in the deliberations of the Congress, state legislatures, and city councils. I am not suggesting that it would be to our advantage to have every citizen share in every decision that affects the complex operations of our nation. But the same technological system that makes possible a robot society, where everything is controlled because all the information needed for control is at the right place at the right time, also can enhance democracy. It can make possible an informed, interested public, the tapping of citizen opinion on issues, and the creating of vast loops of citizen participation in decision making.

Whether we move towards a robot society or in the direction of "on-line democracy" is not determined by technology. Science merely offers us the choice. A misunderstanding of the possibilities, a firming up and prior acceptance of the myth of robotism may keep us from having a choice.

WILL FREE ENTERPRISE BECOME EXTINCT?

The advent of the computer age appears to many to carry with it the dreaded planned economy. They forecast an automated socialism without free enterprise or private capital at risk, a detailed control of the economy, meaning "state control." Their fear is that we will lose creativity, individual initiative, the advantages of competition and incentives and, especially, give up the freedom to take individual paths to the new heights which men can attain when unhampered and not overly controlled. Nor is it consoling that the Soviet Union's attempts to completely control her economy by planning from above is falling short of her goals in practice. The fear is that perhaps the Soviet Union tried to do it without the means; with a broadly based network of electronic information systems it might become technically practical. Then if the government and the people choose to do so, they can arrange for this kind of a controlled society in the future.

To many individuals there appears to be only two paths. One leads to planned economy, the socialistic state. The other winds back to 19th century entrepreneurship, so important in building America's economy but hardly accessible to us now. But it is submitted that there is a third, much more likely path. It leads to the creation of a free market of an unprecedented form and level.

To perceive this route we need to note first that planned economy cannot really control in detail unless the consumer is in the loop. Perhaps the Soviet Union has proven this point; although the government can plan what to produce, it cannot force the citizens' utilization of the products in accordance with the plan. As we try to do this in the technological society again we must assume the existence of an all-embracing electronic information system that can reach every man and machine. If we can communicate with everyone to ensure the working of the economy, as planned, we can also ask everyone what they want out of available product possibilities. We go on now to note, oversimplifying perhaps to make a point, that the essence of true free enterprise and our capitalistic system is the free market. So long as we have a means for people to freely choose on what they will spend their money, so long as the producer is able to offer his ideas and goods publicly for sale, then we will continue to have the advantages of free enterprise. It is only a detail how automated the flow of information is, if that information flow is used to step up the process of consumer selection, capital investment, production, and distribution.

So the basic concept is that a national electronics information system that has the technological capacity to effect a thoroughly controlled economy in principle, run from the top (the government), must also provide the necessary communications for a vast consumer free market network run from the bottom (the buying public). In fact, a nation which has a working electronic information network reaching every nook and cranny of its economy can create a free market of a form and on a level that civilization has never previously known. It could be a market in which everyone knows quickly what is available. A proposal to produce something of interest to possible purchasers could be quickly viewed

and assessed by potential buyers. Each of us in our homes could electronically respond directly to a "commercial" that describes next year's contemplated automobile models and offers a substantial discount for orders placed now. We could step to our sets and push the right buttons to confirm our purchases. This kind of direct consumer information applied to vacations, houses, soap, refrigerators, and even educational courses could be used by the automated network to schedule in detail, from the ordering of basic raw materials to the setting up and manning of plants and facilities. Planning and control is not practical if based on scheduling from the top. But it can work if based on commitments by the buying public that can be followed through, and on the basis of which plans can be made with confidence. With a potential free market which could be made to exist by the year 2000 in an automated, electronic, rapid, all-embracing on-line form, planning that includes the consumer would be possible by the entrepeneur. This is an entirely new form of free enterprise, different and quite superior to the 19th century form. Capital investment in an environment of a fast-responding customer, an "electronically" informed, interested, active market, makes possible increases in the profit-to-risk ratio. Given the opportunity to participate in an accelerating free enterprise which advanced technological systems make possible, then we might expect people to enthusiastically pursue this path. Ideas for new products quickly disseminated would beget other ideas. Efficient scheduling for production and distribution would leave more resources for risk-taking.

It is equally important to note that a strong role for the government will exist. This is to provide service and to referee, control and assure objectivity, honesty, and opportunity in this free market which depends on such a huge national electronic information service. The government will be so busy implementing this service and it will be under such pressure from the voters to further expand individual participation in the free market, both as consumer and risk-taking supplier—everyone will want expanded opportunity to participate and benefit—that we will not need to worry about the government's seeking to plan the economy from the top.

In the future, technology does make possible a kind of automated socialism, a regimented, government-controlled economy, poor as that might be. But it also makes possible a government-aided, unprecedented level of genuine free enterprise with creativity, incentive, competition, and individual initiative carried to a new golden era of opportunity for man.

THE IMBALANCE OF TECHNOLOGICAL AND SOCIAL ADVANCE

The growing myths and misconceptions about the impact of computers and other advanced technological additions to our society have their overall foundation in the mismatch between rapidly accelerating scientific progress on one hand, and lagging social advance on the other. This imbalance is shown in our having developed an ability to release tremendous amounts of energy virtually able to destroy civilization before we have social maturity sufficient to preclude this possibility. We are on the frontier of radical advances in biology that can conquer diseases and prolong human life, while still socially unable to handle birth control, and thus we must be continually concerned with the problem of population explosion. Our large space program, justifiable as it might be for research and commerce, arose not because of an appreciation of these factors by the citizenry, but because of a prestige race with another nation. Finally, we have the possibility through automaticity of providing easily for man's material needs, but we are afraid we may not be able to handle the subsequent dislocations. It is these inconsistencies resulting from our socio-technological imbalance that are producing problems today, and not scientific progress.

The computer is not the source of imbalance, it is a tool that can accelerate civilization's progress and bring technology and society into alignment. Computers give us more brainpower. Properly used they can help us increase our natural brainpower by improved education. They can increase our utilization of man's mind. They can give us tremendously greater informational-intellectual capacities. A man-machine partnership, with the computers handling the mundane, rapid processing of data and providing instantaneous display of information where it is needed, allows man to rise to the higher intellectual pursuits. That is, the computer will make us smarter. Perhaps, we will even become smart enough to broaden our humanistic perspective. Then we can apply scientific innovation universally for the moralistic as well as economic good of our society.

Computers and education

R. W. GERARD

Dean of the Graduate Division
University of California at Irvine

Since I am an enthusiast, rather than an expert, in the computer field and have been asked to talk about the future, anything may happen; but it is reassuring to find experts almost as far out as myself. In fact, it would be unwise to look only at the immediately available technology; and I have often thought of the early days of aviation, when Billy Mitchell tried very hard to convince his superiors that the airplane had a future in war and, as you know, was essentially cashiered out of the Army. He did induce a cavalry general to give airplanes a try, but the experiment was a failure. One of the "crates" of the day flew over a polo field while officers were playing and attempted to hit players with oranges. No hit was scored, which was pretty conclusive proof that there was no future for aviation in warfare!

When I became actively interested in this problem something under three years ago as we were beginning to plan for U. C. Irvine, the experts in the field whom I consulted were of limited optimism. They said, "Yes, the things you want to do will be possible in ten years." Two years ago, taking another sounding, I was told, "Yes, the things you want to do will be here in three years." Last year, at the IEEE, the speaker before me in effect said, "Look, we have available, now, large memories with rapid access, parallel processing, multiple access to the computer, improved input/output terminals, improved (richer and easier) programming languages; and I find no roaring demand for them." I was happy to get up after him and say, rather emphatically, "The roaring demand is here." Now, at this meeting, many of the items which even a year ago were merely potentially available are on exhibit and are in actual use in at least the developmental, if not the completely operational, stage.

Actually, the problem clearly before us today is minimally that of hardware and maximally that of software. This is discouraging to many people. It may take as much as 200 hours of an expert's time, with some additional programmer time, to program one hour of effective tested-out computer-aided instruction. This seems at first, indeed, a devastating problem; but second thoughts are more hopeful. The languages for programming are still relatively primitive, although they are approaching basic English, and will become more efficient. But even if they did not, please consider that since the invention of the earlier technologies of communication—of which language itself is the main one—it took several millennia before man achieved writing, let alone paper and printing; and the software composing a modern language, English or any other, and the entire literature now available in that language, could be replicated on computers (reduced to computerized handling, put in appropriate memories, and so on) in something like 10^{-2} as great a time—and its usefulness probably be made 10^{2} times greater. A gain of 10^{4} is worth the time and trouble and, as I hope to show, there should be adequate funds to do the job.

Incidentally, as Dr. Ramo pointed out, people fear that computers will replace them or, as someone put it a few years ago, that artificial intelligence will replace natural stupidity! What we are really facing, of course, a symbiosis of both, combining the attributes of great speed and vast memory of the idiots that we call computer systems with the imaginative, creative, idiosyn-

cratic, pattern-forming capacities of the human brain and mind. I have felt for some years now that, of the evolutionary epochs in the rise of man from the time his brain became good enough to develop a communal culture and pass on experience, which is essentially education, the major advances were: first, the ability to have symbols at all (which may have been pre-human); second, the organized symbols of language; third, the organized tested symbols which constitute science; and fourth, the prosthesis to extend the mind which is the computer. I would put this present technological evolution as equal to the development of language; considerably more than the development of printing.

Now a few words more on the evolutionary aspects of learning and the role of education. Learning is, in effect, modifying one's behavior in the light of experience; and education, formal education, is an effort to organize the experience to which an individual is exposed so as to develop a maximal change in behavior and capacity along certain desired lines. This is nothing new nor unique to the formal level; indeed, the evolution of the nervous system itself is very clearly the consequence of responding to the species' experiences. There is good reason to think that the cerebrum developed in response to the steady barrage of the brain by nerve messages elicited at improved receptors, particularly the distance receptors. Originally smell, later vision, and to a lesser extent hearing, are the inputs that give an animal warning of changes in its environment before a predator is upon its tender skin or before it flushes its own prey. This maintained input supplied the exercise which increased the brain in the first place and, as current analysis of the anthropological evidence suggests, this process was greatly accentuated when man began to use tools—which rather explosively further enlarged the brain.

Now, in the development of the brain of the individual infant, exactly the same sort of processes are occurring. It has been demonstrated by a rich array of varied experiments that an infant deprived of certain experience fails to develop the capacities for handling such experience. A baby chimpanzee kept in the dark, or even with milk glass over its eyes so that it sees light but not patterns, until it is some weeks old (nothing done to the eye, nothing done to the brain) may never learn to discriminate patterns in the environment. It remains functionally blind. Conversely, if one enriches the experience of a young animal, a rat in this case, there is actually a hypertrophy, a thickening of the cortex of the brain; just as exercising muscles increases their size.

So evolution has depended on the impact of the environment on many generations of individuals in the species; individual development, on the impact of the environment on the single individual. It is too large a story to go into here, but as a biologist, I would maintain that the major theme of all biological evolution has been the increasing responsiveness of organisms to the environment, the developing of malleability—not merely being able to learn in response to the environment but learning how better to learn. The same is true in the education of the individual child; it also has to learn to learn and can thereby learn better. I would, therefore, add to the previous statement, that the computer will be a great supplement or prosthesis to the human brain, and that the computer can also be a great developer of the brain. Here is a tremendous opportunity for the future.

Formal education probably should be traced back to the 12th century when the great medieval universities began, led by the University of Paris. We may forget that the students there were often subteenagers, so a university was not quite what we think of today. The widespread present formal education, essentially in reading, writing, and arithmetic, was not only nonexistent but would have been utterly useless before printing and the widespread availability of books made it possible to read and therefore made it worthwhile to learn to write. It is only within five centuries that our current widespread literacy became possible and was achieved; even reckoning, especially multiplication and division, was practically unknown three centuries ago but was a special skill of highly trained expert clerks. Perhaps because education got started relatively early in the area of behavioral institutions, perhaps because of the inherent difficulty of the subject matter—the vagueness of the resources and even of the goals, the uncertainty of the outcomes, and the extremely varied artistic and uncontrolled methods of procedure—I submit that this particularly important area of human activity (perhaps the most extended and continuous and largest activity of man; wars are bigger but they come and go, science is more continuous but less extensive), education, has lagged very far behind other areas of applied behavioral science. It is not evaluated, it is not analyzed, we really don't know whether or not we have been really achieving anything with all the time and funds and human labor that have gone into this field. The most important impact on education of computer technology (and I use this in its broadest sense) will probably be by supplying a tool for finding out what we are doing, for turning anecdotal impressionistic answers into scientifically testable ones, and so turning what has been almost purely an art into a respectable science—and without eliminating the artistic aspects either. Research in education, advances in educational under-

standing, and education as a behavioral science will be, I think, the most important outcomes.

Let me give you a concrete example. There are two rather important competing theories as to the way learning occurs, either as a steady incremental process or as a step-function increment—differential or integral advances. The two theories make precise predictions as to the best reinforcement schedule for learning. According to one, those items on which an error occurred more recently should be presented oftener; according to the other, there should be no such emphasis and all items should be presented equally but at random. It would be impossible to perform an experiment of this sort without the use of computer technology, computer aided instruction; with this, Atkinson at Stanford is doing just such studies with children.

Let us, then, examine in a little more detail the impact of this new information handling technology on the procedures and the institutionalization of education and try to foresee some of the social outcomes of these changes. To give you a clearer picture of what I have in mind, I shall present our activities and goals at UCI—it will be clear which are which. The campus became interested in the possibility of really interweaving these two great information handling systems. A university is primarily a system for storing, retrieving, processing disseminating, and creating information (research, which creates information, being less present in lower institutions); and computers do exactly the same, even creating information in the sense that mathematics, though a tautology, creates usable information by manipulating existing knowledge and assumptions. Clearly they are made for each other and our hope was to build a total system for information handling by combining the resources of both. We set up, for example, a computer "facility" rather than a computer "center," to imply an interpenetration rather than a boundary.

Although we opened our doors to students only two months ago, a reasonably powerful computer resource has been on hand in trailers for a year with the cooperation of IBM. Dr. Tonge, Director of the University facility, and Dr. Kearns from IBM have led this activity and the development of CAI. There are already in action 18 on-line time-sharing consoles, and a group of some 20 professional people involved in the computer activities, penetrating all parts of the university. On the administrative side, we look to a total systems use in enrollment, in finance, in faculty and student records, in plant upkeep, in classroom assignments, etc.; and several lines are now active. Of course, the system will be used at all levels of research, up to social simulation systems. The bookkeeping aspects of the library—handling of materials, accessions, charge-outs, etc.—are being prepared for automation and we look towards processing the information rather than the documents that handle the information, to get information on an on-line rather than batch processing basis. Beyond all these, and our major thrust, has been a concern with the possibilities of computer-aided instruction in the educational process, itself; and it is only of this that I shall continue to talk.

Let me give you a vision of what is to come, whether in years or decades remains to be seen. I would suggest, however, that the reason the experts in the field almost invariably are rather pessimistic in their guesstimates of the future is that, while each is acutely aware of the bugs that have troubled and delayed his own developments, he is likely to overlook the countervailing effect of parallel activities by many people trying to do similar things. The total impact of this collective thrust is much greater than the frustrations and delays in the individual case, so advance moves faster than the general expectation.

I like to think of the total system as a sort of sandwich, of data bank on one side and users on the other, interacting via the information processing apparatus. The data bank includes not only the ordinary internal and external memory devices of the computer but the mobilization of any kind of material which can be recorded—on video tapes, movie tapes, micro forms, slides, phonograph records, thermoplastics, what you wish. This great data bank need not be located physically in relation to any particular processor but be made available to many by networks. Such banks can be accumulated in a very few places (conceivably in only one except for the danger of loss or damage) and, by the communication networks, be made available at least areawide to all sorts of users in all aspects of education. On the other side, I see the user sitting at a terminal, a console or carrel, at a convenient location, including his own home, and able to communicate in and out not only by typewriter, by cathode ray screen plus light-pen, and by voice, but also doing these with relatively simple buffer arrangements and using an adequate communication language. Beside the individual user, group interaction should be handled—a seminar with the instructor and a dozen or more students interacting audio-visually via a communication network and controlled through the computer processor, much as if all were in the same room.

The processor itself I like to call the tutor because here is an agent potentially updated in subject matter; and learning improved heuristics of the precise educational techniques to be used for this or that kind of individual with this or that previous training and with this or that temperament—a hostile youngster who likes

to fight back or a passive one who must be coddled and brought along. The tutor will have built into its own memory a detailed knowledge of each student with which it is working, in terms of that individual's background, personality, and achievement in the particular field; will be able to give the tutee immediate individual attention and to do so without threatening and with infinite patience.

Such an entry of computer technology into the educational process will have far-reaching and crucial consequences. I will not have time to go into all of those but, fortunately, the problems have already been discussed: the danger of regimentation; the danger of Big-Brother in "1984"; the danger of depersonalization or dehumanization or social anomie—existentialists insisting that people will lose their identity, individuality—all these certainly need more consideration than I can now give them. At least note that the book, itself, interposed between humans has not dehumanized them, nor has Othello as played by Olivier on the movie or TV screen; nor has mechanization of the kitchen ruined the home.

Now a brief look at costs. If it took, say, a hundred hours to program one hour of computer-aided instruction and this program were used by a hundred different teachers only once, the time cost would balance. In terms of financial cost, I have picked up a few figures here and there. For example, in the elementary grades it costs 27 cents per pupil per hour for teaching (this is, I believe, a national figure; the cost in California is nearly double); the work (including all development) cost $1.00 per hour per pupil per simple terminal according to Atkinson. But vastly more is done in that hour; in fact, it is reported possible to teach a child to read in 200 hours on a terminal—a cost of something like $200 per child! Since only a seventh of teacher-children contact time is spent in teaching, say five hours a week, it is easily understandable that great savings in student hours are possible. Further, if one teacher taught 25 children to read during one full year, this would cost well over $200 a child in her salary alone. There is much reason to believe that we could squeeze as much as three years out of the K to 12 period of schooling and not leave out anything of worth. In effect, during the 10th, 11th, and 12th years students are doing nothing productive in society and are costing a great deal of money; cutting these years is estimated (Machlup) as giving an annual saving of $15 billion. The cost of computerizing the whole of education, bringing all the resources—all libraries and everything else—into a machine-handable form, building the necessary programs for very rich Socratic tutorial interaction with students even at fairly high levels, would be paid for in very few years.

At the university level, if all undergraduate teaching were done solely on terminals (which, of course, would not be for a long time and probably would never be desirable), and if one achieved a gain factor in time of three to one (a gain of over five to one has been claimed in some early studies in this field), four courses at present requiring three hours each a week would need four hours a week on one terminal per student. If a terminal were used only 40 hours a week it would service 10 students; actually, longer use is easily obtained; so, say, all the work of 15 students could be carried on a single terminal—for 10,000 students less than 700 terminals would be sufficient for the whole educational process. At a student-faculty ratio of 15:1 and an average salary of only $10,000, teaching cost alone would be nearly $7 million for 10,000 students. The largest present hardware systems rent for $0.5 million a year and require a like sum to staff; double this for capacity and add another $2 million for terminals, and the cost is still half that of human teachers. Further, if three man-years are needed to program one course, and 100 courses would cover the bulk of undergraduate needs, all this software could be produced for some $3 million. Even if obsolete in one year (three is more reasonable) this cost is trivial if the system is widely used.

Well, aside from the financial aspects, the advantages of such a learning arrangement are only to be enumerated to be recognized. For the student it offers: (1) better and more comfortable and faster learning—he can time his learning experience at his convenience, go at his own pace and catch up missed time; (2) better teaching at many levels and in many areas; (3) particularly important—personalized tutoring, individual attention (I remind you, what is so often forgotten, that Thurstone's original study on the primary mental abilities showed ratios as high as a hundred to one in favor of child A over B for ability 1 and the reverse ratio of a hundred to one in favor of child B over A—the same two children—in ability 2; there are clearly fantastic differences in human beings and it is high time that we stopped batch-processing them through the educational machine! Let them take the initiative for actively learning in their own ways); (4) automatic measurement of progress, by keeping appropriate records of responses; when the course is finished, the examination has been taken, and examination neuroses are bypassed; (5) vastly richer materials, demonstrations, exhibits, travel material, on-site work at archeological excavations or ocean bottom laboratories, are available for presentation; (6) more sophisticated problems can be included in instruction even to the level of simple research, lift-

ing the drudgery of sheer repetitive computation.

For the teacher, the system: (1) also takes away a great deal of drudgery and repetition; (2) allows the teacher herself or himself to be updated effectively, without allotting a summer or a year for subject matter refurbishing every three or five years; (3) encourages frequent changes in the actual material used; (4) makes much more time available for real teaching—recall the estimate for grade school that teaching contact between teacher and student averages less than 15% of the time they are together.

Besides student and teacher gains, there are wider social goodies: (1) the very best materials can be produced by master teachers; (2) these can be used widely and repeatedly at least for a limited time; (3) individual modifications in the program can be made almost at will, in contrast to the delay and pain of a new edition of a book; (4) the great information systems can be tapped freely; and (5) perhaps the most important of all, the desperate shortage of teachers can be relieved. In "teachers" I include good teachers or average teachers or almost any warm body (physically, not psychologically); there just cannot be enough humans for the jobs to be done. Think, further, of the teaching needs of the emerging countries which must get themselves instant education. If one teacher can teach a few dozen students who then become teachers who can teach a few dozen, the multiplying or avalanching effect is just impossibly slow; so that some of these countries are making a quantum jump into advanced technologies—just as some countries earlier went from bullock cart to airplanes and bypassed the wagon and automobile. In our own country, autistic children—who rock all day, insulated from the world—have been led into participation by interacting with computers when even master teachers have failed (nonthreatening objects, infinite patience with repetition); and the vast needs of our Headstart program, to bring mere symbolic thought to underprivileged babies now growing up practically without language, cannot be approached with the supply of human teachers present or future.

Even beyond the immediate teaching possibilities are the exciting social outcomes for education. I shall briefly mention three. First, there will be greatly increased flexibility not only in handling the individual child, but also in handling the materials. If one can break the teaching sequences into relatively small units then, as with a Mechano set, a few kinds of pieces can build a great variety of structures. Instead of having many different courses in statistics—one for psychologists, one for biologists, one for public health, one for engineers, blocks of basic elements and particular uses can be put together in well-tailored fashion. This also means that it should someday be possible to get away from the lock step of the semester or the quarter system. Large course blocks should become relatively meaningless as each student goes through a learning experience cut to his shape more or less continuously.

A particularly important consequence could be the separation of the function of certification from the function of education. They are now part of the same process; and getting good grades, in order to enter the next school and get good grades in order to enter the next one above, has so come to dominate the whole thinking of the students and the teachers that whether one gets an education or not has often been pretty much forgotten. Progress can be certified and mastery tested in a much more certain and objective way with these newer technologies, leaving the process of real education (and with teachers used in it especially) to occur as needed.

Another important outcome, already implied, is the possibility of spatial dispersion of the learning experience. If certification is covered, then actual class attendance to learn becomes immaterial. I strongly expect that, in the not-too-far future, there will be an opportunity for the individual to interact through a console with the great array of knowledge and even with other humans, so there may not be geographic entities like a campus. I am well aware of what this means—you cannot have football and orchestras without coming together; in fact, it has even been stated that "it takes two to tango"—but for dealing with ideas, physical contact hardly seems needed; and high-level seminar-type interaction is possible over video conference hookups.

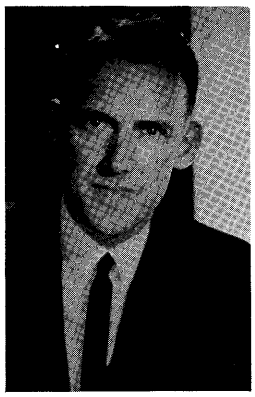
Teachers have been doing a great deal more than merely help develop the information processing capacities and resources of their students; they have been friends, they have been hero figures, they have supplied motivation, they have been shoulders on which to cry. All these things are very important but perhaps they are not all functions of the same individual at the same time; so I see the possibility of again splitting the separate roles. Indeed, the medicine man once served his primitive community as priest, lawyer, doctor, teacher, and entertainer; these have been separated into different professions, but have sort of collapsed back together in the teacher and now may again be separated out. In fact, I think it likely that a very different kind of good teacher will come into being and that new sorts of people will surge into the field of education. The great teacher of the future is likely to be more like the author or the composer or the director than he is, as at present, like the performer or the actor or the concert player.

Well, rather than continue about changed content and goals, let me close with just one thought. Man truly no longer lives in an outside world. As the cells of the body in the multicellular organism have created an internal environment in which those cells live, and which is made possible by the collective action of the billions of cells, each group performing its own role in the body organism, so men, in the same way, have created societies, groups of individuals in a multi-individual epi-organism, have created an internal environment of the society, a culture if you will, in which we live. This is also regulated against extreme swings; we are now little concerned about the physical and biological problems of the environment, there is no worry about wild beasts, we go to the supermarket for our food; houses and clothes protect against weather; we are conquering disease. The pressing environmental problems with which we live are those that man has created for himself—by increasing the ease and frequency and range of communication, the number of people who communicate, and the richness of the material which we now can communicate to each other. We are rapidly raising a sea of information in which we must either swim or drown, and the way we must swim is by enhancing the problem-solving resources of man and society.

The new computer technology will allow effective studies of human ecology; of the distribution of, and environmental influences on, physical and mental health and disease; of employment needs and their projection, so that appropriate special training of properly selected people can produce the round pegs for the round holes. But of the many opportunities for aiding man to handle himself collectively, in my judgment, the improved teaching of the young, to be effective members in society, is the greatest of all.

I cannot close without quoting to you one verse of my favorite poem, O'Shaugnessy's Ode:

> We in the ages lying
> In the buried past of the earth,
> Built Nineveh with our sighing
> And Babel itself in our mirth;
> And o'erthrew them with prophesying
> To the old of the new world's worth;
> For each age is an age that is dying
> Or one that is coming to birth.



Computers: The physical sciences and medicine*

JAMES V. MALONEY, JR.

Department of Surgery/Thoracic Surgery, School of Medicine
University of California at Los Angeles

Although computer science is in its infancy, it has already contributed significantly to society in the fields of business and commerce, communication, exploration, and scientific discovery. In contrast, the contributions of computers to medicine have thus far been minimal for reasons which I shall discuss in a moment. It is this lack of application of computers to medicine which makes medicine one of the most fruitful areas for the computer-oriented scientist. I can promise you the rewards will be great for those of you who choose to apply your talent for the benefit of human health and welfare. There are two reasons for this:

1. The systematic methods of scientific thinking which naturally lead to success in the application of computers to a scientific discipline have already been developed, and they have proven phenomenally successful in such fields as high-energy physics and molecular biology. Discovery in the field of medicine waits like a ripe apple to be plucked by the computer-skilled scientist.
2. Society will generously support your efforts. The value judgment that society places on health and longevity ultimately is the same as that which the individual places on them. I have yet to see a man who is dying of cancer of the lung, whether he be laborer or corporation president, who would not trade all of his money, business, commerce, communication, and transportation for 18 more months of healthy existence. Therefore, the computer scientist who devotes his effort to promoting human health and welfare will be both generously supported and greatly appreciated by his fellow man.

What do we expect computer science to do for medicine? The daily newspapers would lead us to believe that computers will diagnose disease, store medical records, interpret X rays, and perhaps remove a patient's appendix untouched by human hand. Perhaps this will come to pass—but I do not find the possibility intellectually exciting.

Each of you has a much more important attribute to bring to medicine: the problem-solving technique of the physical scientist. You would be amused—and I embarrased—to see a full exposition of the way most of us in medicine go about the business of scientific discovery.

The difference in problem-solving techniques of the physical scientist and the physician is best illustrated by my favorite analogy. Isaac Newton, after watching the red apples fall from the tree for some weeks, finally formulated a generalization which described the behavior of falling bodies. A little boy came up and said, "That's all right, Dr. Newton, for falling red

*This work is supported by grants-in-aid from the United States Public Health Service (HE-05357, GM-13242, and FR3).

apples, but how fast do acorns fall?" Newton is said to have given the boy the formula $D=\frac{1}{2}gt^2$ and said, "Run along and figure it out for yourself, sonny." A physician, to answer the problem about falling red apples, would instead have gone to the store, bought a bushel of apples, a stopwatch, paper and pencil, and a slide rule. He would have dropped the apples one at a time from the top of the apple tree and carefully observed and recorded the time it took them to reach the ground. At the end of that time he would have been able to say that it was 2.3 seconds with a standard deviation of $\pm.25$ second. If the same little boy had come along and said, "That's fine, Doctor, but how fast do acorns fall?" the 20th-century physician would then have run down to the store, bought a bushel of acorns, another stopwatch, paper and pencil, and repeated the same procedure. The modern physician is inclined to gather masses of random information without ever defining his question and its answer in exact physical terms.

As Vincent Dole of the Rockefeller Institute so colorfully puts it, "Physicians are obliged to swim in an ocean of detail, surrounded by great truths in high dilution."

The physician already has available more facts than the human mind is capable of bringing to bear on the solution of a single patient's problem or in the performance of a single medical experiment. "Like gold in seawater, facts lose value when mixed with irrelevancies and can be recovered only by special techniques of analysis."

It is the conceptual approach to problem-solving which I believe will be the computer scientist's greatest contribution to medicine. John Platt of the University of Chicago has attributed the stunning successes in the fields of high-energy physics and molecular biology to the regular, systematic application of Francis Bacon's system of inductive inference to the design of experiments. The physician has much to learn from the physical scientist in this regard.

There are many impediments to the introduction of computer technology into medicine:

1. There is a communication barrier. Unfortunately, the biologist is poorly equipped by virtue of his education and thought patterns to make use of the physical sciences.
2. Physicians are understandably fearful of machines and methods they don't understand.
3. There are limitations in both hardware and software. For example, we can train a medical student to recognize appendicitis, but we can not train a machine to do so.

It is well recognized that appendicitis is frequently characterized by nausea, abdominal pain, fever, and a high white blood count. And yet a medical student will not fail to diagnose appendicitis even if only one of these four cardinal signs is present. On the other hand, he may make a correct diagnosis of intestinal flu even if all four of the signs of appendicitis are present. Twenty medical students may all arrive at a correct diagnosis of appendicitis, but each one may do so by a different logical pathway. In contrast, we seem to be able to approach the problem by computer only by (1) Beysian statistics, which give us a probability of the diagnosis of appendicitis; or (2) sequential branching logic, which usually leads us to the wrong diagnosis.

Several problems with our hardware and software seem to be:

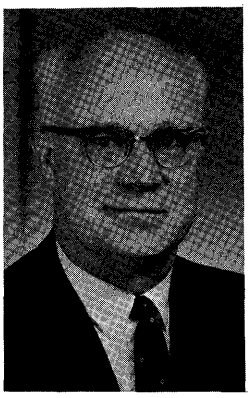
1. Computers do not have 10^{12} bits of storage like the human brain.
2. Computers do not have redundancy like the human mind.
3. We do not understand the pattern-recognition of the human mind sufficiently to be able to duplicate it in software.

Neverthless, I am encouraged. The educational background of our medical students is improving tremendously. For example, in our current freshman medical student class, 90% of the students have completed undergraduate courses in calculus. Unfortunately, these freshman medical students just out of college have another four years of medical school, another five years of postgraduate training, and several more years after that before they become productive and begin to reap the benefits of their physical-science orientation.

In our Department of Surgery at UCLA, we are overcoming these impediments, bringing together both physicians and physical scientists to work in cooperation on common problems in a medical environment. A pilot program has been established with Drs. Edward C. DeLand, James DeHaven, and Norman Shapiro of The RAND Corporation. By supporting this program, The RAND Corporation has given us in medicine a unique insight into the potential contributions of the physical scientist to biology and medicine. We have taken engineers with advanced degrees, brought them into the University hospital, and started them on the road to a Ph.D. degree in biological science. We are introducing physicians who have had from two to five years postdoctoral training in medicine to mathematical modeling and other computer techniques. The physical scientists and physicians work side by side in the same laboratory. In this way we achieve some of the same

benefits as if each individual were educated in both sciences. Thanks to the progressive attitudes of the National Institutes of Health and its civilian consultants, it has been possible for us to fund several million dollars' worth of hardware and other facilities to bring medicine and computer science together in a hospital environment. Although the program is new and it is too early to judge it critically, we are tremendously excited by the strides that have been made by bringing computer science and medicine together.

During a recent 36-month period, 4 Nobel prizes in physiology and medicine and in chemistry, were given for computer-based discourses. From this evidence alone it is apparent to me that computer science is going to be at the forefront of medical discovery in the coming decade.



Impact of computers on retailing

C. ROBERT McBRIER

Vice President, Finance,
Woodward & Lothrop
Washington, D.C.

My presentation today will be confined to systems in retailing that are either in operation or in the development stage. There is little point in describing retailing as seen through the eyes of a customer since a store is a familiar sight to each of you. Retailing as seen through the eyes of a computer is quite a different picture however. For you and me, who are interested in cybernetics, it is an equally exciting one.

I deliberately used the term "eyes of the computer" since optical character recognition is essential to implementing the concept for use of computers in retailing, just as magnetic character recognition is basic to banking.

I do not have the time to round out a full concept for use of a computer in retailing in the half-hour allotted to me, so with your indulgence I will refer only to the major systems.

There currently are very few "real" time applications in retailing and because of the economics involved there is primary interest in on-line systems supplemented with some real time. The term "right" time is now being used to identify this combination of systems.

The primary files in use are:

Item merchandise
Resource
Employee
Customer
Financial data

These files are addressed by numeric codes containing a check digit. Since data often has to be referred to outside the computer in an alpha sequence, such as customer records, we are confronted with the need to establish a numeric system with gaps to realize both an alpha and a numeric sequence. With the input of numbers in a data recorder by thousands by salespeople in an environment that is not conducive to accuracy, check digits are required to maintain a satisfactory level of performance.

For years there has been recognition that a "point-of-sale" device for input of data relative to a sale of merchandise is essential to an advanced system. At Woodward & Lothrop we have installed more than 800 Data Recording Electric Accounting Machines (DREAM) throughout all of our 11 stores. This equipment performs all of the functions of the traditional cash register, but with the introduction of stylized font to print the journal tape, a non-add key for entry of descriptive data, and a few other changes, an on-line system has become available to retailers. The journal tape is read each night and the following data is available for each sales transaction:

Salesperson's number
Employee number—for Calculation of discount
Department
Class
Item number
Amount
Customer number
Type of sale

There no longer is any limitation on salespeople or the merchandise they record in a register.

Department, class, and amount are entered in one pass. Other data, such as salesperson number, customer, item number, is indicated with a supporting code number and the data key.

The equipment is not limited to the entry of sales data since it acts as a communications device from the sales area to the central information center. Other data such as the following can be entered with appropriate codes for identification.

1. Time of day—to determine sales by periods of day.
2. Stock counts—to update perpetual inventory records.
3. Employee time of signing in and out—for payroll.

Variations of this equipment include the capturing of data on magnetic tape, punched tape, or punched cards in a central area and punched tape at the point-of-sale.

With the installation of these registers supported by computers, Woodward & Lothrop overcame the hardware limitations for applying management sciences to retailing. Effective supervision and discipline now remain the limiting factors. In order to implement these we make available a report daily identifying by registers all salesperson errors. The computer currently performs 36 audit checks and as additional data is introduced the audit routines will be increased.

I took the time to describe the "point-of-sale" device in some detail since it does constitute a major breakthrough for application of computers in retailing. I would now like to direct your thinking to the files that will be maintained.

ITEM MERCHANDISE

No one system is adequate for the management of a department store inventory and we often characterize the item as being one of the following:

Staple
Fashion
Big Ticket
Mixed

Generally speaking, systems supporting different classes of items will include the following:

Staple

1. Mathematical models are developed from past history of the same or related items which can be used for forecasting sales to support an order.
2. Sales activity developed from entry in the "point-of-sale" device, periodic stock counts or print punch marking tickets.
3. Purchase orders prepared by computer on a regular weekly or biweekly basis for delivery direct to stores or warehouse. In the foreseeable future there should be machine-to-machine communication. Retailer-to-Resource.
4. A punched card activated marking machine is projected for the near future. The marking, as well as a turn around document to support receipt of the merchandise, is then a by product of preparing the purchase order.

Fashion

1. Criteria is developed by price line, class, and department from past history to identify if a style is following a normal pattern and will sell out on schedule, if it is "fast-selling" and should be considered for reorder, or if it is "slow-selling" and should be returned to resource or marked down.
2. File updated daily from sales recorded by item number in "DREAM" point-of-sale device, print punch ticket or manufacturer's ticket. The item number will constitute the computer address, but all reports will include the full description of the item understandable to the buyer, vendor, style, etc.
3. A daily report of exceptions, fast and slow sellers, is prepared for buyer action.
4. A plan of model stocks by category of merchandise by store will be maintained by the computer to guide buyer in distribution of incoming merchandise.

Big Ticket

1. A reservation file is maintained by all items by location, which is interrogated from the selling floor. When the sale is completed, the necessary forms are prepared in the warehouse to pull stock and effect delivery.
2. The report is intended to keep buyer informed of stock status and sales activity to stimulate action on items that are not moving according to plan.

The item merchandise files will include a reference to the resource files, but the latter will be maintained separately for accounting purposes to effect payment of invoices, as well as for evaluation of overall profit performance.

Resource File

1. Ideally for the retailer, the item of merchandise should be marked by the resource in such a way that the record of its sale can be perpetuated mechanically.

Efforts to accomplish this are currently manifest in print punch tickets that have a uniform format for manufacturers of category of merchandise, as well as distinctive resource numbers. Interestingly, the more sophisticated systems of retailers conflict with this concept, since an item number is required to identify the sale, and this cannot be accomplished with a group of manufacturers.

2. In order to simplify payment procedures, retailers and manufacturers have cooperated with Dun & Bradstreet to develop a manual of distinctive numbers for resources. These are included on their invoices and used by retailers to address the file.
3. Where the order is on record in the computer, a turn-around document will explode the programs to adjust perpetual inventory records and effect payment of invoices.

EMPLOYEE

We are working to reduce the number of employees in the application of computers but they will always remain the principal element of success in retailing, since there is a personal contact at the point of sale.

One file will satisfy all the requirements of personnel, payroll and the supervisor for evaluation of performance.

Where the individual's performance affects the accuracy of the central information center, error records will be maintained which will be important in reviews.

The successful implementation of computer systems in a department store with decentralized input demands that the highest practical standards of accuracy be achieved.

CUSTOMER

With the initial application of computers to prepare customer bills, both retailers and customers were upset that the customer becomes a number on the records of the company. I well remember the letters we received at the time of our conversion. Some even punched holes in the tab card or tore it up to show their frustration.

Our plans for the immediate future are exciting because we are talking a personalization of the file beyond what we could ever achieve with clericals.

How does this sound?

1. At the time a customer opens an account she completes a questionnaire from which we build a profile of the family. If the questionnaire is not completed, then the computer will build a similar record from purchases.
2. The detail of the individual's purchases is identified by store, department, class, item number, price, and possibly size. The customer number is also input in DREAM.
3. The detail is stored for billing, but also summarized to indicate categories of merchandise and price lines purchased. The proper coding on the descriptive bill will enable the store to be selective in inserting direct mail advertisements that should have special appeal. Feedback of sales should maximize the effectiveness of promotions.
4. A profile of payments on account enables us to personalize collections and make decisions relative to authorization of purchases. We propose within two years to interrogate the computer from the selling floor by telephone and authorize purchases with a voice answer.
5. As complaints on service are received they can be accounted for and controlled to enhance service.
6. As we desire to follow-up on anniversaries, birthdays, big purchases, specific items, this can be accomplished.

FINANCIAL INPUT

Planning, control, and evaluation of performance are, of course, basic to the success of a business. We accept the fact that from the records of past performance, dynamic plans can be made; from the daily record of activity, effective controls can be maintained and management will be able to evaluate performance from reports prepared on an exception or other specific basis to inform the executive.

The timeliness and selectiveness of reporting has been immensely improved with use of the computer, but we look for exciting innovations in this field. As the effectiveness of management can be enhanced, this may well be the most significant contribution of the computer to retailing. We now accept the man-machine relationship.

Our immediate plans include variable budgeting. Expenses are now being reported compared to plan and last year. We propose to identify the extent to which each account is fixed and variable, and the cost re-

lated to workload beyond the fixed point. When this is accomplished, we propose to have the computer develop budgets based on proposed volume and then to determine the percent of budget realization based on actual performance.

DEMISE OF INEFFICIENT RETAILER

The computer will not accomplish for all retailers the benefits noted. For many years I believed the computer might help the independent retailer to be more competitive. I now appreciate that many of the concepts referred to are not available to the small retailer because he does not have the finances or the staff to implement them. Service bureaus and arrangements to share computers with banks or other businesses will help, but they can never match the efforts of the big chains or the large independents. The computer may well then speed up the demise of the small retailer, except as the advantages of the computer are offset by superior personal management. Certainly the future of the complacent, inefficient store is questionable.

Retailing however will get a bigger share of the customer's spendable income.

PRIMARY IMPACT OF COMPUTER ON RETAILING

I hope you share my enthusiasm for the use of computers in retailing, even if my reference to specific applications has had to be brief. In the few minutes left to me I would like to summarize this impact.

Inventory Management

1. We will operate with lower stocks and achieve a faster turnover.
2. Balanced assortments will be maintained with fewer stockouts.
3. Markdowns will be reduced and sales increased as a result of the above.
4. Buyers will be relieved of clerical functions to direct their atteniton to creative activities.

Customer

1. As profiles are maintained for each family, we will be able to render a more personalized service which is vital when the same merchandise is offered by several retailers.
2. Promotions will be more personalized.
3. Retailing should realize a bigger share of spendable income.

Credit

1. City-wide numbering is now developing which will be a future basis for combined authorization, billing, and collection, with material savings in postage and duplicated effort.

Simulation

1. As decision rules are defined and systems developed, we will simulate performance to determine the level of customer service we desire to achieve related to:
   a. Inventory investment
   b. Achieving ideal schedules for salespeople
2. Through use of management games, executive training can be expedited and an earlier indication of executive ability determined.

Management, Planning, Control and Evaluation

1. Centralization of functions with improved efficiency at lower costs will be realized.
2. The effectiveness of each executive will be materially enhanced.
3. Share of the market and net profit will be improved.
4. With higher earnings on invested capital, retailing will be more attractive to the investing public.

THE CHALLENGE

The promise of EDP in retailing exceeds almost every other business and the problems of implementation are just as difficult. Ours is a business of people—salespeople, buyers, clericals, executives, warehousemen, delivery drivers—trained to buy and sell merchandise and services to other people—the customers. All of these people resist regimentation and covet the independence to adjust to current situations. Our success in the past has been due in large part to the flexibility and judgment of these people.

The application of the computer demands that certain procedures be executed with a high degree of accuracy. The term discipline is now being referred to more and more as necessary to achieve higher standards of performance. As the management of a store understands the potential for computers and gives leadership to the changes required, it becomes an exciting experience to participate in a team effort where the results are impressive. With other immediate problems demanding attention, management effort is often directed elsewhere and the progress is slow.

The large chain has the advantage here since a staff of experts can be directed to develop pilot operations

in stores, which can then be extended to others. Enlightened management in a large independent store however can achieve ever more impressive results.

A premium is developing for the quality of leadership, knowledge and ability represented by you in attendance here today to accomplish the improved performance in retailing through use of computers that can be realized in no other way.

The application of computers to domestic and international trade

WILLIAM I. MERKIN
and
RAYMOND J. LONG

United States Department of Commerce
Washington, D.C.

INTRODUCTION

The Act establishing the Department of Commerce provides, in part, that the Department shall "foster, promote, and develop the foreign and domestic commerce . . . of the United States." These broadly described functions are carried out under the policy guidance of Assistant Secretary of Commerce for Domestic and International Business Alexander B. Trowbridge.

The Assistant Secretary is supported by a Domestic and International Business (DIB) organizational structure comprised of the Bureau of International Commerce, the Business and Defense Services Administration, the Office of Field Services, the Office of Foreign Commercial Services, the Office of Publications and Information, and the Office of Administration.

In the performance of their assigned functions, these DIB organization units must generate a wide variety of economic and commercial data in order to meet the increasing needs of government and business for timely commercial information. Trade regulations, commercial trade (both foreign and domestic), balance of payments, world markets, and the problems of businessmen and investors are of concern to the Department. The importance of gaining access to pertinent facts affecting national policy cannot be overemphasized. Timely evaluation of these facts is essential.

It is logical, therefore, that the Department of Commerce should look to automated techniques to satisfy the need for speed, accuracy, and economy in the acquisition and dissemination of commercial information. As a result the Automatic Data Processing Division of the Office of Administration, which is responsible for planning and implementing electronic digital computer and mechanical tabulating systems for the DIB organizations, undertook a DIB-wide assessment of potential automatic data processing applications. This study, which was completed in March 1965, identified nine areas as feasible for early automation. These areas are:

American Traders Identification
Foreign Traders Identification
Dispatch Loan Service
Unit Value Index
Export License Application Control
Tailor-made Foreign Trade Statistics
Economic Research
Automation of Domestic and International Business Publications
Statistical Data Bank

These systems do not employ what would be considered new and unique technology. Rather they apply proven techniques to new areas of data processing.

Four of the systems are now fully operational: the Unit Value Index, the Export License Application Control, the Tailor-made Foreign Trade Statistics, and the Dispatch Loan Service.

Three systems are partially operational: the American Traders Identification, the Foreign Traders Identification and the Economic Research System.

The remaining two systems, consisting of the *Commerce Business Daily* and the Statistical Data Bank, are in the systems design stage.

Together these systems represent the nucleus of what will eventually be an "Integrated Commercial Intelligence System."

AMERICAN TRADERS IDENTIFICATION SYSTEM

The Domestic and International Business area has established a master magnetic tape file of detailed financial and commercial data on more than 430,000 U.S. business establishments. Recorded thereon is referenceable information about a company's net worth, number of employees, sales volume, type business, products, and other business data. This system speeds up and broadens distribution, upon request, of commercial information, including international sales and other business opportunities, tailored specifically to the needs of the inquirer.

There are three basic classification of files for applying this system; the data on all 430,000 U.S. business establishments (called the master file), the data on U.S. business establishments interested or engaged in foreign commerce (called the American International Traders Index File), and the data on U.S. business establishments engaged in foreign commerce who would be willing to carry, "piggy-back" fashion, another company's products into the foreign market, utilizing their own distribution system and organization for this purpose.

Prior to the initiation of the master file, firms were invited to participate in foreign trade exhibits based upon manual search of a limited reference file consisting of information on approximately 5,000 U.S. business establishments. This was supplemented by personal knowledge of Commerce employees on firms which regularly participated in these activities. Now, by using a generalized computer search strategy based upon the theme of the fair, a search is made of the master file which has stored information on more than 430,000 business establishments. Experience shows that a usual search of the automated file yields on the average of 4,000 firms per fair. A mailout on one such output showed that 13% of the selected firms responded with an affirmative desire to participate. This was slightly more than 500 firms as compared with about 100 firms under the old manual method. As to speed, 99 fairs can be processed in one pass of the file and a listing of prospective interested participant firms retrieved—listed separately for each fair, based upon the theme. The pass takes five hours. Previously, it took from one to six months to prepare a list for each fair (depending on the theme and the information available).

The interest of American businessmen in foreign trade opportunities prompted the U.S. Department of Commerce, using a computer routine, to inquire of 340,000 U.S. business establishments whether they currently were engaged in foreign commerce, and if not, would they be interested in entering foreign commerce. It was estimated that about 30,000 business establishments would respond. Actually, almost 60,000 favorable responses were received. Using these responses as a basis, the American International Traders Index File was established. A second mailout was sent to those firms which requested volunteer commercial information. A total response of 25,000 is expected from this mailout. The ultimate file will carry information on products (by Standard Industrial Classification), country outlets, foreign trade representation, and other pertinent commercial information. The information furnished by responding U.S. companies will be maintained in confidence within the government and will not be published. This master file will be used for spotting trade opportunities. In addition, there are many firms manufacturing products which can and should be introduced into foreign markets but who do not have the capital or facilities for sales and distribution support to compete abroad. Certain business establishments already engaged in international commerce are willing to use their established sales and distribution organization to carry, piggy-back, expanded product lines for foreign sales. The challenge is one of introducing these two types. Inquiries from both types of firms interested in the piggy-back technique can be matched and satisfied by searching for products and carriers in this master file.

FOREIGN TRADERS IDENTIFICATION

Closely related to the American Traders Identification File is the Foreign Traders Identification File. The Foreign Traders Identification File will be used to develop foreign trade lists; to identify manufacturers, agents and distributors; to give leads on investment opportunities abroad; and to assist in our export expansion program by providing rapid retrieval of information on export markets by country, company, and product.

Using the country of Japan as a prototype, a magnetic tape file was established containing current commercial intelligence information on more than 4,000 Japanese firms. This information covers the name, address, products, size, capital, sales and other pertinent facts about

each company. From this an alphabetic listing was made on all firms; later, an alphabetic listing by product was produced; and still later, a profile on each firm was developed; and finally, a generalized search strategy for the retrieval of information from the file was programmed. Based upon our experience with the Japan file, systems modifications were made and plans for phasing in the remaining countries are presently being implemented. Arrangements are being made for the collection, coding, keypunching, and verifying of the input data to the system. Once fully established, the file will contain commercial intelligence information on more than 300,000 foreign business establishments.

The search strategy will be tied into the strategy used in the American Traders Identification File. This is in accord with our plan for a single search strategy for all commercial intelligence files.

DISPATCH LOAN SERVICE

The Dispatch Loan Service is a prototype of what will eventually be a large-scale information storage, retrieval, and dissemination system. U.S. Foreign service posts prepare dispatches on recent developments in products and with foreign business firms. These dispatches are sent to the U.S. Department of Commerce where they are indexed and a country file maintained. A monthly index of accessions is published and business organizations may request a loan of the dispatches, usually through a Commerce Field Office. The entire operation is currently performed by manual methods. The accession rate is about 600 dispatches per week.

The automation of this file will be in three phases: first, the indexing and coding of source documents to put them into machinable form; second, developing program routines for immediate use of data; and, third, developing and analyzing various techniques for application of data in a larger system. The larger system we are ultimately concerned with is the storage, retrieval, and dissemination of information on incoming dispatch communications received by the Department of Commerce. Approximately 2½ million copies of communications were received in 1964.

There are some ready-made descriptors to facilitate coordinate indexing for putting data into machinable form. In addition to the usual identifying information such as reporting agency, country, date, etc., we have the Standard Industrial Classification (SIC) for products, and have, among other possible descriptors, the "Harvard Marketing Mix" for the Marketing Factors. The incoming dispatches will be indexed and stored on magnetic tape. The monthly index of accessions will be printed on the computer and duplicated by reproduction techniques.

Dispatches will be reproduced for distribution on a request-loan basis.

After the file is operative for immediate use, other indexing techniques will be tried such as KWIC (key word in context), and consideration will be given to more sophisticated search strategies, such as use of weighing factors, links, and roles.

Using the firms that now use the loan service as a nucleus, a Selective Dissemination of Information System will be established. This information-sharing concept will assure timely dissemination of information to the business community.

UNIT VALUE INDEX

The Unit Value Index System produces commodity value indexes for both imports and exports. These indexes are used extensively by private industry and government agencies for economic analysis and forecasting, for the study of price movements of merchandise exports and imports, and for analysis of U.S. foreign trade volumes.

The system is comprised of three phases: the base weight computation for commodities which is produced at the beginning of each calendar year, the data tape which uses U.S. Census monthly commodity value statistics for the creation of the commodity value data tape for both imports and exports, and finally the computation of the indexes for each commodity by applying the data tape to the base weight tape.

It is important for businessmen to know the relative price of a particular commodity in the world market. This cannot be done effectively unless base weights are used to determine indexes for specific commodities. The Unit Value Index System does this by applying a computer routine and logic to integrate and synthesize available information.

The information used is published in many periodicals including the *Overseas Business Reports*. In addition, among the many private and governmental organizations which use the data are the United Nations and the National Bureau of Economic Research. Proper and effective analysis of these data assists the U.S. Department of Commerce's mission of developing a more favorable balance of trade.

EXPORT LICENSE APPLICATION CONTROL

The Office of Export Control of the Bureau of International Commerce, administers the authority provided in the Export Control Act of 1949 as amended, to control the flow of United States exports. Generally, this control is exercised by the issuance of licenses.

Approximately 700 license applications are received daily by the Office of Export Control.

The initial phase of the automated system for this application has been implemented. This initial phase consists of a complete reporting system (covering License Applications, Rejections, Issuances, Listing of Company, Commodity, Country, Quantity, etc.) with daily, weekly, monthly, quarterly and annual reporting cycles. Since the quarterly reporting cycle is of significant importance, we are conducting a three-month parallel operation. The parallel run will be completed on January 1, 1966.

After completion of the parallel operation, the next phase will be started. This consists of the processing of licenses by utilizing computer decision tables and other types of table look-ups.

The total system ultimately will include processing applications, generating reports and statistics, preparing pre-licensing lists, printing approved licenses, file maintenance, updating, and information storage, retrieval, and dissemination.

TAILOR-MADE FOREIGN TRADE STATISTICS

This system can best be explained by first considering the manual methods which were used.

The Business and Defense Services Administration (BDSA) in support of the U.S. Department of Commerce's export expansion program, among other things, reviews the impact of imports and exports in domestic business. Statistical indicators of progress and status in product commerce and marketing provide an invaluable tool for this review. Under the manual method the average division required as many as 536 different source documents monthly for study to develop its analytical product statistics.

Under the automated system, source data are procured from the U.S. Bureau of the Census in the form of magnetic tapes carrying monthly import and export information. These tapes are processed through a tailored routine which produces monthly for each division commodity and product information formats ready for review and analysis. Quantity and value of imports and exports by countries of origin and destination is included in the tables provided, in addition to cumulative data for the year in progress. Volume data for each import and export classification are also summarized.

This gives the commodity analyst one source document from which he can gather statistics. Timeliness is an important factor in all reporting systems. The statistical data are now provided the analyst within one week after the data are produced by the Bureau of Census. Under the manual system, it was weeks and in many instances, months, before the analyst gathered the statistics for use.

ECONOMIC RESEARCH

Currently, this system consists of a series of FORTRAN programs for statistical computations and correlations. These program routines are primarily used in the production of an annual economic study of major manufacturing industries featuring statistical projections and forecasts on expected levels of activity.

In addition to the FORTRAN programs, there are several programs for financial analysis. The financial programs include: companies listed by various search criteria, report on past financial data and ratios for specified companies, a report on the comparison of a particular company as it performs within its industry, a report comparing a particular company's sales and earnings against the industry and the economy, and a report of up to five companies showing financial and market information.

Consideration also is being given to the construction of econometric models to determine what would be the relative impact on major industries given varied economic conditions, to analyze cyclical fluctuations in commerce and marketing, and to provide productivity projections.

AUTOMATION OF DOMESTIC AND INTERNATIONAL BUSINESS (DIB) PUBLICATIONS

The *Commerce Business Daily* was selected as a prototype for developing this application because representative problems associated with automating DIB-type publications are featured in its production.

Federal procurement officers transmit information by wire and mail on U.S. Government invitations-to-bid and contract awards for publication in the *Commerce Business Daily*. Foreign trade opportunities are provided daily by the Business and Defense Services Administration. The *Daily* usually has eight pages of trade information, of which four pages are a procurement synopsis. Presently, the circulation averages 18,000 copies.

The purpose of this application is to convert the publication of the *Commerce Business Daily* to a computer-controlled printing process. Once accomplished, this will facilitate the production and mailout of the publication on a more timely basis, thereby providing opportunities for more firms to bid on government procurement opportunities. This application will pro-

vide the improved facilities required by the Department to fulfill, more adequately, its responsibility for the prompt and comprehensive daily publication of federal government procurement actions as required under Public Law 87-305.

This routine input received by teletype will go directly into the computer and be stored; other input will be edited and keypunched for storage. The computer will perform final edit, line justification, and hyphenation. The completed and formatted information will be produced on paper tape which will drive the "hot metal machines" at the Government Printing Office to create plates. The final makeup and camera operations, as well as the printing, folding, and mailing of the *Daily* would be performed in the usual manner.

When the Government Printing Office's high-speed photo-composition production system becomes fully operational, additional page makeup composition operations could be performed by the DIB computer. Fully prepared information could be transmitted by magnetic tape to the Government Printing Office high-speed composer. Film pages for the *Daily* could then be constructed as page makeup negatives ready for conventional composition, platemaking, printing, and binding. Ultimately, the system could expand to a total automated concept for all phases, including mailing.

As we have already stated, this application is a pilot project for several other Department publications (not yet selected) for publication by automated processes.

STATISTICAL DATA BANK

This application consists of developing a data bank for statistical analysis and data retrieval.

At the present time, the data bank consists of magnetic tape files of stored data covering commercial information on 430,000 U.S. business establishments, information on the financial history of 1,000 U.S. business establishments, information on tariff and trade agreements, and statistical information on U.S. imports and exports.

Currently, the automated application of these files fall into four categories; data retrieval, statistical analysis and correlation, statistical research, and file analysis.

Specific programs include file-profiles, information retrieval, and statistical applications.

Many of these techniques have been discussed under other applications so we will not repeat them here.

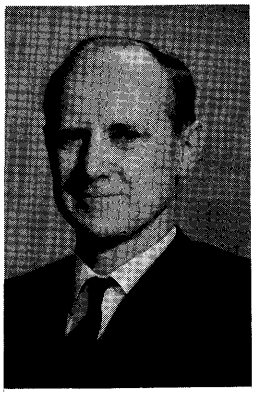
The Statistical Data Bank Application is the obtaining, developing, updating and utilization of commercial-economic data of a statistical nature or with the possibility of generating statistics from the data.

SUMMARY

The above described systems represent a cross section by technology and application.

As to technology, the applications include, among others: information storage, retrieval, and dissemination; photo-composition; data reduction; data processing; statistical manipulation; report generation; and decision making.

By application, we have a cross section of DIB organizations and a cross section of their tasks. The data are classed into two broad categories: narrative and statistical. The nine systems described form the basic ADP pattern. New systems will be tied into existing systems, and the existing and expanded systems will be extrapolated periodically. Eventually, the total system will become a mechanized integrated economic information system.



The role of computers in space exploration

C. R. GATES
and
W. H. PICKERING

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

INTRODUCTION

The modern digital computer has been fundamental to the space exploration program. Computers have profoundly affected almost every aspect of space technology, including spacecraft design, celestial mechanics, mission control, and the gathering and processing of data generated by the spacecraft. Indeed, the evolution and growth of computer technology is suggestively parallel to the growth in space technology.

Perhaps the most revealing way to examine our indebtedness to computers is to recall conditions before computers were available. Fifteen years ago our main project at JPL was the Corporal, a ballistic missile with a range of some tens of miles. The calculation of its trajectory took three weeks of intense labor; now we can compute the trajectory of a spacecraft to Mars in a few minutes. Formerly, solving a problem in control stability or structural design involved many hours of hand calculation and was likely to require extensive laboratory experimentation; now, many problems in these fields are solved more cheaply, more accurately, and more quickly by means of the computer.

In the last analysis, the effect of modern, high-speed computers on the space exploration program has been to increase performance—the performance of the spacecraft, with respect to the economy and precision of its design; the performance of the navigation system, in the accuracy with which the spacecraft can be guided to the moon or Mars; and the performance of the data gathering system with respect to the amount of data which can be gathered and the speed and accuracy with which it can be analyzed.

In my discussion I will describe some of the more prominent ways in which computers have been used in the space program.

TRAJECTORIES AND CELESTIAL MECHANICS

The most dramatic usage of computers in the space program is probably in the field of celestial mechanics. Celestial mechanics is the oldest of the sciences, and since a spacecraft in flight behaves like a miniature planet, our trajectories and navigation are based on this ancient and ancestral science.

Mathematically, celestial mechanics is concerned with the solution of Newton's equations. From Newton's time until very recently, solutions were obtained only with prodigious labor and by means of amazing and ingenious mathematical contortions. The digital computer and the impetus of the space program have profoundly affected celestial mechanics. Computer calculation of the trajectory of a spacecraft such as Mariner IV is quite straightforward, even though the numerical techniques used are sophisticated. In the selection of trajectories for Mariner, many hundreds of trajectories were calculated and analyzed in order to determine those trajectories which give the best accuracy, payload weight, and picture quality at Mars.

The computer also greatly increases the accuracy

with which a spacecraft such as Mariner can be guided. For the guidance of Mariner, the orbit is determined from radar data, and a small maneuver to be executed by the spacecraft is calculated. The computer permits these calculations to be carried out swiftly and accurately while the spacecraft is in flight; without the computer, approximations of much less accuracy would have to be used.

SPACE FLIGHT OPERATIONS: SPACECRAFT COMMAND AND CONTROL

The largest use of computers at the Jet Propulsion Laboratory is in the real-time processing of spacecraft data for command and control purposes; we refer to this process as Space Flight Operations.

Our Space Flight Operations Facility, or SFOF, will shortly contain three "strings" of computers, each string consisting of three computers, an IBM 7040, 7044, and 7094. Into the SFOF come spacecraft data obtained from tracking stations located throughout the world. The data may be from the spacecraft, concerning either the condition of the spacecraft or conditions in space, or the data may be navigational, referring to the position or velocity of the spacecraft. These data are sent in digital form over telephone circuits to the SFOF, where they are routed into the computers. After processing, the reduced data are presented to scientists and engineers for analysis and interpretation. The engineers conducting these analyses have access in parallel to the computer and are able to request in real time various programs, options, and methods of display.

Real-time command and control centers are also used extensively in the manned space flight program. The control center for Mercury was at Cape Kennedy, supported by facilities at Goddard Space Flight Center. The control center for Gemini and Apollo is at the Manned Spacecraft Center in Houston.

Two basic types of computer processing are carried out in Space Flight Operations. In the first type, the real-time data stream is simply separated, translated into convenient units and symbols, and presented directly to the engineer or scientist. This is the "real-time mode," and it has been the backbone of spacecraft analysis. In the SFOF at JPL, one computer per string is devoted to this function, which includes sorting, decommutating, formatting, and preparing printer and plotter output.

In the second type of processing, a quantity of accumulated data are analyzed and certain parameters extracted. An example of this type of processing is orbit determination, in which several hours or days of tracking data are analyzed in order to determine the orbital parameters of the spacecraft.

We have been one of the pioneers in time-shared multiple-channel computer usage, in which a number of users can simultaneously communicate with the machine and receive output. At times I think we would have gladly relinquished the pioneering honor to someone else; the problems in such a system—the interaction between programs, the difficulty in diagnosis, the intelligibility of a program only to the programmer who wrote it, the difficulty in reproducing a condition in which a failure occurred, the problems of meeting schedules and keeping adequate documentation—are well known. However, the system has carried out its flight mission reliably and successfully, the experience gained is valuable, and most of these difficulties appear to be behind us.

SPACECRAFT DESIGN AND SPACE TECHNOLOGY

In a spacecraft such as Ranger, Mariner, or Surveyor, subsystems derived from a variety of technical disciplines must all function together with tightly knit precision and harmony. Guidance and control, communications, science, structural, thermal, and propulsion subsystems must work together. Furthermore, in spacecraft design it is necessary to consider almost all of the physical properties of these subsystems—electrical properties, weight, mass distribution, heat generation and conduction, reflectivity, etc. Also, there are overall constraints of weight and volume placed on the spacecraft. And for a planetary spacecraft there is a totally inflexible constraint of time; the spacecraft may be launched only at infrequent intervals, necessitating accurate and reliable scheduling procedures.

The use of computers has increased the accuracy and speed with which the spacecraft and its subsystems can be designed. Using computer techniques to explore a whole spectrum of design possibilities, design parameters can be selected quickly and precisely. The result is a materially increased performance. Without sacrifices in cost, reliability, or schedule, the application of computers results in a spacecraft which will send back more and higher-quality scientific data regarding conditions in space.

As an incidental aside, it is interesting to note that our Mariner-Mars spacecraft contained four digital computers. These were used for science data handling, engineering data handling, command processing, and command sequencing.

The impact of computers on engineering technology, in fields such as structural design, circuit design, heat

transfer, etc., has generally followed a pattern of great interest and importance. Initially, computers were used to solve problems in their traditional form. For example, the solution to a differential equation would be obtained by hand in series form, and the series was then evaluated on the computer. In the next stage of development, the computer was given the differential equation directly, which was less work for the engineer and usually yielded a more accurate solution. Today, it is common practice to state the problem to the computer in terms of the end result desired, and the computer both formulates and solves the differential equation. Thus, the computer has been applied further and further upstream.

CONCLUDING REMARKS

In summary, we find that the computer has touched almost every phase of the Space Exploration Program. Spacecraft design is now based on computer analyses, and the result is a better spacecraft; furthermore, a steadily increasing portion of the design process is being carried out by the computer. In celestial mechanics, the effect of computers has been dramatic. And in space flight operations, all data gathered by the spacecraft are processed by computers Indeed, the number of computers we will shortly have serially in the data stream is startling; counting computers in the spacecraft, at the tracking sites, in the communication link from site to SFOF, and at the SFOF, we will have up to 10 computers handling the data.

The role of computers in semitechnical and nontechnical areas has also been vital. Computers are used for configuration management, budget preparation, inventory control—the list is endless.

Turning to the future, we are looking forward to the attractive features offered by the new generation of computers—the much-needed increases in capacity, speed, memory, reliability, and ease of programming. However, the impact of computers on basic technology is likely to have the most significant future effect on the space program. When I was a student, a large part of the technology was devoted to problem-solving methodology. Today much of this methodology is becoming obsolete. The new methodology, based on fundamental principles and the computer, makes possible the ready treatment of larger systems. Instead of having to analyze the relationship between resistor A and resistor B, the engineer may treat the relationship between circuit A and circuit B, or system A and system B. The viewpoint is less microscopic, and more macroscopic. For example, we are beginning to treat system characteristics, such as cost and reliability, as design parameters. Thus the symbols by which the technology is represented, and hence our way of thinking, are being changed by the modern digital computer.

The impact of computers on the government

JAMES A. WARD

Office of the Director of Defense Research and Engineering
Department of Defense, Washington, D. C.

Imagine a large mass of nonhomogeneous metal. Imagine that it is struck a severe blow by a large hammer. The shock wave of the impact will travel throughout the mass and arrive at different parts at different times. Furthermore, this shock wave will be reflected from boundaries and regions of nonhomogeneity. These reflected shock waves will affect some portions of the mass time and time again. Occasionally, shock waves reflected from several directions will converge on some internal region to give an aftershock as great (or greater than) the original shock. Such shock waves may persist for a considerable time.

The impact of computers on the federal government has been analogous to such a hammer blow. There has not been *an* impact, but many impacts. These have struck various government agencies at different times. In some places the impact has struck again and again.

However, the impact of computers differs in a very fundamental respect. The shock waves in the metal attenuate as they convert into heat, whereas the "shock waves" in government which generate heat seem to become amplified and increase the heat. Let us look at a few of these shock waves and the impact they have had on the federal government.

The first electronic digital computer was the ENIAC. It was placed in operation at the Moore School of Engineering just 20 years ago, in the fall of 1945, and was used for 10 years. It was built by the University of Pennsylvania for the Army at a cost (including peripherals and air conditioning) of $486,804.22. As for the impact, its 30 separate units weighed over 30 tons!! It had 19,000 vacuum tubes, 1,500 relays, hundreds of thousands of resistors, capacitors, and inductors and miles and miles of wire. It used 200 kilowatts of power, including power for the air conditioners.

The ENIAC was a computer which could add, subtract, multiply, divide, and extract square roots. By today's standards, however, it had a very limited capability. Its total memory was 20 words (which was not *really* a memory in our present sense of the word), more like hardware holding registers. (However, a memory of 100 words of core storage was added in 1953.) Programming was somewhat analogous to plugboard wiring, and maintenance problems were severe.

For a time, though, the ENIAC was the best general purpose digital computer in existence. It was invented to solve trajectory problems using the differential equations of motion, and, as such, it was an astounding success. A trained operator with a desk calculator could compute a 60-second trajectory in 20 hours; the ENIAC could do it in 30 seconds, just half the flight time. It became the major instrument for the computation of all ballistic tables for the U.S. Army and Army Air Force. In addition it solved problems in weather prediction, atomic energy, cosmic ray studies, thermal ignition, random number generation, wind tunnel design, etc.

Following the ENIAC, as each new computer was placed in operation it made a considerable impact, both in the government and the scientific world. The speed with which it could solve scientific problems and the number of desk calculations and clerks it could replace were items that impressed everyone.

The UNIVAC at the Bureau of the Census in 1951 was the first computer applied to business type problems and as such made a tremendous impact. Those using it said at the time that it was doing a better and more complete job for half the cost. Speaking of this computer, a writer in the *Scientific Monthly* for February 1953 said: "There is in sight a clerical revolution that could match the Industrial Revolution." Though this sounded far-fetched at the time, it has turned out

to be prophetic.

Here are some data from the "Inventory of Automatic Data Processing Equipment in the Federal Government 1965," published by the Bureau of the Budget. There are an estimated 2,188 digital computers in 35 agencies of the federal government. In FY 1966 automatic data processing utilized 63,000 man-years of personnel and cost $1.1 billion for acquisition and operation.

What are all these computers doing? Two-thirds of them are in the Department of Defense. There they do missile simulation, test data reduction, and war gaming, and are applied to many kinds of scientific problems; they do inventory and management work, too.

For instance, one of the largest applications is in the Air Force Base Supply System now being implemented. When completed there will be some 150 computers in a single network using standardized programs and procedures that will enormously simplify management. It will reduce the training necessary for operation and maintenance and is expected to reduce the overall manpower required by some 800 spaces. Most impressive of all will be the benefits from better efficiency and faster response times.

The DOD spends almost half a billion dollars a year on jet fuel. Every six months the Defense Fuel Supply Center receives and evaluates about 100 bids from industry to supply the 2 billion gallons required. To make sure that "the price is right" requires over 20,000 arithmetic computations. To do the task a computer system uses several different mathematical models. One model solves the complex bid conditions and takes care of the import quotas. Another finds the least cost route from every source to every destination. A third considers the most logical answers and determines the minimum cost award pattern, assuring that the problem is considered as a whole and that the least costly solution is used. Solution of this complex problem formerly required seven weeks by hand; now it is done in one or two days.

In the Washington suburbs is an organization of another type that keeps on magnetic tapes inventory of all the tactical equipment on each Polaris submarine by name. This saves a great deal of time and effort in modifying and supplying them.

A different kind of organization is the Naval Command Systems Support Activity (NAVCOSSACT) set up by the Navy to develop and test the programming for the large naval systems. Among these are operations control centers for the Atlantic, for the Pacific, and for the Naval Forces Europe. NAVCOSSACT is a large operation with an annual budget of $14,000,000. It has nearly 300 programmers and analysts and is trying to hire 50 more.

After the Department of Defense, the world's largest user of computers is probably the National Aeronautics and Space Administration. Computers are a vital part of the NASA system that places humans into orbit, monitors their position in the space environment and brings them safely down. These computers receive radar data, compute orbital parameters, direct communication antennas, and provide handovers from one tracking site to the next, thus enabling us to observe, on TV, long continuous periods of communications with the astronauts. On a typical Gemini flight there are some 34 computers on-line receiving data in real time and another 30 or 40 computers working off-line. About a third of the on-line computers are at Houston, Goddard, and Cape Kennedy. The rest are at other sites around the world.

The Mariner flight to Mars is an astounding achievement—again, vitally dependent upon the use of computers. The navigation and midcourse maneuvers are two of the most important applications of computers.

The Atomic Energy Commission has received one of the greatest computer impacts. With extremely large computation requirements in design, simulation, and data reduction, the AEC has been a pioneer in large-scale computer usage. This application has been told elsewhere.

In 1964 the Treasury Department issued 370,000,000 checks and bonds. Had it been necessary to type them out it would have required 9,000 employees. By use of computers the job was done by 1,300 people; besides, the recipients got their checks much faster. Between 1949 and 1962 the cost per thousand for issuing checks has been cut in half.

The Internal Revenue Service reports a very unusual impact of computers. Its plans to use computers to check personal incomes in one of seven of the national regions were announced to the public; the result was that in 1964, the public reported $2 billion more interest income than in 1963!

The Bureau of Census used computers in the 1950's at a considerable saving. The Bureau obtained better computers and peripherals for work during the 1960's. Though the amount of work has increased (for example the publications doubled) it accomplished its work 6 to 12 months earlier—and with 6,600 fewer man-years, a saving of 25%.

A catalog of the impacts that computers have made on each government agency could go on and on. But there is another kind of impact. The very large activity in computers in 35 government agencies has caused considerable indirect or secondary impact on other government agencies.

For example, the Bureau of the Budget prepares an annual inventory which lists the government commercial computers. It has published an "Automatic Data Processing Glossary," and has prepared a report to the President on the Management of Automatic Data Processing in the Federal Government. It has developed procurement procedures that make sure each installation concerned gets a computer properly suited to its needs and that all qualified suppliers are considered equitably.

The General Services Administration negotiates with computer suppliers and publishes price schedules of equipment. Thus government agencies do not compete with one another in obtaining computers. GSA is expected to play a larger role soon.

The Civil Service Commission has set up separate job descriptions for 14,000 Civil Service personnel now working directly with computers. It had been estimated that the number of government clerical positions would decline with the advent of computers, but this has not been the case. Though over 5,000 jobs have been eliminated, more than 7,000 have been created. The number employed in automation is increasing at a rate of about 1,000 per year.

The National Bureau of Standards provides advice on computers to all government agencies and will continue to play a leading role in computer research.

Since a number of computers are sold abroad (over $200 million worth in 1964) they make an impact on the gold flow problem and the Department of Commerce gets involved.

The Department of State is involved in the use of computers too. Under the Mutual Defense Assistance Control Act of 1951, there is a list of items embargoed for sale (or sold only under restricted conditions) to the Communist Bloc countries. Computers are on the list. At present, representatives of State, Commerce, and Defense are in the process of redefining the U.S. position on computers and other items in this list.

Since 1957 the National Science Foundation has aided educational institutions in their work with computers. This agency helps colleges and universities with computer rental and purchases at an annual rate of $9.5 million. It also finances computer research and conducts summer institutes. In all, it spends over $10 million per year on computers for others.

This introduces computer education, an important topic in itself. For reasons to be brought out later, computer education (or training) is in relatively "short supply" at all levels in the Department of Defense. Last year special courses were given to 30,000 people. In addition, 200 were sent to colleges and universities for graduate work in computers. These courses included programming, hardware design, systems design, numerical analysis, etc.

Each of the military Academies, Army, Navy, Air Force, and Coast Guard, has one or more computers for use of the faculty and cadets. As of this year, each entering cadet is required to take a course in computer programming. Almost all of them are required to use computer programming in solving problems in their other courses. Each Academy also encourages the cadets to use computers on their own, and many do.

A great many senior officers and civilians in the Department of Defense have to work with computers. However, computers were invented after our present senior officers and civilians finished their formal schooling. To assist them, there was set up, last year, the Department of Defense Computer Institute (DODCI). At present it gives two courses: a 1-week (50-hour) course for generals, admirals and supergrade civilians and a 2-week (80-hour) course for Colonels, Naval Captains and civilians of comparable rank. (An additional course is planned to begin next year.) The idea is not to make these senior managers experts, but to give them some of the concepts, some of the language, and some notion of what it takes to make a computer system (especially a military system) work. The courses have been enthusiastically received.

Interest in computers has also extended to the Congress. Many hearings have been conducted, and during the last session the Brooks Bill was passed. Among other things, this Bill arranged for a computer administrator in GSA for the management of computers in the federal government and for a pool of funds for procurement. Just how this arrangement will interact with the present arrangements has not yet been worked out. Perhaps this aspect would make an interesting topic at a future meeting of AFIPS.

Thus we see that, in addition to those who work directly with computers, there are many others—even entire agencies—that feel the impact of computers on the government.

As was stated earlier, the Bureau of the Budget inventory lists some 2,200 computers in the federal government at an annual cost of over $1.1 billion including manpower and operation cost. Every one of them is needed. During the current fiscal year some 260 more are expected to be added, and the total cost including operation is expected to rise slightly. Besides these, contractors are estimated to utilize about another 1,800 computers on government contracts.

However, there are other entirely different and additional types of computers that are perhaps even more important to the Department of Defense—the military computers utilized as integral parts of weapon

systems. These come in various sizes, shapes, and capabilities and some of them perhaps can only be classified as computers in the broadest terms.

It is difficult to determine the total number of U.S. military digital computers. An informal survey of our manufacturers show that over 12,000 have been delivered with 1,800 of these delivered this calendar year. It may be of interest to note that three manufacturers have delivered over 1,000 military digital computers each. They are, in alphabetical order, Hughes Aircraft, North American Aviation, and Raytheon. One of them had a special news release a few months ago on the occasion of the delivery of its 3,000th computer.

A vital part of our national defense is our missile capability, and therefore this defense depends on the computers that these missiles utilize. Such computers are different in many respects. For example, they are programmed by the manufacturer and management is not worried about how much they are utilized!

For instance, most types of missiles have inertial platforms that provide data to special purpose digital computers which perform the missile navigation computation. Such a computer usually includes a digital differential analyzer (DDA) with a wired-in program. However, it has a memory, can add and subtract, can make decisions, accept data and give output.

The first missiles were checked out by hand, but with the vast number of electrical levels that must be set and verified, it soon became evident that a computer could do this much more quickly and with greater accuracy.

In Civil War days the cannoneer sighted down the cannon barrel, estimated elevation, and fired. To aim missiles at targets hundreds of miles away is an entirely different matter; this is a task for a computer which determines the proper parameters and enters them into the guidance computer contained in the missile.

This "aiming" computer on a POLARIS submarine is a very respectable computer. The POLARIS submarine also has computers for navigation. The Ships Inertial Navigation System (SINS) contains a DDA to integrate the output of the accelerometers. The SINS is connected to a general-purpose navigation computer. All these computers are replicated for reliability. Computers have made quite an impact on the "submarine business." Instead of being restricted to attacking targets that they can "see," the computers enable submarines to act as a strategic weapons platform that can be used to attack targets over 1,000 miles away.

Digital computers in military aircraft aid the pilot in navigation, receive radar data, calculate intercept courses, assist in the aiming of missiles, in dropping bombs, etc. In fact, inertial navigators with digital computers are in test phase on several large commercial jet planes.

Radars used for missile and satellite tracking have special-purpose digital computers. Some also have general-purpose computers for converting orbital parameters into look angles, and vice versa.

Another interesting application of computers is the Department of Defense communication network AUTODIN (Automatic Digital Information), a nationwide network for the transmission of digital messages. There are 6 "switches" with up to 250 subscribers per switch; each switch contains a computer with considerable storage capability. From the header of an incoming message the computer determines where the message should go and its priority. As soon as its priority warrants, the message is forwarded to the next switch and/or to its destination. It is an all-automatic network.

Then there are many kinds and sizes of military general purpose computers. Most of them were designed by R&D contracts and especially suited for a particular task. The ANFSQ-7 used in the SAGE centers is an early example. It is very reliable, and has the capability to handle an unusually large number of inputs and outputs.

Most general purpose military digital computers are built to withstand a physical environment of temperature extremes, vibration, and shock that could not be tolerated by the average commercial computer which operates very well in a stable air conditioned room. They are also required to pass more stringent reliability tests.

When John Paul Jones stood on the quarter deck of the *Bon Homme Richard* he could see and evaluate every threat to his own ship and to any in his task force. Today a task force covers hundreds of square miles with picket planes out beyond those boundaries. Their combined radars cover thousands of square miles. By means of NTDS (Naval Tactical Data System) all the targets of all these radars are processed by computers, transmitted, correlated and displayed on a scope for the task force commander. With the aid of computers (and radar and communications) the commander can "see" his situation as well as could John Paul Jones. To date the system is on 10 ships and has cost somewhat more than $300 million, including RDT&E. There are two to three computers per system and it is truly a multicomputer system. Other multicomputer systems that I know of have one computer operating and another on standby; the NTDS system has the job split up among the computers.

The Army has the well-known truck-mounted BASIPAC and MOBIDIC. In addition, it is in process

of fielding a substantial number of FADAC computers. These are computers designed to operate in the field to compute firing data for an artillery battery. Napoleon once said that God fights on the side with the "most cannon." The FADAC will add so greatly to the surprise ("first round accuracy") and effectiveness of the artillery, that it will have the effect of being the "most cannon."

No account of the impact of computers on the government or national computer community would be complete without mentioning SAGE, i.e., the Air Force Semi-Automatic Ground Environment system. Undoubtedly, a nontrivial portion of this audience has been directly affected by it and many others have been indirectly affected.

SAGE is an air defense system whereby radar data is digitized, transmitted to computerized centers, processed, displayed and, in some cases, transmitted to other centers. It involves a great many technologies, but let us look only at the computer aspects.

Some people measure impact in dollars: the computers for SAGE cost $1 billion, and the programming an additional $110 million.

There are 150 special-purpose computers to convert the radar data to digital form for transmission to the computerized centers. In each center there are two (for reliability) large special general-purpose computers.

The first installation of the system was placed in operation at McGuire AFB in July, 1958. SAGE built up to 25 centers, and at present 19 remain in operation.

The programming for SAGE was a very large task. The cost of programming Model 1 was $18.5 million. As more capabilities have been added the programs have been repeatedly revised until today Model 9 is in operation. The programming contract for the current fiscal year for modifications and improvements of Model 9 is budgeted at $8.4 million. The number of instructions in the present system is over 400,000 with about 100,000 at each installation. In addition, the active utility programs contain about 750,000 instructions.

The number of programmers required was originally estimated to be about 600, which at that time (1951) was about the total in the U.S. Hence, SDC (the programming contractor) has had to train (partly as a result of personnel turnover) some 2,000 programmers.

This is not the time, place, nor the speaker to comment on the operational achievement of the entire SAGE system. However, the computer systems therein have made quite an impact. Some important points are:

1. The ability to process radar data in real time.
2. A man-machine interaction through data display and manual input, bringing the computer capability to engineers, managers, and decision makers.
3. The extensive use of computer-to-computer communications over a distance of miles.
4. The feasibility of over 100 people time-sharing a computer on a real-time basis by means of a device like a satellite computer.
5. The ability to switch automatically over to the standby system.
6. The development of sophisticated software utility systems.
7. The COMPOOL concept of many subprograms sharing and processing data for each other.

Another class of general purpose computers is used in Command and Control Centers. There are many others such as general-purpose computers designed to operate in airplanes, in vans, in helihuts. There are also a number of one-of-a-kind computers and those designed to be used as training devices.

There are "odd ball" computers in the government other than military. NASA has put especially designed computers in satellites. Others are used in launching and tracking stations.

Process control computers are also used in the government. Most of those sold now are general-purpose computers with special I/O capabilities and designed for great reliability and rugged environment such as vibration. They not only control machine tools but power distribution networks and power generating facilities.

The TVA has 200 power generating units with different kinds of fuel, different efficiencies, and differing costs of operation and stream flows. It has 1.5 million customers spread over 80,000 square miles. How to generate and distribute the power in the most efficient manner when loads are changing by the second is a task that computers do for TVA.

Let us now consider computer research. A little over a year ago, in an article in *Datamation,* a well-known computer user had this to say:

> Five years ago, the Government should have initiated research into the problems of constructing digital computers 100 to 1,000 times as fast and 1,000 to 1,000,000 times as large as far as rapid access storage is concerned. It is absolutely absurd in a world in which science and technology are obviously of major importance to allow the development of this fundamental scientific tool, the digital computer, to be left to the whims, chances and tax write-off of commercial computer companies.

Of course, our government continues its very large

support of computer research and development—but, understandably, in ways that are expected to result primarily in the solution of its own pressing problems, especially in national defense. A rough unofficial estimate of Department of Defense annual expenditure on computer research, development, test, and evaluation (including hardware, software, peripherals, and weapons systems) is $280 million. The present day development of computers has resulted, to a considerable extent, from the continued cooperation between government and industry beginning with the Army's original contract to build the ENIAC.

There could be several reasons why the major government support of research on computers is perhaps not as generally well known as it might be. Most of the work is done (and properly so) by the computer manufacturers and even though funded by the government, is reported upon and administered by those individual industrial concerns. Only a relatively small portion is done in the government laboratories, though most of it is monitored by and funded through them. Also, computer R&D cost is often lumped in with the weapon system in which the computer is used, and separating it is like extracting a pound of flesh without loss of blood. A third reason is that the payment for a contractor's R&D may be indirect. For example, government procurement contracts allow "overhead" and, under certain conditions, this overhead can be used to support a company's "own" IR&D (Internal Research and Development). Just which company tasks are government-funded is settled by negotiations and does not necessarily appear explicitly in any contract. The amount so spent is over $1 million per year by each of a number of companies.

The computers in each of the weapons systems have gone through RDT&E. Such costs for NTDS (Navy Tactical Data System) have been over $95 million. The Marines are developing their second tactical data system. The Army is developing a computer for the new NIKE system. The Army also expects to spend $10 million this year on their ADSAF program (Automatic Data System for the Army in the Field). For this they are planning to use van-mounted computers for the troops in operations. They are planning a supply system, a tactical fire planning system, and a tactical operation system for the commanders of field armies. Napoleon could stand on a hill and see his and the opposing forces and their movements. A modern army covers thousands of square miles. The Tactical Operation System is a realtime, on-line, computerized, communication and display system that will give the commander an up-to-the-minute view of the tactical situation.

Besides the many large systems presently being developed throughout DOD there are many hardware development programs: associative memories, cryogenic computers, mass memories, complete new computers, physically small memories, sub-nanosecond circuitry, and various peripheral devices including displays.

It would seem that in a tactical environment (airplane, ship, truck) a mass memory with no moving parts would have an advantage over a drum or disk. Such a device should be cheaper than core (even if the core is fabricated in Hong Kong), small, and rugged. So far 13 techniques have been investigated and several are being funded. Some of these methods are (1) a plating and etching process that gives the effect of wire-strung cores, (2) thin-film plated wire, (3) wire screen woven on a special loom, (4) a "delay line" that remains frozen but can be rapidly shifted as desired, (5) moving domain walls, (6) the residual magnetism left in a metallic rod where an electrical pulse overtakes an acoustical pulse, and (7) artificial crystals.

Also, entirely new types of computers are being developed. For instance, the Air Force is building one without conventional commands. The programmer specifies in a separate memory what his commands are and then writes a conventional program in the conventional way. Such a computer will be able to operate like—not simulate but operate like—any of several different existing computers and accept their programs. It is also a tool that can be used to determine what hardware organization and command repertoire is optimal for a given job or with a given compiler language.

The first computers were vacuum tube computers. In the transition to transistors the immediate achievement was the reduction, by a factor of 10 or more, in the physical size, power requirement and air-conditioning requirement. However, the real impact has been in reliability increase and cost reduction.

In the tube computer days the maintenance man arrived before daylight to check out the computer before the working day began, and he was in attendance as long as the computer was operating. Early reports on the first UNIVAC at the Bureau of the Census show that they wished to run the computer 24 hours a day, 7 days a week. However, during some weeks the computer was down 60% of the time. Let us be generous and say that the average vacuum tube computer had a mean-time-before-failure (MTBF) of 10 to 15 hours.

In the early days of transistors a contract request for delivery of a new military computer was almost turned down by one manufacturer because of the reliability requirement: it was required to have an MTBF of 50 hours. That would be almost like asking for a 50-year MTBF today. Now many military computers have well

over 1,000 hours MTBF and there is one case of a military computer that ran over 10,000 hours without a failure. The reliability know-how developed for military computers has been applied to the design of commercial computers so that they also can be quite reliable now.

Along with this increase in reliability has come a reduction in cost. For instance, one particular computation problem that cost $14.40 in 1948 is expected to cost 2¢ in 1966. Reduction in cost and increase in reliability have been probably the major reasons for the rapid expansion of computer use in the government and in industry. This pattern of reduced cost and increased reliability is expected to continue in the third generation of computers—that using integrated circuits.

For some time now the Department of Defense has been sponsoring research in integrated circuits. As usual, the first application of this technology has been in military computers. The result has been a remarkable reduction in size, weight, and power. A dozen or more companies are now in the process of building such computers, and several are being shown at conventions. One such operating computer is the guidance computer in the Minuteman II: it is 6.3 × 10.3 × 21.3 inches, including memory and power supply. However the most remarkable aspect of integrated circuit computers is the increase in reliability. Its computed MTBF is measured in years.

Compare this performance, if you will, with that now considered acceptable commercially. I know of one computer, on two-shift operation, that is "down" about four hours a month; the manager feels this is normal.

In advancing from tube to transistorized computers, the MTBF went up by a factor of 10. At least another factor of 10 increase is expected going from discrete to integrated circuits. In a short time, hopefully, this technology will be cheap enough to be used in commercial computers. Some companies have announced computers using such circuits.

Already coming over the horizon is what looks like the fourth generation of computers: MOS (Metal Oxide Semiconductors). Their use promises to yield yet another major reduction in size and increase in reliability. Already, manufacturers are seriously considering using this technique to put the entire arithmetic unit of a computer on a single 3 × 4 inch logic card, or a complete DDA (except for memory) for a missile guidance computer on a single chip the size of a quarter. With some of the new miniaturized memories it seems likely that a few years hence a computer with the capability of the most common ones now in use could be carried in a briefcase—or even attached to the underside of the input typewriter!

The speed with which the technology is advancing is aptly illustrated by a recent note in the *New Yorker* magazine. The note consists of two sentences. One sentence quoted from the text of a new book: "While you are reading this sentence, an electronic computer is performing 3 million mathematical operations!" The author probably had in mind a well-known commercial computer that is advertised to perform 3 million operations per second. But before the book could be published, the author and publisher heard of newer developments so that the dust jacket has the following sentence: "While you are reading this sentence an electronic computer is performing 4 billion mathematical operations!"

Considerable research is also being done in software. This includes language translation, compiler development, information retrieval, pattern recognition, operational systems, and time-sharing systems, to name but a few.

However, it is now quite clear that a great deal of further research is needed to make computers more generally useful. The original computers were designed to solve differential equations and other mathematical problems. The design was extended to make business type applications easier. They were designed for batch processing and to be programmed in machine or symbolic language.

Now computers of this design are being used for nonnumerical operations such as command and control, information storage and retrieval, language translation, etc., and are programmed by compiler. What logical hardware organization is needed for this kind of work?

More efficient compilers are needed to make existing machines operate more efficiently. A language with 5-character labels will not work very efficiently on a computer with 4-character words!

Computers are designed and then compilers are written for them. In my view this process should be completely reversed. If a computer is intended to be used mostly through a higher-order language and compiler, it should be designed to do this easily. In fact the language-compiler designer, the executive system designer, and computer hardware designer should cooperate on an integrated hardware-software package. The designers of the hardware and software for the new computer for Nike X have so cooperated. Though the forthcoming computer is faster than the preceding ones, the throughput increase is much greater than the hardware speed increase.

Also, better peripheral equipment is needed, especially for tactical military systems. Better kinds of equipment and programming systems for man-machine

interface is needed. More people need to be educated on how computers work and what their limitations and capabilities are.

In summary, then, we have seen that computers have made many kinds of impacts on the government. An annual cost of $1.1 billion and 63,000 man-years of effort is no mean impact. By taking over repetitive tasks the computers get the work out faster, more accurately, and with fewer people—as illustrated earlier by the Treasury Department check-writing example. Saving substantial amounts of money and time always produces an impact. Computers enable tasks to be undertaken that would otherwise be impossible such as the NASA manned space flights. The utilization of computers has caused indirect impact on other government agencies, such as the Civil Service Commission, the Bureau of the Budget, the National Science Foundation, etc.

The Department of Defense has received exceedingly strong impacts. Not only does it have two-thirds of the commercial-type of computers in the Federal Government, it also has many and diverse types of military computers; many weapons systems have computers as integral parts thereof; computers are also essential parts of many tactical and strategic systems.

The Department of Defense (and to a lesser extent other government agencies) is spending a very considerable effort on computer research across the board (hardware, software, peripherals, and systems). As they become available, the fruits of this research will cause impacts that will continue to reverberate throughout the government and in industrial and commercial enterprises as well.

It is told that someone asked the late Norbert Wiener if the government could now do away with its computers. He is said to have replied that the government, having had computers, could not possibly go back to doing without them: that it was like Adam and Eve having eaten the forbidden fruit—they could never be the same as before.

I would like to add that eating the forbidden fruit gave Adam and Eve the ability to tell right from wrong and the incentive to acquire wisdom. I trust that all of us will use our computer-aided wisdom to make the sensible decisions for better utilization of the computers we have and to invent better ones.

Communications, computers and people*

PAUL BARAN

The RAND Corporation
Santa Monica, California

INTRODUCTION

Communications and computers are today becoming what the economists call "complementary goods"—one without the other is of much lesser value—like pen and ink, pretzels and beer, and gin and dry vermouth.

Let us first briefly consider the impact of the computer technology upon the communications business and, conversely, how good, widespread, low-cost digital communications will allow a dramatic increase in the creation of new types of computer systems. Then we shall get down to the meat of the talk—a few of the unappreciated social consequences possible and, lastly, we shall proffer remedies in advance of the time society realizes there is a problem. If the order of things appears backward, with remedies being offered in advance of the patient's complaining of an ailment, it is due to our belief that the lead time for the cure of social ills is often longer than the gestation period of the disease. Only we who appreciate what is happening to computer development may be in the best position to see the thunder clouds.

THE IMPACT OF COMPUTER TECHNOLOGY ON COMMUNICATIONS

Communications equipment is sometimes categorized into switching equipment, transmission equipment, or terminals. As we expect to be talking about digital uses which by definition are digital terminals, we shall confine our observation to telephone switching and transmission.

At present, the telephone plant, our prime data carrier, is almost exclusively based upon electromechanical switching—that most primitive form of computer logic —and one that we in the computer business haven't seen around for years. Transmission is by means of frequency division multiplexing—or about as analog (or undigital) an operation as we computer types can envision. The only kind words a computer man can have for this system is that it works; it works well and has been working well for many years—for the purpose for which it was designed.

While perhaps slow by pace, electronic switching has arrived on the scene for the telephone company. At least two separate systems in this country have now passed field trials and are being installed commercially. This new equipment may be representative of the future telephone local central offices. At present, these electronic switches are not believed to be more economical than their earlier electromechanical switch counterparts. But their prime advantage lies in the *new additional services* that they offer because of the general computer nature of the control mechanism of the switching center. For example, it will be possible to dial only two digits to reach the few numbers that you call often. It will be possible to relay a call to another telephone if you are temporarily away. Automatic diagnostic routines will permit repair and mainte-

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion of the policy of any of its governmental or private research sponsors.

nance by inexperienced personnel. Further, electronic switching is a new technology whose price is expected to decline rapidly in the future.

So much for switching. We also see computer technology creeping into the picture in transmission. The Bell System T-1 multiplexing system samples 24 analog voice channels about 8,000 times per second producing, together with synchronizing information, a data stream of 1.54 megabits per second which can be transmitted over ordinary copper pairs using pulse regenerative amplifiers. This pulse code modulation technique is being developed simultaneously in many countries and is in use but presently restricted for links on the order of magnitude of about 20 miles.

As pulse code modulation is the most economical of the multiplexing systems, it appears destined as the transmission direction of the future. Even though digital technology is entering the telephone plant slowly and in a piecemeal fashion, it is arriving and will make its impact felt. Specifically, most of the growth of the telephone plant may be expected to occur within these digital techniques areas. The implications to us are severalfold.

First, as we expect to see a rapid drop in the cost of digital circuits, we may expect continued drops in the price of digital communications in the future. We would also expect to see even more marked savings to the digital communicator as systems evolve which are more amenable to the all-digital processing of information from user to user. Today emphasis must be given to complete compatability with the large existing analog system in being where periodic reconversion to analog signals is required. Thus, one day we envision the bulk of the telephone system being built entirely of digital processing assemblies in lieu of the all-analog systems as of today. When this day comes, we computer types would view the telephone system as merely another particular computer application and not necessarily a specialty field unto itself. If the old-time telephone engineer fears that the computer types are taking over, he is probably right. So much for what computer technology might do for or to communications, depending on where you sit.

THE IMPACT OF COMMUNICATIONS UPON COMPUTERS

Using telephone lines modified to handle digital data, we are able to build an increasing number of geographically distributed time-shared computer systems. Many individual users are connected to a common computer data base. Examples of such systems include airline reservation systems for civilians and fancy display "command and control" systems for the military.

Simple record keeping, a mark of a highly developed economy, has been a prime area of development of these large computer file/communications systems where much of the routine clerical work is transferred to the computer with human interrogation of the system. As time moves on, the number of people who will be able to interrogate the system and the geographical distance between them and the machine will increase.

SOME INDIVIDUALLY USEFUL SYSTEMS

Today we see time-shared file systems used for insurance records, for checking automobile tags, to locate outstanding criminal warrants, and for credit check investigations (using drivers' license numbers) in cashing checks. The systems built to date pose no overt social problem. The information handled is not highly sensitive and access to it is generally limited.

THE TRAIL OF ARTIFACTS IN A CIVILIZED LIFE

As we pass through life, we leave a trail of records widely dispersed and generally inaccessible except with a great deal of effort and diligence.

We start with a duly recorded birth certificate. We leave behind hospital records and our pediatrician adds to our medical records. We are deductions on our parents' income tax. School is a place where we busily generate record upon record of our scholastic grades, our attendance, our IQ test records, our personality profile records, volumes galore. With automated teaching coming to the fore, we can expect better record keeping. The volume of data we will record per child may be expected to increase even more markedly ("in the best interests of the student"). Between terms we get our social security card and a job, and we start leaving behind us a long history of employment records. We reach age 18 and are entered upon the records of the Selective Service. We get a driver's license and, if we are lucky, we will be able to avoid having arrest and jail records. Most of us will apply for a marriage license, some of us will collect divorce decrees which will end in voluminous court records. We move from job to job in a mobile economy creating moving-company inventory records of our goods. Even as we move from place to place we leave behind short records of our airplane reservations and for some reason every hotel makes a ritual of acquiring and preserving the names and addresses of its guests for posterity.

This list is only a partial one. Play the game yourself

and think of all the records you leave as you go through life.

WHY SO MANY RECORDS?

One does not create records merely for the sake of creating records. But rather there is the implicit assumption that the records will be of some use some day. In order to be of use, there must be some means of interrogating the files to resurrect the information sought. Thus, we envision large families of systems, each individually useful. For example, an Internal Revenue Department investigator might wish to have immediate access to the tax returns of the associates of a man who is being audited to check for consistency of financial relationships.

A company may wish to have rapid access to its personnel files to know whether to give a good reference to a former employee.

A doctor may wish to trace the entire medical history of a patient to provide better input into a diagnostic computer.

The Veterans Administration may wish to examine a man's complete military record and *possible other* previous medical records to see whether the ailment claimed as being service-connected really is.

A lawyer for the defense of a man will wish to search for jail records, arrest records, and possibly credit records of *all* witnesses for the plaintiff.

Professional licensing boards may wish to delve into *any* records that may indicate the applicant lacks an unblemished character.

The military in filling extremely sensitive positions may even wish a record of all books borrowed by the prospective applicant to insure that his interests are wholesome and he possesses the proper political bias desired.

ACCESS TO THIS INFORMATION

Today one does not gather such information about the prospective examinee easily. If one went through the direct channels and asked most sources for their records about a person, he would most likely be told to go jump in a lake, if for no other reason than the information is not available—cheaply. Even if the information were a publicly available record, the investigator must be expected to spend a great deal of time and effort delving to discover pertinent data. Today, as through a practical matter, if one wishes to obtain much of this information about a person, he hires a private detective who charges a great deal of money and expends a great amount of time obtaining a little information available from a portion of these potential records. The price for a fishing expedition for information is high and most of the fish are inaccessible.

THE IMPENDING PROBLEM

So much for the pleasant past. Consider the following argument:

1. A multiplicity of large remote-access computer systems, if interconnected, can pose the danger of loss of the individual's right to privacy—as we know it today.
2. The composite information data base may be so large and so easily accessible that it would permit unscrupulous individuals to use this information for unlawful means.
3. Modern organized crime should be expected to have the financial resources and access to the skills necessary to acquire and misuse the information in some of the systems now being considered.
4. We are concerned not only with the creation of simple "automated blackmail machines" using this information, but with the added implication of the new "inferential relational retrieval" techniques now being developed. Such techniques, when fully refined, could draw chains of relationships from any person, organization, event, etc., to any other person, organization, or event.
5. Humans, by their day-to-day necessity of making decisions using totally inadequate evidence, are innately prone to jump to conclusions when presented with very thin chains of inferred relationships. For example, merely plastering a man's name on billboards will markedly change the outcome of an election, if the other candidate's name is not equally displayed.
6. The use of private detectives to unearth derogatory information on political candidates *and* their associates has become an increasingly prevalent feature of elections. This practice is expected to increase in the future.
7. The cost-per-unit-dirt mined by unautomated human garbage collectors can be cut by orders of magnitude once they obtain access to a set of wide-access information systems which we now see being developed. It is the sophisticated form of chain-relation blackmail that may be of most social concern. We generally pass through three stages of information storage development. First, we start by keeping manual records employing clerks. Next, we get rid of some of the clerks when we put all the records into a single central computer file with the readout controlled from a single point. The next step is the creation of remote interroga-

tion devices to interact with the file from a large number of points. The payoff for instant access is often high as it eliminates all delay to the file user.

8. This development of geographically widespread access systems requires the use of communications lines to connect the users into the computer. There is a widespread belief that somehow the communications network used will possess a God-given sanctuary to privacy, but "it ain't necessarily so . . ."

DIRECTIONS TOWARD A SOLUTION

1. Assume that not everyone is as honest and as trustworthy as ourselves—but is just as diabolically clever.

2. Appreciate that we will be increasingly dealing with complex and, hence, difficult-to-understand-all-the-details types of systems in the future.

3. Probably the only people who best understand the operation of each system will be computer design engineers who build the system in the first place.

4. Often the only time that the fundamental safeguards that we seek can be applied is at the time of the initial system design. "Software patch-ups" at a later date may generally be relatively ineffectual compared to good initial design—good design being defined as including an awareness of the existence and importance of the problem.

5. Do not expect help from the legal profession in lieu of good design. Even ignoring the social lag of the legislative/judicial procedure, the detailed subject matter verges on or beyond the limits of their comprehension.*

6. Laws and laws alone have been almost totally ineffectual in the growth of widespread electronic eavesdropping and wiretapping. At most, all the courts have accomplished is to prevent the police from using the same techniques available to the private detective or the criminal—or even casual readers of an electronics technician magazine.

7. While I have little faith that laws in themselves will *solve* the problem, laws could be helpful in two ways: (a) Laws outlawing certain practices will be of minor help in increasing the price of the act and making its commission less flagrant; and (b) laws can be written so that potentially weak systems cannot be built unless adequate safeguards are incorporated throughout for the protection of the information stored.

8. This last direction is to me viscerally unsatisfying as it carries with it a built-in loss of freedom. The thought of the creation of another governmental agency peering over one's shoulder contains the seeds of the possibility of bureaucratic decay and arbitrary conclusions based upon an incomplete understanding of complex problems.

9. Historically, government regulatory agencies start as highly effective bodies but lose momentum as the original personnel leave and their replacements come from the industry being regulated. (Where else are you going to get competent people who know the business?) The extreme competence that we need in a regulatory agency of this type is too rare a commodity.

10. If we are to avoid external regulation, then it behooves us computer-communication system designers to start working, or at least thinking, about the problem. We should take the initiative and the responsibility of building-in the needed safeguards *ourselves* before Big Brother is forced to do it himself and we are not too happy with the way he might want to do it.

11. Safeguards, whether they be screens around moving machinery or circuit breakers, cost money. Every design engineer is reluctant to add anything that costs money and buys little *visible* protection. But the writer believes that it is time to start regarding such added costs as *necessary* costs—a price to society for the privilege of building a potentially dangerous system.

12. This is not a new concept. We have, for example, been practicing this in the design of sewage systems and in electrical distribution systems for some time. But, historically, it has generally taken an epidemic to build a local sewage disposal system. It took a series of disastrous fires to get our electrical codes.

13. The national geographical extent of the new data systems, their impact, and their investment are so large that the price of a "retrofit" after the calamities occur may be a higher price than we need have paid if we had used some preplanning.

PROPOSED SPECIFIC SAFEGUARDS

To be more specific, what safeguards do I envision? Of course, we don't know all the answers yet. But, clearly, there are several steps that we should be considering, including:

1. Provision for minimal cryptographic type protection to all communications lines that carry potentially

*Eight months after this talk was presented, a special Subcommittee of the Committee on Government Operations headed by Representative Cornelius E. Gallagher looked into this problem. (These hearings, entitled "The Computer and the Invasion of Privacy", by a Subcommittee of the Commitee on Government Operations House of Representatives, Eighty-ninth Congress, Second Session, July 26, 27 and 28, 1966 are available from the U.S. Government Printing Office for $0.75.) Because of these hearings and the resulting interest and action, many of these words are now obsolete.

embarrassing data—not super-duper unbreakable cryptography—just some minimal, reversible, logical operations upon the data stream to make the eavesdropper's job so difficult that it isn't worth his time. The future holds the promise of such low-cost computer logic, so this may not be as expensive as it sounds.

2. Never store file data in the complete "clear." Perform some simple (but key controllable) operation on the data so that a simple access to storage will not dump stored data out into the clear.

3. Make *random external* auditing of file operating programs a standard practice to insure that no programmer has intentionally or inadvertently slipped in a "secret door" to permit a remote point access information to which he is not entitled by sending in a "password."

4. When the day comes when individual file systems are interconnected, let us have studied the problem sufficiently so that we can create sensible, precise ground rules on cross-system interrogation access.

5. Provide mechanisms to detect abnormal informational requests. That is, if a particular file is receiving an excessive number of inquiries or there is an unusual number of cross-file inquiries coming from one source, flag the request to a human operator.

6. Build in provisions to record the source of requests for information interrogations.

7. Audit information requests and inform authorities of suspected misuse of the system.

This list is open-ended, and it is hoped that more suggestions will be forthcoming. But, to serve as an example of the need for and type of safeguards we are talking about, to illustrate how such thinking can ameliorate the problem of loss of privacy, consider what we might do in the case of our present telephone system.

ONE EXAMPLE OF INCLUSION OF PROTECTIVE MEASURES: THE TELEPHONE SYSTEM

Today we are deluged with bogus telephone advertising, crank calls, bomb threats, false fire and police alarms. Obscene telephone calls, particularly to single women, have become so prevalent that it has been publicly suggested that female names be listed only as initials.

In Washington, the number of these calls has become so great that after much Congressional and press discussion, the penalty for making obscene calls was raised from $10 to $500. Of course, it is a rare event when a person making an obscene telephone call is caught, so the deterrent effect is almost nil. But an increased penalty hidden in a law book is the standard legal response to a basically technological/social problem. This writer would prefer to see *technology* which *created* this problem be required to provide more effective safeguards.

For example, each telephone (or at least those plagued with these calls) should have a button which when pressed bridges the call to a bank of recorders at the police station and a teletypewriter message with the name, address, and telephone number of the calling party transmitted to the nearest police car. It wouldn't take long to clean up the undesired callers.

If you were to make this suggestion today you would be told that this is not practical because it would be prohibitively expensive since this requirement did not exist when the original electromechanical telephone system was set up. This is true, but let us look at the emerging use of the all-electronic switching centers we have been talking about. It will be relatively easy now to add such an immediate track-back feature. Will we do it? I don't know. It would cost money and there are many reasons telephone companies would wish to avoid getting involved—but here is a perfect example of the social implication of the instrument which can violate our right to be left alone. The telephone can be designed (at a somewhat higher cost) to provide safeguards forming added protection to prevent it from being socially misused.

Clearly here is an example of the trade-off between dollars and the type of society we want. It will fall to the computer system engineers to make such decisions more and more often in the future.

What a wonderful opportunity awaits the computer engineer to exercise a new form of social responsibility. The advent of the new computer-communications technology need not be feared with trepidation as we approach 1984. Rather, we have in our power a force which, if properly tamed, can aid, not hinder, raising our personal right of privacy.

If we fail to exercise this unsought power that we computer engineers alone hold, the word "people" may become less a description of individual human beings living in an open society and more a mere collective noun.

It may seem a paradox, but an open society dictates a right-to-privacy among its members, and we will have thrust upon us much of the responsibility of preserving this right.

The impact of computers on urban transportation

KENNETH J. SCHLAGER

Southeastern Wisconsin Regional Planning Commission
Milwaukee Wisconsin

Dr. Ramo has said that one of the purposes of this series of talks was to try to "close the gap" between the information technologists and the applications of their technology in the real world. If this is one of our primary objectives, I don't believe there is a field in which the gap between the technologist and the application is so great as in the area of urban planning, of which transportation is a part. There's definitely a gap between the person who knows the problem and the person who knows how to solve the problem. The reasons for this gap are historical and complicated. There's no need to go into them at this time, but the gap, I think you'll discover from this talk, very definitely exists and needs to be closed.

When one considers the concept of the impact of computers on urban transportation, he is almost forced to remark, "Impact, what impact? Change, what change?" Looking around today in urban areas, you see crowded freeways, decrepit, decaying transit systems. It somehow all seems the same except that it's getting worse. The automobile is certainly not a product of the computer. Our old rail and bus transit systems are certainly not a product of the computer. So why should we blame it on the computer? Just what is the impact? What has it been? What is it going to be?

At the other extreme, the Sunday supplement articles recite the wonderful hopes of the future: automatic highways, completely controlled freight terminals, exotic transit systems. What is the real truth? What kind of impact has there been and what type of impact seems to be coming?

The true situation is somewhat between these two extreme views. The first indications of an impact are now apparent. More precisely, a dual impact of computers on urban transportation has become evident in the last few years. The first impact relates to the use of computers in the planning and design of transportation systems. And inevitably this design application results in an involvement with the whole problem of urban design because it is impossible to separate the transportation or circulation system in a city or a metropolitan area from the spatial pattern of the city itself. The second impact is more direct and involves the use of computers as part of the transportation system itself. Today in transit systems, and in the future even in individual vehicle-type transportation systems, this second impact will be felt.

Now I believe very strongly that the first of these applications is by far the most important: the influence of computers on the whole urban environment. The technologist might disagree and consider the other application far more interesting. Thoughtful consideration, however, may cause him to alter his views. Next to the issues of war and peace, themselves, the problem of whether we can create a livable urban environment is certainly very high on the agenda.

There is also reason to be very pessimistic. Looking at our traffic-choked, polluted cities, for all of our advanced technology, for all of our wealth, we live in kind of a squalor. Frank Lloyd Wright once said that America seems to have an unrestrained, impassioned lust for ugliness. I don't like to be unkind to our host

city, but if you walk around Las Vegas at night—even a person who has very little aesthetic sense can't help but question what our wealth and technology have done to our urban environment.

The reasons behind this decayed environment will make it obvious that the computer is not just a vehicle for the automation of existing operations. Its role rather is so fundamental because our urban environment has gotten out of hand.

The causes for this unfortunate situation are two: First of all, the urban environment, the metropolis, has become so complex that it's impossible for any person, architect, planner, economist, or anyone else to really grasp and to create a design to fill the objectives that everybody seems to desire. Secondly, even if he were able to design such a city as the grand master planners of old envisioned, even if he were able to comprehend this complexity, he could not do anything about it because urban decision making and control have also deteriorated to the point where no individual or group can really influence the shape of the changing environment. It isn't just a question of political power. Even if this power was given to those people urging metropolitan government, it really couldn't be used intelligently because of our sparse knowledge of the urban growth process. The computer has arrived none too early. Let us examine its current role in urban design.

Transportation and the urban land pattern are intimately related. At the subdivision level or in a great metropolis, common sense logic supports the idea that the location of activities and the circulation between these activities is interrelated so that the problem of urban transportation planning is really a problem of total urban design. Attempts to solve this design problem have gathered together an interesting group of people in metropolitan transportation studies. City planners, traffic engineers, highway engineers, systems and computer specalists have formed a multidisciplinary team. Some progress has been made. To understand the reasons for this progress, a discussion of the methodology of these studies is appropriate.

The first task in the urban transportation planning sequence is one of data collection, processing, and analysis. A large quantity of information is a prerequisite for intelligent planning. The land use pattern must be determined in great detail. The detailed land use pattern has never been measured in most urban areas. So it is usually necessary to start a new data acquisition program. Data on the travel habits of people are also needed. This data is obtainable only by actually interviewing individual people to find out the origins and destinations of their trips in order to establish a pattern for projecting travel patterns into the future. Other data on resources such as soil and water must also be collected because of their limiting effect on the final plan.

An idea of the scope of this data collection may be inferred from experience in Southeastern Wisconsin, an area with about 1.6 million people, which is actually one of the smaller metropolitan areas. The basis planning information in this area involved 200 to 300 million characters of data. The area served by the Tri-State Planning Committee in New York City requires a data bank of 2 to 4 billion characters of data.

That computers have made it possible to even consider handling this vast quantity of data at all is a truism. But serious problems in the form of data processing "bottlenecks" have appeared. Typically these studies start with high hopes. A large scale data collection is the first step. Some areas never have been able to successfully digest the data collected and the planning process slows to a halt. Although the computers have raised bright hopes and fostered high ambitions, the weakness of current information retrieval and analysis software has tended to dim these hopes and discourage excessive ambition.

Data handling in these urban transportation studies has the disadvantages of both business data processing and scientific computations. Extensive computations are accompanied by a large quantity of input. The lack of sophisticated information retrieval and analysis techniques has led to the use of crude "brute force" processing practices.

The second task in the planning sequence is that of forecasting. In order to make a plan for an urban area it is necessary to look at least 20 to 30 years into the future. The problems of future land commitments and the life of transportation facilities extend the time horizon of urban plans. Forecasts of land requirements reflecting quantity and quality are also needed to provide for future transportation facilities.

In forecasting, the role of the computer has been in support of economic forecasting models. In fact, next month I will be a member of a panel in Washington, D.C., where the question will be asked: "Have mathematical models and the computer solved our forecasting problems?" The answer will very definitely be no, but a start has been made. These models, although far from perfect, have forced a rethinking of the economic variables and relationships related to forecasting.

The next phase of the planning process is that of design. Very little has been accomplished to date in this area, but it is here that the most important role for the computer probably lies. After all of the in-

formation has been collected and processed, after the forecasts have been estimated, the central problem of transportation planning or, more inclusively, urban design still remains.

The urban design problem may be defined as follows: Given certain objectives (the varied needs of the urban population), given certain constraints (technological constraints and human constraints) and given the related costs, how do we design an urban pattern and related urban transportation and utility facilities? Despite the large amount of data collected by urban planning agencies, this data is rarely used in a direct manner in urban design. It is not possible with existing analytical techniques to use the data directly in plan synthesis. The relationship between urban design and systems analysis has been very indirect. It would seem that all of this data collection analysis and forecasting will do little good unless it can be integrated into plan design. I, and a number of other urban system analysts in the United States and Europe, are now involved in an effort to develop mathematical programming type models in which it will be possible to quantify objectives, estimate costs, insert constraints and provide urban design patterns to aid the planner in the critical function of urban design.

The last function in planning where computers have had effect and have made great progress is that of plan simulation or test. Simulation models of highway and transit networks have reached a high state of development. Models exist for forecasting the number of trips that will be generated by residential, industrial, or commercial areas. Other models will then distribute this traffic by origin and designation using certain gravity-type formulas. Still other models will assign this traffic, both with or without a capacity constraint, to certain freeways or arterials in an intuitively conceived system. These kinds of traffic models have been applied in urban areas. I think this is greatly to the credit of certain federal agencies, mainly the Bureau of Public Roads, that have developed very comprehensive program packages for the application of these models. Despite these noteworthy advances, these simulation models can only test intuitively conceived networks and do not aid directly in the design process itself. Future development of design models will probably eventually obsolete this class of simulation test models.

So much for the application of computers to system planning and design. The other function of the computer, and up to this time not a very important one, but one that has great promise, is the use of the computer as part of the transportation system itself. In both types of urban transportation, transit systems and individual vehicle systems, the role of the computer is becoming more apparent. Computers are being tested in the roles of vehicular controllers as well as traffic controllers. The roles in planning and design are not independent areas. An example of their interaction in transit is illustrative.

There's a very strong political and even emotional pressure in the United States to persuade cities to adapt rail transit systems. That a traffic problem exists in most large cities is obvious. There's a great tendency, however, for people, not only politicians and the general citizens, but even people with technical qualifications to ignore the nature of the overall problem of the city and its needs and to question whether these transit systems are even economically feasible much less socially acceptable or aesthetically desirable. Some studies in Southeastern Wisconsin have indicated that the free market has already provided an economically sound solution for urban transportation. Fancy rail transit systems, automated or not, often encounter market difficulties because they do not fill any previously neglected market need. For a new transit market to emerge, the pattern of the city or the metropolitan area itself much change. So there's a very strong interaction between the basic design of the urban pattern and the role of the computer in transportation systems. Operating applications of computers in the system may be discussed in two time-sequenced phases.

Looking at the here and now, the first applications in transit systems are under way in the Bay Area Rapid Transit System (BARTS) near San Francisco. Four separate companies are testing control systems, all of which involve computers to a greater or lesser degree. The most complete system includes all three separate uses: in vehicular control, in transit scheduling from a series of satellite computers, and finally in overall control from a centralized computer. The second application, in dispatching, schedules the transit system in such a way as to provide service with minimum cost, and the third is in the more conventional areas of business data processing extended most noticeably into fare collection. These systems are now being evaluated. Only one of the four systems actually involves using a digital computer for all three of these roles. Two use analog computers for vehicular control. Two of them use an additional computer for dispatching, and one of the systems applies analog computers to vehicular control dispatching, and a digital computer only for business data processing. BARTS is very definitely a feasible application. There's no question about it, and the real question here is only which is the best system from an operational as well as a maintenance point of view. It is easy to exaggerate the importance of the

economic effects of transit automation. Automation does play a role in the transit economics, but it is becoming pretty well understood that this role is not economically decisive. In other words, if a profitable transit market does not exist in an urban area, automation is not going to change the situation. There was a great excitement a few years ago over monorail systems. There are various other emotional flurries at times towards other transit system proposals, but even with the aid of the computers and automation, the limiting factor is still an adequate market demand in most urban areas. A minimal level of transit demand is required to support a transit system. The computer is not likely to influence this basic market demand.

Another important "here and now" application of computers is in traffic control systems. Some years ago, the first attempt at traffic control using computers was initiated not in the United States but in Canada. Traffic Research Corporation developed a system in Toronto. This system has been extended to the control of 300 intersections. These traffic control systems, by controlling traffic signal cycling in response to volume and spacing of traffic, can actually increase the capacity of systems as much as 50 or 60 percent. High hopes exist for a system being developed for New York City. The basic limitation in such systems does not lie with the sensing equipment, the communication equipment, or the computer, but, as in most process control applications, with the knowledge of the process characteristics. Little is known about traffic flow as a process. Although it is possible to simulate transportation systems in an aggregate sense, knowledge of microscopic traffic flow is still very slight. The success of the traffic control system in Toronto has been dependent, I think, on experimental knowledge not based on a theoretical model of the traffic flow process.

Looking into the future in the light of present accomplishments, it is evident that applications to date, except in urban design, will probably not revolutionize urban transportation because they really do not relate directly to the basic problem of urban transportation. There is much concern about this urban transportation problem today. The HHFA (Housing and Home Finance Agency) has been raised to a cabinet-level department. The HHFA has sponsored a number of transit demonstration grants. The success of these programs has been somewhat questioned. President Johnson has recently placed great emphasis on transportation with his recommendation of a cabinet post for this function. New studies are aimed at the formulation of a National Transportation Policy. A major program has been initiated for the northeast section of the U.S. known as Megapolis. The Department of Commerce has recently sponsored a series of studies at Cornell and MIT. The Bureau of Public Roads is also developing studies in this same area. Finally, all of these studies must solve the one basic problem. Amidst all of the hand-wringing and arm-waving, it really reduces down to one fundamental problem: urban areas. The way such areas are developing is fundamentally toward very low-density, sprawling-type development. With this type of development, the automobile, however horrible it may seem to some planners and other people, is actually an ideal form of individualized transportation. It may not seem desirable at times for the community, but it is for the individual. People can complain about congestion, air pollution and other ills, but until some alternative to the automobile is suggested (and it certainly will not be in the area of rail transit), until some system is developed which has flexibility, which faces the problem of air pollution, and which is able to provide something equivalent to the automobile, the situation is probably not going to change much.

Urban transportation studies have now analyzed a large amount of data to discern the nature of transit-riding. They have discovered that many of the people who utilize transit have no other choice. They are known in the trade as "captive riders." Given a free choice, many of these riders would not choose transit most of the time. Some of the new transit systems being suggested do not seem to face the captive nature of transit riders. Future increases in transit ridership must come from the noncaptive, who must be sold on the benefits of transit travel.

Some of the new designs are not only technically weak but psychologically unattractive. Vehicles, that for all of their automation technology still resemble a New York Subway Car (with standees during the rush hour) are probably not going to convince many people that they should abandon their automobiles for the transit car.

If there is a solution, what role will the computer play? Although no generally acceptable complete solution has yet been forthcoming, the general consensus seems to involve an individual electric vehicle that can operate in these low-density areas. The Cornell Study suggested an Urbmobile. The Urbmobile could be driven locally for pickup and distribution and also on automated freeways for travel over longer distances within the urban area. Although this system is still at a very conceptual stage, it at least has the potential for the solution of the basic urban transportation problem that involves high volume traffic in certain areas at certain times, and simultaneously low-volume travel to a wide dispersion of trip origins and destinations. A system

such as the Urbmobile will obviously require computers for vehicular control and for traffic control.

In other areas of transportation, only partially urban and mostly intercity, the computer will also play a major role. The automated highway for intercity transportation will probably come along in the next 20 or 30 years. The problem is not as overwhelming as the urban one and probably not as critical, but is much easier to solve with existing technology. The problem of terminal operations and freight transportation will also very definitely involve the computer.

In summary, the role of computers in urban transportation systems will be important and sometimes crucial. This role will definitely not be peripheral. People now seem to desire a better urban environment, but the complexity of the urban patterns has need of the computer to solve the problems of urban design. The computer will also have a role in the operation of transportation systems. To realize this potential of the computer in urban development, two needs are apparent. There are some subsidiary technical needs in data collection and data processing of computer software for information retrieval and analysis, but I think these are very minor compared to the major problem, the problem we have addressed ourselves to today, namely, the marriage of analytic and substantive knowledge. The greatest technology can be very sterile unless it is combined with substantive knowledge. I think the experience of operations research and management science in industry testifies to this need. Many operation's research projects have failed in application from a lack of substantive reality. In urban problems the need for this marriage of the professional, who is not information technology-oriented, of the politician, and of the information technologist is even more critical. This marriage is the central problem that must be solved if we are to realize the potential of the computer into the field of urban planning and transportation systems.

Panel Discussion Records

A Panel Discussion

The computer industry in the buyer's market

J. A. RICCA, *Raytheon Computer, Chairman*
J. P. ECKERT, *UNIVAC/Sperry Rand Corporation*
W. J. GALLAGHER, *Radio Corporation of America*
R. W. HUBNER, *International Business Machines Corporation*
R. D. SCHMIDT, *Control Data Corporation*

SUMMARY

The panel, in general, agreed that the computer industry is now in a buyer's market. There was one notable dissenting point of view which indicated that the technology is still rapidly changing, particularly in peripherals, and thus, that the buyer's market would not prevail until at least 1975. The following industry characteristics appear to support the existence of a buyer's market.

1. The rate of change of the technology has slowed down considerably, and fundamental breakthroughs are becoming less frequent. There is now a fair degree of engineering standardization.
2. Computer users strongly influence product design, and the product planning procedure of designing to meet user needs has become a basic marketing tool in the industry. Thus, the user has a wide variety of choices at attractive prices.
3. The nature of the competition is very vigorous, and prices are getting lower. In addition, other inducements to buy are common; such as software thrown in and offering of special services.
4. The number of competitors has been reduced in recent years, and it is more difficult for newcomers to enter the race. The entry fee, already high in the past, is getting even higher.
5. Massive investments are being made for tooling and mass production of standard building-block product lines which cover a large percentage of user applications. The introduction of integrated circuits is further accelerating this trend.

CHANGE OF TECHNOLOGY

In the determination of whether the computer industry is still in a seller's market or has entered a buyer's market, the state of the technology is pivotal. To facilitate this appraisal, computer history has been divided into three eras of technology as follows:

1. Early era—1943-1950. The industry was forced to use existing components, such as vacuum tube logic and Williams tube storage. There were considerable new system ideas but user requirements were not very well known or understood.
2. Growing era—1950-1960. Considerable development of components specifically adapted to computers took place in this period, and the so-called solid state computer emerged using transistors, diodes and ferrite cores. Magnetic tape units, disks and other mass storage units came into being. Although the system concepts were not basically changed, refinements were made in adding items like index registers and floating point arithmetic.
3. Refining era—1960-1975. Discrete transistor and diode circuits are now being replaced by integrated circuits. This trend has just started and will continue for several years before IC technology stabilizes. There is considerable hope that peripheral development will move forward at a rapid pace and catch up with the highly developed central processor. Refinements will continue in system concepts but no fundamental change is expected.

The state of the technology is changing and will continue to change. The key issue is whether the rate

of change of the technology and the frequency of fundamental breakthroughs have created the characteristics requested for a buyer's market. It is apparent that the rate of change has slowed down considerably and fundamental breakthroughs are becoming less frequent. These trends tend to create the climate required for a buyer's market.

Another test is the amount of engineering standardization. When computers are being sold on a custom engineered basis, the user pays for all the special engineering and software, and this situation favors the seller. There is now a fair degree of engineering standardization in the industry, and the manufacturers have heavily committed engineering and software expenditures in advance of offering the product line to the marketplace. These practices favor the buyer.

USER INFLUENCE IN PRODUCT PLANNING

Now that the user requirements can be defined, the traditional product planning process has taken over. Each manufacturer solicits user groups and structures new research and development projects that will result in products that satisfy these user needs.

Because of the relatively large number of manufacturers, the user has a wide variety of choices at attractive prices. This is an essential ingredient for a buyer's market.

VIGOR AND COMPETITION

The nature of the competition is very vigorous, and this supports the existence of a buyer's market. It can be argued that the number of competitors has been reduced in recent years and it is more difficult for newcomers to enter the race. The entry fee, already high in the past, has gone even higher. Nevertheless, the quality and the strength of the remaining companies has increased. There are now several competitors who can compete with IBM on a broad scale basis.

Because of the highly competitive situation, prices and profit margins are lower. This would appear beneficial to the user, at least on a short range basis, but could be harmful on a longer range basis if research and development spending for new products were curtailed. This does not appear to be the case, for all companies are maintaining or increasing their research and development expenditures. In order to get back lost profit margin, a general attack is being made on manufacturing costs. Also, continued industry growth will spread the research and development investment over more dollars of income.

In addition to outright price cutting, other related sales inducements have become common. A particularly popular one is "throwing in" software. Special services are also provided as the occasion may warrant.

TREND TO MASS PRODUCTION

As the rate of change of the technology diminishes and engineering standardization increases, a new form of competition is developing in the manufacturing area. The new "building block" product lines and the use of integrated circuits encourage huge investments in plant and tooling aimed at lowering manufacturing costs and improving the reduced profit margins. Because of these huge investments, the life cycle of new models will lengthen, and engineering changes will occur less frequently. The trend towards mass production is a clear indication of the maturing of the buyer's market.

MANAGEMENT OF INDUSTRY: SCIENTIST VS. BUSINESS MANAGER

In the early period of the industry, the scientist emerged as the dominant managerial figure. The key problems were mastering the new technology and building computers that would work with reasonable reliability. User needs and requirements were of secondary importance. Now that the pioneering days are over and computers have reached a high degree of reliability, the business aspects of the industry have taken on increased significance. The business problem is generically difficult because of the complex nature of the computer system itself, the impact of leasing and its requirement for keeping the installation sold over a period of time, the difficult installation and servicing problems of computers and the vast size and growth of the industry. To cope with this problem, industry must develop top managers who are both technically and business oriented. The engineer turned generalist must acquire business knowledge and acumen in order to manage. Similarly, the businessman who comes up from marketing, finance, or other business disciplines must acquire enough technical orientation to successfully manage the business. The development of this new breed of manager has become one of the critical problems of the industry.

The Technical Program Committee invited other members of industry who were not represented on this panel to add their views. The views of Mr. Max Palevsky, President, Scientific Data Systems, Santa Monica, California, are appended below:

A discussion of the computer market place that is not cast in terms of the influence of IBM in structuring that market place has the ring of the discussions about the Emperor's new clothes. The premise of the discussion at the 1965 FJCC—that a basic change has oc-

curred in the computer market—seems to have little relationship to the relevant facts. What are the relevant facts? Every industry has cyclical factors which affect profit performance. These include demand, plant capacities, technological innovations, etc. When demand is high relative to supply, the market is said to be a seller's market. The only measure of this relationship is general profitability. The computer industry, since its inception, has *never* had a period of general profitability. The total losses, excluding IBM, are astronomical and continue to grow each year in spite of constant mutterings about "turning the corner." Probably no other industry of the size and importance of the computer industry has experienced an era of rapid technological change in which a seller's market never existed. Recent examples of industries that have experienced rapid change and have had a consequent period of general profitability are color television and semi-conductors. If such a period will ever exist in the computer industry appears open to question. The cash flow of IBM is of the order of $1 billion a year which dwarfs the strength of even IBM's largest competitors. The discussion of other factors affecting the market place—such as the rate of technological change—are not totally irrelevant but are of secondary or tertiary importance. Because of the character of the corporations that are IBM's competitors, there has been a reluctance to openly discuss the problem of economic concentration. For a professional organization, however, to avoid the central issue is not in the best interests of the industry in the long run. Obviously there is a limit even in American industry to the investment that large corporations are willing to make unless there is a strong probability of return. Without such investment on a broad scale with many competing firms, the potentialities of the computer field will never be fully realized. That is the central issue relating to the kind of computer market we have and the consequences that can be predicted. If this central issue is not treated, the discussion is really about the Emporer's new clothes.

A Panel Discussion

The overseas computer market

Chairman:

MILTON C. MAPES, JR., *Deputy National Export Expansion Coordinator*
U.S. Department of Commerce

Panelists:

DONALD F. ORR, *Vice President of International Operations*
UNIVAC Division of Sperry Rand Corporation

JAMES G. MILES, *Vice President for Marketing Development*
Control Data Corporation

NORMAN J. REAM, *Director, Center for Computer Sciences and Technology*
Bureau of Standards, U.S. Department of Commerce

THEODORE L. THAU, *Executive Secretary, Advisory Committee on Export Policy*
U.S. Department of Commerce

MR. MILTON C. MAPES, JR.

Our procedure will be to devote the first half of the discussion to presentations by the panelists on various aspects of the overseas marketing problem, and the second half to questions and participation from the floor.

I know many of you are deeply involved in export marketing and the international business aspects of computer sales; in 1965 U.S. computer exports are going to run approximately 400 million dollars. Our imports at the same time are running approximately 60 million dollars. It might be significant to mention that computer exports are up almost 300% from the 1958 figures, only seven years ago. At that time our computer exports totaled only $103 million, as shown in Table 1.

TABLE 1. Computers—Exports and Imports
(*values in millions of dollars*)

	Exports	Increase in Exports (*percent*)	Imports
1958	$103	—	$15
1963	302	193	57
1964	369	22.2	57
1965 (est.)	400	8.4	60
1966 (est.)	445	11.3	65

The United States' share of the world computer market is a little difficult to determine. I do have a figure which includes both computing and accounting machines. On this basis the United States has 39% of the total world market in international commerce, followed by West Germany with 14%, the United Kingdom and Italy each with 11%, and France and Sweden with about 9% each. That includes accounting machines, which still comprise a very large part of the market. I suspect that if it were limited to computers alone the U.S. share of the world market would be substantially greater. As to where our computer exports go, I also have some pertinent figures. In 1964 our major cus-

TABLE 2. U.S. Exports of Computing and Related Machines
(*values in millions of dollars*)

	1963		1964		Percent Change 1963-64
Japan	$63	21%	$71	19%	13%
United Kingdom	35	12	62	17	77
Canada	39	13	55	15	41
France	41	14	38	10	—7
West Germany	38	13	28	8	—26
Australia	5	2	20	5	300
Italy	8	3	11	3	38
Switzerland	8	3	9	2	13
Sweden	6	2	7	2	17
Other countries	59	17	68	19	15
	$302	100%	$369	100%	

SOURCE: Bureau of the Census.

tomer was Japan, which took 19% of our exports; our second major customer was the United Kingdom with 17%, and the third was Canada with 15%. The worldwide distribution of these exports for 1963 and 1964 is shown in Table 2.

I'd like to emphasize that the potential for growth of this market in the future is tremendous. For example, the total market for computers in Europe in 1965 is estimated to be about $450 million; by 1970 it is expected to be in excess of $1.5 billion. Our problem is how to penetrate that market most effectively. The United States, if it intends to maintain anything like its present share of the export market, is going to have to get out into the world market more than ever before. The tendency in the U.S. is to concentrate on developing the domestic market. There is very little tendency by most businessmen to export. When you talk to businessmen who have 12 to 18 months backlog tied up with orders merely from the United States, you can't interest them in getting into the intricacies and what may appear to them to be strange procedures of export marketing. I think though, that in the gradually growing one-market world which we find coming upon us with increasing speed, the United States businessman is going to have to get out and sell on the main streets of the rest of the world if he is going to compete on the main streets of the United States itself.

I want to discuss briefly the Export Expansion Program in general terms, and then Mr. Orr will present an introduction to the export of computers. We have had serious balance of payments problems, as you have probably read, consisting of a deficit in our balance of payments. Every year since 1950, except one, we've had a deficit in our international financial account. Since 1958 our gold reserve has dropped off from more than $22 billion to less than $14 billion at the present time. This is particularly significant because the dollar is the primary monetary unit in international exchange, and since that is backed up by the Fort Knox gold hoard, if that becomes substantially lower, or if there should be a lack of confidence in the dollar and unwillingness by foreign nations to hold the dollar as backing for their currency as we hold gold, it could result in an international monetary crisis and a serious collapse of the entire international financial structure. So we have inaugurated a voluntary program limiting overseas investments by U.S. corporations, which has been a subject of considerable discussion. We have also had a voluntary program administered by the Federal Reserve Board to limit all foreign lending this year to 105% of the total of all loans outstanding at the end of last year.

Then we have the National Export Expansion Program, in all its aspects. First, we have the services of the Bureau of International Commerce in the Commerce Department. I am running through these because I think many of you may not know what services are available to you in the international market directly from the Commerce Department. The Bureau of International Commerce has programs carried out by its offices of International Trade Promotion, International Region Economics, International Investment, and Commercial and Financial Policy. We have permanent trade centers in six major cities around the world. These are permanent exhibits of U.S. products and are always available for foreign buyers and businessmen who want to learn what is available from the U.S. Those trade centers are in London, Tokyo, Milan, Frankfurt, Bangkok, and, for the first time this year, in Stockholm. We also have business information centers all over the world and U.S. foreign service commercial officers. There are presently 156 commercial officers with the U.S. Foreign Service, and their work is coordinated by the Commerce Department. Their function is to service American business overseas. We have increased the budget during the last three or four years for trade fairs and trade shows and the Commerce Department has promoted U.S. commercial exhibits. We have trade missions, consisting of United States businessmen, which go over with Commerce assistance. They represent the entire United States business community for their own specialty and arrange for agency relationships, overseas branches, licensing agreements and even initiate negotiations for joint ventures. These businessmen do not represent only themselves. They represent a broad sweep of the firms involved in their line of business. These trade missions have been extremely successful in establishing relationships and doing business for U.S. firms overseas.

We have in the United States 42 field offices of the Department of Commerce. I would urge any of you who are interested in getting into the international trading community and have not done so, that your first point of contact should be the field office nearest your home base. It can supply you with information, printed material and very genuine technical know-how on how to trade overseas.

We also have 42 Regional Export Expansion Councils, which are composed of local businessmen in each of the regions served by the Field Office. They have an operation whose purpose is to increase the participation of businessmen in foreign trade. The name of it is Operation 10,000—the object being to get 10,000 additional businesses into the overseas market. In addition

we have the National Export Expansion Council which is composed of top level businessmen from all over the country. It recently appointed three action groups. In the last two months these groups have been studying specific problems—one doing export financing, one studying ocean freight rates, and the third working on tax incentives for export.

The Export Expansion Program is largely not subject to being programmed in the electronic sense. One major exception to this was recently acknowledged, when the Business and Defense Services Administration in the Commerce Department established a computer program to bring together overseas trade opportunities and the export capacity of United States firms. Approximately 50,000 U.S. manufacturers are now registered for this program. The idea is that as trade opportunities come in from overseas they will be mailed to those equipped to handle international trade. Perhaps much more will be and can be done in the line of automating and programming the export effort. At this time not very much has been done.

I ran across an interesting quotation the other day by Dr. Herbert Simon in his book on automation discussing the problem of the general problem solver. He stated: "Problem solving proceeds by erecting goals, detecting differences between present situation and goal, finding in memory or by search tools or processes that are relevant to reducing differences of these particular kinds, and applying these tools or processes. Each problem generates sub-problems until we find a sub-problem we can solve—for which we already have a program stored in memory. We proceed until, by successive solution of such sub-problems, we eventually achieve our overall goal—or give up."

Our overall goal here is to increase our overseas sales of computer materials. Fortunately we can break this up into sub-problems by areas and types. This panel has been designed primarily to do this.

With that introduction, I want to introduce Mr. Donald F. Orr, Vice President for International Operations of Univac Division of Sperry Rand Corporation. Mr. Orr is in charge of all the Division's international operations (engineering, marketing, and manufacturing) overseas. He has been in New York for the past five years. His previous 13 years were spent overseas with Sperry Rand and its predecessors. He is a graduate of the George Washington University School of Foreign Service. His subject is going to be "A General Introduction to Computer Marketing Overseas" with emphasis on Europe, both East and West, and the developing countries. Mr. Orr will also discuss some specific problems relating to the import of computers.

MR. DONALD F. ORR

During the past 13 years, the number of computers in the world has grown from a few to over 35,000 systems. In Japan and Europe alone some 7,800 computers are in use today, representing a value of $1.3 billion. This market is growing at the rate of about 20% a year, and it is expected that more than 22,500 computers will be in use in this area by 1971, representing a purchase value of over $4 billion.

The steady growth of the European market should bring it to about the same level as the United States within another decade. This is on top of the fact that because of the traditionally lower labor costs in these markets, the economic savings that a computer installation offers a businessman are not quite as easily justified as in the United States. Therefore, as a general rule I have found among European and Japanese users a relatively high degree of sophistication in the use and application of their computers.

If anything inhibits the growth of the computer market in Europe, it will be the shortage of qualified programmers and operators. At this time there is a need for 120,000 people with computer knowledge, and by 1971 this demand is expected to rise to 300,000. The U.S. computer manufacturer will face increasing competition from the local manufacturers in a growing number of the foreign countries where we are now selling our products. These competitors have indeed mastered the art of building computers and in some cases their technological designs are in advance of our own. To be competitive we must be fully knowledgeable of the hardware and software needs of these markets and understand the specific requirements of our overseas customers. We should be prepared to build these needs into the products to be shipped to these areas.

While export shipments of computers from the United States to other countries are higher than ever before, the ability to import computer systems into many of these countries at the same time becomes more challenging as time goes by. The changing tariff picture in Europe, the emergence of the European Economic Community, restrictions on importations of certain sizes of computers into Japan, are but a few of the factors which oblige the U.S. manufacturer to seek new means of maintaining his share in these markets and to be able to continue to support present customers.

Such steps have brought about a variety of arrangements such as joint ventures and licensing for manufacturers, prevalent in Japan, mergers and partnerships with foreign computer manufacturers, and the establishment and expansion of wholly owned U.S. plants overseas. In many cases such actions appear contrary to the government's program to improve our own balance of trade as well as correct the present balance of payments deficit through voluntary restraints on new

investments abroad. No one country, or no one company, for that matter, can feel secure that it has a permanent lead in this fast-growing computer industry that is continually being pushed forward by major technological breakthroughs and the demands for its products. Therefore, the U.S. computer manufacturer is obliged to take such action as is necessary to maintain his present position in these markets.

To meet the challenges the overseas markets offer, our industry is continually seeking new ways of financing our export business. In this regard, for example, some provision for financing the leasing overseas of U.S.-manufactured computers would be helpful.

Turning to Eastern Europe for a moment, the lure of large potential markets within the Soviet bloc has been getting increased attention from western manufacturers. While it can be assured the need for computers in these countries may be as great as in Western Europe, in proportion a relatively small number of computers are in use behind the Iron Curtain today. Most of these have either been supplied from Russia or from firms in Western Europe.

Up until now, our own Export Control Act, which classifies computers as strategic goods, has restricted U.S. firms from doing business in Eastern Europe while Western European manufacturers freely export equipment of comparable technological design into these same countries. Perhaps we may hear a little more on this subject from my co-panel member, Ted Thau, later this morning.

Building a computer market for the less developed areas in the world should present us the greatest challenge of all. This area encompasses parts of Central and South America, the Middle East and Africa, and the Far East outside of Japan and Australia, excluding, of course, the Chinese mainland. These markets are indeed far flung on the map. They represent more than 40% of the total world land area with a population of 1.2 billion people. This area has probably the greatest need for computers, but at the same time is the least prepared to use them effectively.

Mr. U. Thant, Secretary-General of the United Nations, has said, and I quote: "The computer is the means by which the developing countries can bootstrap themselves to reach the technological level of the industrialized countries." Almost without exception, each of these areas is caught up in a fast-growing internal economy coupled with the entry into world trade on a competitive basis that is calling for the need for better controls, lower costs, and more efficient ways of conducting business. Electronic information processing techniques, through the use of computers, will play an important role to bridge the gap between their century-old business practices and the modern methods of business administration, control and decision making.

Before this can become a reality however, there must be faced up to and resolved the problem of a severe scarcity, and in some areas an almost complete lack, of qualified people who can be trained to make use of computers in business. In commenting on this, E. Dinah Gibson of the San Diego State College, who has studied the problem, stated that many of these countries are still not even teaching business administration in their universities or other institutions of higher education. This is a "must," he feels, as this is the base on which business data processing must be built. The development of the computer market, therefore, will be limited to quite an extent by the business education background of business executives at all levels until a way can be found to provide this background and ability for them.

The manufacturers are still the original trainers in many of these less developed areas. Several companies have set up training facilities such as in North Africa where customers and prospect personnel may come to be indoctrinated. This is not enough as it does not go far enough. It is my suggestion, therefore, that through our own government and/or U.N. sponsored programs, together with educational bodies of the more developed nations, we join together with the governments and centers of learning in these lesser developed countries to provide a solution to this educational program for training people not only in electronic information processing techniques, but in the basic concepts of business itself. This program would be a big step forward in enabling these countries to make the technological and economic progress that is essential to their citizens' well-being. I am sure that the various computer manufacturers throughout the free world could make a worthwhile contribution in one form or the other through such a program and would be ready to collaborate if called upon.

At the present time it is estimated that there are less than 350 computers in use in the less developed areas about which we have been speaking. These are mostly being used by government agencies and in some universities and by the larger, internationally oriented, foreign-owned companies. This total could well increase by tenfold in the next 10 years, provided skilled personnel are available.

The establishment of adequate sales, support, and service facilities on the part of the manufacturer in the less developed countries is vital if he expects to successfully compete in these markets. This will call for in-

vestments of both money and talent. The volume that could be anticipated from such a market and whether this can be profitably obtained will be an important factor in making any decisions to set up these facilities in the first place. Since we are dealing in areas which may be plagued with unstable currencies, higher rates of borrowing and other requirements such as prior deposits, restrictions on remittances and so forth, a means of sound financing of our computer export sales to these countries is a very important factor.

As a means of sharing part of the risks of doing business in these areas, local distributors may be the answer. They would be fully acquainted with the local market, understand their laws and business customs, and, hopefully, provide the necessary local investment and the financing of resulting sales. Joint ventures, made up of the manufacturers and the local interest, thus assuring the manufacturer of a share in the development of the business as well as overseeing sales and service standards, is still another approach. At Univac we have been successfully operating overseas using a combination of these methods in addition to a substantial number of wholly owned subsidiaries and branches.

Day to day operations are another thing. Underdeveloped and overloaded communications will restrict the applications of real-time techniques and other features that American manufacturers can offer in their computers today. Software, as we provide it, will not necessarily meet the requirements of these areas and, if anything, should be made more easy to use than it is today. Electrical power requirements are different and vary even between locations in the same country. Such concepts as operations research, CPM, and other management aids are not being applied and even less widely known. We can expect to have the computer operating in environments far below what we would consider ideal, or possibly even acceptable, in the United States. Unstable power supplies, problems of heat, dust, and humidity, are a few of the situations that the computer exporter may be faced with, depending on the country.

Spare parts backup and logistics involved in supplying and maintaining these inventories in the respective countries where computers are installed is an effort which gets special attention at Univac. Many of these parts are carried on the strategic commodity classification list and require export licensing to ship from the United States. So that we may better serve our overseas customers we are hopeful that the Department of Commerce will see fit to liberalize some of these regulations, particularly for shipments within the western world.

The challenges and opportunities that lie ahead for the American businessman in the less developed countries can be shared with others besides the computer manufacturers. I envision an important role that can be played by professional groups such as the independent consulting firms and the EDP service organizations. There is a real need for "turnkey" type of services to be made available and offered in these areas, which would provide a potential user with a complete EDP systems service encompassing all aspects of problem definition, development of procedures, system installation and operation of the system until local capability has been trained to take over. With the resources and experience behind these professional groups, this should be a natural as well as profitable expansion of their business.

I also envision and encourage the leaders of the emerging nations that already possess EDP know-how to pool their financial and human resources and establish national government computing centers in their respective countries, thus providing electronic information processing techniques on a "utility" basis to all segments of local government and commerce. In this way, all may reap the benefits that computers offer.

There is no doubt in my mind that the overseas market offers a real potential for American computer products in the course of the next few years. Our courage to face up to the challenges these markets present to us, and our ability to meet the competition from foreign computer firms can represent an important contribution to the President's program for export expansion, which Mr. Mapes has mentioned earlier, as well as provide new profits for the American exporter.

MR. MAPES

One of the founders of Control Data Corporation eight years ago, James G. Miles has a Bachelor of Science in Electrical Engineering from the University of Nebraska. He worked on radar development during World War II and somewhere along the line picked up an LLB from St. Paul College of Law. Jim's first subject today is going to be marketing to the countries of Eastern Europe. As Don Orr mentioned, there are a number of governmental problems involved here. We will hear more about them later from the government's side. Secondly, Mr. Miles will discuss problems of marketing in the developing countries, and third, the potential for computer sales as devices for impact on the less developed countries, promoting their development and promoting the interest of the free world in these areas.

MR. JAMES G. MILES

The people sitting in this room this morning are privileged to be associated with probably the most dynamic business in the world today—the making, selling, and using of computers.

Probably never before in history has a single set of tools been developed which inherently contains so much potential for constructive use for all people in the world. We are making extremely good progress in the use of these tools in this country, a fact which is well publicized. And the rest of the world observes, and all desire the same for their countries.

You are all aware of the literally thousands of facets of our economy and lives in which computers are playing major roles today—from education to forest management, from heart analysis to steel mill controls, from banking to advanced communications systems, and on to space systems. And soon, the use of computers in total management systems to permit the optimization of our business enterprises will be achieved. And the rest of the world observes and desires the same for their countries.

There are several ways to look at this: Consider such a large capability and such a large technological lead and such a monopoly in the computer power of the world (in this regard I refer to the fact that 95% of the world computer market is vested in U.S. manufacturers) as carrying with it a proportionate responsibility. We are, in effect, the inheritors of a wonderful set of talents in this country—a combination of creative energy, financial capability, and a free enterprise system which has permitted this. By virtue of this extreme good fortune, we are, or should be, in the position of beneficial trustees. As a corollary, with this goes an extraordinary set of responsibilities which virtually place, or should place, their owners in the position of "trustees" in the legal philosophy. These talents, under this context, are not entirely the property of the presumed owners to hoard and dispense at their sole pleasure without consideration for others.

When we discuss the *export* of computers, and the export of computer technology, we are discussing a combined set of problems which probably offer more opportunities, and which are at the same time complicated by more problems than almost any other export subject that could be discussed today. These problems are not all internal to the United States.

- They cut across the technological capabilities of many countries to make and use computers, as Mr. Orr indicated.
- They involve complicated problems of international finance.
- They involve the need (in the less developed countries the desperate need) for computers to help them organize, build, and operate their economies.
- They involve the *laws* of the United States, and the *laws, policies,* and *objectives* of many countries who want to import our computers, technologies, and our data processing techniques.
- These problems heavily involve the foreign policy objectives of the United States, and heavily interact, on the other hand, with the aspirations and foreign policy objectives of many other countries. And, when the U.S. attempts to put undue restrictions on the exports of these commodities, we often obtain very serious reactions from the other countries.
- These problems involve also multinational trade agreements such as the COCOM (Coordinating Committee) Agreement, which in many respects are more realistic and liberal than the laws and policies of the United States.
- And these problems involve a "balance of power" in technology which can simultaneously work both for and against the United States, depending on how we handle it.
- Above all, these exports, or their denial, offer the greatest of opportunities for orienting many countries toward the United States, or otherwise.

Mr. Orr discussed the need for computers in less developed countries and the widening technological gap which is occurring between us and the less developed countries. This is a very real thing. These countries aren't even keeping up, in most cases, with their growing populations and in their ability to feed them, house them, and provide them with the other amenities of life which, by way of worldwide communication and publications, they are easily able to discern is not their lot to have, vis-a-vis the United States. We sit around worrying about whether these same countries may become oriented with the Communist countries. We wonder how and what to do about it. The consequences, in my opinion, of this widening economic gap between the less developed countries and the United States should be of the utmost concern because the continuation of this problem, uncorrected, I believe, carries with it the seeds of increasing dissension and may give these people reasons to believe they should orient themselves with the Communist countries instead of with the United States. So we really have a responsibility in this country to dig in and help these countries to establish their ability to acquire a few computers and get them trained along the lines Mr. Orr discussed. As quickly as possible, also, we must help them develop in-country capabilities to make these systems, so they don't need to rely entirely upon us for exporting or for importing.

I am now going to discuss a subject which is highly controversial—*trade with the Communist and Communist-controlled countries*. Various people of equally good will hold decidedly diverse opinions on this subject. Passions are keen. Some people think of the subject in terms of black and white; they may be right. Personally, I and my company, Control Data Corporation, take a different view.

President Johnson said in his State of the Union address in January 1965, and restated in a speech on May 8, 1965, significantly marking the 20th anniversary of the end of World War II, the following:

> Here is some of our unfinished and urgent business.
>
> First we must hasten the slow erosion of the Iron Curtain. By building bridges between the nations of Eastern Europe and the West, we bring closer the day when Europe can be reconstituted within its wide historic boundaries.
>
> For our part, after taking counsel with our European allies, I intend to recommend measures to the United States Congress to increase the flow of peaceful trade between Eastern Europe and the United States.

And again on May 23 at Lexington, Virginia, President Johnson said:

> There is no longer a single Iron Curtain. There are many. Each differs in strength and thickness, in the light that can pass through it, and the hopes that can prosper behind it. . . .
>
> We will continue to build bridges across the gulf that has divided us from Eastern Europe. They will be bridges of increased trade, of ideas, of visitors, and of humanitarian aid.

I think we should take the clue from this enlightened attitude of President Johnson. Whenever I go abroad, my main objective is to make friends for the United States. I believe, among other things, that if I can achieve this, the relatively simple process of taking orders will essentially take care of itself.

Then comes the frustrating part.

I am told by people in Washington on one hand, before I go on these expeditions, that I can pursue these objectives without limits in all countries including Soviet Russia, but excepting of course, Red China and Cuba and one or two others. Then when we go out and obtain some of these orders we come back and find that there is no feasible way to obtain export licenses. This is frustrating to the people in the Eastern countries because, I guess, they presume by the very fact that I showed up in their countries there is the opportunity to do business. This does not too often turn out to be the case when the export license is denied by the United States.

Computers are, because of their inherent usefulness, among the most valued, valuable, and useful items for foreign trade. Consider the *people* they reach. First of all, they reach the *top people* in each country. They reach the top educators, the top scientists, and the top businessmen; the top industrialists, and the top government people. There have been several countries in which we have obtained large computer orders where the head of state, the top man, has signed off on the order before it has come to us. If you are building bridges, or if it is your intent to build bridges, and it is the intent to try to wean some of these countries away from their past orientations (which, incidentally, many incurred because of enslavement, and not because of desire on their part), I believe that *computers* are the *most important commodities to be traded*. I don't think there is any particular point in going over and offering to trade tables and chairs, because they can probably manufacture such items as well or better than we can. Besides, the trading of tables and chairs does not reach influential people; such items are not matters of top national concern. It's the *people* these reach that is the important thing, and not just on Control Data's behalf, either. As we would bring computer users into the fold of our customers, they automatically join an international fraternity of computer users which exchange information and techniques. Fortunately most of this international fraternity is oriented toward the West politically and philosophically. These new users will become associated with a myriad of users in the United States and in West Germany, and in Scandinavia, England, Australia, France, etc. I believe that *if you want to build bridges, this is the way*. If you want to bore big holes in the foundations of Communism, this is the *best* way to do it, because of the *influential* nature of these tools, and the *people* that they reach.

How do we proceed to obtain the conditions by which this can happen, assuming that it is desirable. Well, of course, as Mr. Thau is going to tell you in a few minutes, there are many, many influences at work in the U.S. in this regard. First of all, there are laws of the United States which were established by the Congress; these laws include the Export Control Act, the Battle Act, and several others; these laws establish the basic framework of ground rules. On the other hand if you read those laws, it is very interesting to see that there appear to be many interpretations of those laws which don't necessarily have to be read into them. Oftentimes the Administrative Rules and Regulations which derive under these laws are much *more restrictive* than at least I read into the four corners of the words of those statutes. I'm not the only one that holds this opinion.

Then there are international agreements, such as the

COCOM Agreement which has rather set up some terms and items which involve supposedly enforceable multinational embargoes regarding certain so-called "strategic materials" shipped to the Eastern countries. Different countries put different interpretations on these multinational agreements, but our country puts the most strict interpretations on them, or at least imposes different sets of regulations which are more restrictive—and, I believe, too severely restrictive.

Beyond that, of course, there are the *executive functions* of the U.S. government. The major departments reporting to the President—Commerce, State, Defense, Treasury and Justice—all get involved in these matters and all insert their opinions. And I'm not saying that's entirely wrong, because the Export Control Act provides that this should and must be the case. But all of these opinions and decisions pile up on top of each other with compounding-restrictiveness, I believe. It's possible, of course, that the only way to get this really clarified is for Congress to study the matter again and come up with a revised or a new set of laws. And I certainly don't blame the very competent administrators in the administrative branches of the government who are charged with the responsibility of enforcing these laws for their overall-conscientious approach to the problem. But the laws, I must say, might be interpreted in any of several ways, and may be somewhat unclear.

About the most difficult problem that appears here is, apparently, one that centers around a word that is thrown around, called "strategic." I suppose there are many interpretations you can put on this word. I don't recall if it actually appears in the Export Control Act, or some of the other Acts, but the interpretation is always one of "strategic-for-military-purposes." This appears to be of very great concern, particularly among Defense Department people, and I'm sure that no one in this room would deny the "possibility"; I certainly won't.

It is mainly a matter of *who* is the prospective customer, and *what* he wants to *use* the computer for. Most prospective customers are quite frank and open regarding what they desire to use their computers for—for management information systems, industrial process control, product engineering, etc. In this regard we should rely on their representations with a reasonable amount of faith, meanwhile observing through the customer liaisons open to the computer manufacturers, that the computer is in fact being used as represented.

An item which appeared in *Business Week,* October 30, 1965, reported that COCOM regulations were recently modified to allow the shipment of nuclear reactors into the Eastern countries, provided that sufficient controls are imposed to allow inspection which would assure that those reactors were being used for peaceful purposes. I presume that this inspection includes that by-products of these reactors (which can, incidentally, be fuels for nuclear weapons) can also be monitored. Now, if nuclear reactors can be shipped, with their potential for strategic weapons usage, I cannot see why computers cannot be shipped, because at most, computers are only indirectly possibly useful or "strategic" for military purposes. Computers are not "weapons" per se. I know even this point could be debated, so I will get off it.

But, I say, there is a greater alternative possibility here than we have tapped, and that it is the possible "strategic" use of these computers for *peace,* to help to politically orient these people toward *our* point of view. I believe there is a very interesting problem here, the answers to which hold tremendous opportunities for the United States, and which can, and should, be resolved in favor of the United States, vis-a-vis our competitors.

MR. MAPES

Mr. Norman Ream is one of the best-known figures in the field of systems research in this country today. He received his Bachelor of Science degree at the University of Illinois and is a licensed Certified Public Accountant in Illinois and California. Starting out in the petroleum business with Pure Oil, he moved on to become Director of Accounting Research for IBM. After a subsequent tour of duty with Lever Brothers, he moved on in 1953 to Lockheed Aircraft Company, where he served as Director of Systems Planning until, only a few weeks ago he was appointed to his present position as Director of the Center for Computer Science and Technology in the Bureau of Standards. Mr. Ream is going to discuss problems of marketing computers in Asia, especially in Japan, as well as in some of the less developed countries. He also has a few ideas on the educational aspect of overseas computer sales, which I think he may share with us.

MR. NORMAN J. REAM

I think most of us are in favor of increased freedom of trade with the rest of the world, and there are usually two sides to every question, but Mr. Miles' suggestions on trading with the Communist countries move me to express the personal opinion that we have to be very careful about trading computer dollars for loss of security. But that isn't what I came here to talk about today.

Mr. Orr touched on the computer market in Japan. My knowledge of that market has developed from a series of lecture trips I have taken to Japan over the last three or four years. I have visited Japan about eight different times to lecture to the Japanese in-

dustrial communities concerning the use of computers in the United States and to discuss with them the potential of their use in their country.

The Japanese computer industry is very interesting in that today there are six major computer manufacturers active in the Japanese market. We have American computer manufacturers who are also active there—IBM, operating a large plant through their Japanese company which is 96% owned, and Univac being active in a joint venture company. Also there is NCR, Burroughs, and CDC. Japanese companies that are most active are Fuji Communication Apparatus Company, Miksabichi Electric Manufacturing Company, which is part of the Miksabichi family, the Oki Electric Industry Company, the Tokyo Shaboro Electric Company and Hitachi Ltd. Hitachi is the largest company in Japan and has a very active computer manufacturing group.

The latest available figures I was able to secure date back to September 1963, at which time there were actively used in Japan 440 Japanese-manufactured computers and 285 American computers (of which 163 were IBM). Of course when you think that the first computer was introduced in Japan in 1959, this figure has now probably about doubled. But in increasing at this rate the Japanese companies have gained a greater percentage of the market than the American companies. In other words the balance is swinging to the Japanese manufacturer.

It is also interesting of course that Japan ranks second to the U.S. in the use of computers and I think we will see a diminishing number of American computers in use in Japan in future years. I will try to tell you why I feel this way by talking about some of the problems in the development of the use of computers in Japan. Their problems are not too dissimilar from many of those we have here in the U.S. There is a great shortage of skilled personnel. This they are attacking through their universities, probably on a more formalized basis than we are. There is a direct emphasis on this. Also recently there was a program introduced in Japan by NOMA—the Nipon Office Management Association; however, we shouldn't compare it to the NOMA in the United States. It's quite a different type of organization. They are embarking on quite a computer programming training course and they will probably have 20,000-30,000 students in this course within the next 18 months. There also exists in some areas in Japan the problem of management attitude that is a sort of reluctance. However, I think that this reluctance on the part of management in Japan is not nearly so pronounced as it is here in the United States. They are also faced with the problem of program language development. Most of the programming languages in use in Japan are imported from the U.S., although there are activities under way to develop their own languages.

Additionally, they also have some rather distinct problems which are not as familiar to us. One is the economic evaluation of the use of computers. Mr. Orr touched upon this. They cannot justify computers solely on the basis of replacing people. Currently their labor costs are quite some bit below ours. But this problem is changing in a very fast rising economy; however, it will be quite some time before their rates will be comparable to ours. Consequently they are looking toward the more sophisticated areas which Mr. Orr mentioned.

A second problem that they have is that they have only nominal government support in the defense area and admittedly in our country our government has contributed very heavily to our defense efforts and all our research and development efforts in the computer industry. Japan also has minimal use of computers in the scientific areas because most of their industries are not heavily engaged in R&D activities, not as perhaps many of us would like to think of it. We like to think of Japan as being very active in R&D, and they are in the electronic areas, but in many other areas they look to licensing arrangements and the import of technical know-how from outside of Japan. This is understandable, considering financial conditions.

They have another peculiar problem. The labor market is not nearly as fluid in Japan as it is here in the United States. This is due to the paternal instincts which are deeply imbedded in the Japanese industry, which is sometimes referred to as "lifelong employment." When a man or woman joins a Japanese company, normally they stay with that company for the rest of their business life. This means that in the introduction of computers into a Japanese company they must start at the very bottom and train their people to become accomplished and acute. They cannot proselyte, as is done here in Las Vegas, where there is sort of a trading mart for personnel. I believe, however, that in spite of all the problems they are faced with, they are going to make a very extensive use and a very sophisticated use of computers, especially in the management areas. I believe that their management will react faster to change, once the requirements for that change have been determined.

Looking at the Japanese government support of the computer industry, we find they are extremely active and feel that the development of a very strong computer industry is basic in developing the well-rounded electronics industry, for which they are very famous.

In 1961 they established the Japan Electronic Computer Company, which is a leasing company to Japanese industry. A special committee has also been set up under the direction of the government. They are doing extensive work in peripherals, and they are developing equipment which will handle the common language, which will ease the use of computers. One of the recent developments on the part of Hitachi has been the electrostatic high-speed printer, which prints at 6,000 lines a minute. Two of these are currently in use in Japan and of course they are developing printers which will print the common language, this being one of these developments. Their manufacturing techniques are extremely advanced and their quality assurance programs are equally as good as any of those that are in existence in the U.S.

While talking about some of the problems of the American computer manufacturer, the Japanese government has for all practical purposes banned the import of small and medium sized computers into Japan and those that are imported from foreign sources have a 25% duty on the import. Of course this places the foreign manufacturer at a considerable disadvantage unless he is working directly with a Japanese company. IBM has been manufacturing the 1401 and has been granted permission by the Japanese government to manufacture the 360-20 and 360-40. This means that most American manufacturers in order to get into the Japanese market are probably going to have to go through licensing arrangements or joint venture companies with Japanese nationals.

There are several types of companies that can be formed in Japan. One type is 100% American owned. This is an extremely difficult task to accomplish and one that is not used very extensively at this point. The joint venture method is easier and one that is normally used. In this situation we would find a Japanese control being exercised. In other words, they would own more than 50%. There are many problems associated with American companies working in Japan, i.e., the problems of transport of technical know-how into Japan and the usually associated slow start-ups associated with a company until good working arrangements can be made.

In Japan we are going to find that the Japanese government, while it is a democracy, is much more influential than we may suspect. We are going to find that they will be very influential in supporting the use of Japanese-developed computers in Japanese industry. This does not mean that American computers will not be used in certain areas, but I do think that the percentage of use of the American computers in Japan will decrease in time.

As we look to the Orient in the lesser developed areas, we have an entirely different situation. The Japanese, American, and Western European manufacturers will be competing in a more or less open market. Also there are many problems associated with the use of computers in these less developed countries and we are going to find, as Mr. Orr says, that these are initially going to be used in the areas of government, or they are going to be used by the larger foreign companies operating in these areas. I cannot foresee a situation where local companies in Formosa, Thailand, and Malaysia are going to have a need for large computers at this time. This perhaps will develop over a period of time, but I think the market here is much more limited than we would like to realize.

I do believe, however, that in these areas there is a great possibility for the American computer industry to make a great contribution and to considerably improve the image of the United States. This is in the area of education. Those of you who have visited these countries recognize that there is a great void of middle class people and that the industries are not going to grow in these countries until the void is filled. It can only be filled through better education. I believe that there can be an extensive market developed in these countries through an educational field. However, this is an area in which we have not done too well in the United States. Perhaps by real emphasis in this area on the part of computer manufacturers, the computer industries, and the educational institutions, a great contribution can be made.

MR. MAPES

Mr. Theodore L. Thau is the Executive Secretary for the Advisory Committee on Export Policy. He went to the University of Chicago, receiving a Bachelor of Philosophy and a JD from the law school there and engaged in private practice in Chicago and New York. Then he was Assistant Solicitor for the Securities and Exchange Commission in Washington. He spent a good many years as Assistant General Counsel for the Department of Commerce in the field of export control until 1961, when he was appointed to his present position. Ted has suggested that perhaps we could forgo his initial presentation in the interest of opening a longer period of time for questions. I don't think I can let him off the hook that easily, but I can open the question by asking him to give us a brief presentation concerning the parameters of the term "strategic," which was discussed a few minutes ago, and also by covering the special problem of the export of computers to France.

MR. THEODORE L. THAU

A few years ago, for someone like myself associated with controls over exports from the United States to

be a member of a panel concerned with overseas markets for anything would seem kind of phenomenal. That's no longer the case and there is good reason why I am here today—because the Export Control Act which I'm concerned with is no longer regarded as a complete bar to exports to the East European Communist countries, including the USSR, as it once may have been considered.

As a preliminary matter, however, I want to tell you—and this may seem a bit unfair, but I assure you it had to be worked out this way in the interest of brevity, among other things—Mr. Miles and I made an agreement last night that if I didn't answer specifically everything he said today this would not necessarily mean I agreed with him.

First: Computers are licensed for export from the United States to all free world countries, excepting Canada, and to the East European Communist countries, including the USSR. I except Canada because since World War II days we have had an arrangement with that country whereby we do not require export licenses for goods intended for use and consumption there.

Our reasons for requiring licenses to export computers to the free world countries are not that we are concerned with preventing them from getting computers (with certain limited exceptions which I am going to refer to a little later), but rather in order to prevent unauthorized transshipments, re-exports, and diversions from the free countries to the Communist world, including the East European Communist countries, the Asiatic Communist countries, and Cuba.

We also require validated licenses fo rthe Communist countries that I have described. Those countries really no longer comprise one world, you know, but several groups, toward which we have different levels of controls, reflecting the requirements of the law and policy of the United States to safeguard our security and welfare from those who might have hostile intentions toward our country. This means that we do not deny all applications for all licenses to all these different groups of countries. We use the licensing technique, instead, as the device to screen, from the strategic point of view, orders from Communist countries for computers. We do not grant any licenses for computers to the Asiatic Communist countries, for strategic and foreign policy reasons. I will not go into that any further here today. The same applies to Cuba.

With respect to the East European Communist countries, our restrictions are more selective. You will notice that I have been careful not to refer to the Soviet bloc. The reason is that we no longer regard all of the East European countries and the USSR as constituting a bloc. Just a couple of years ago we came to the conclusion that it no longer made sense to refer to anything called the Sino-Soviet bloc. So, now we consider that the East European Communist countries and the USSR are all to be treated individually under our export policies. For some of those countries, this will mean more favorable treatment than others. For example, since 1958 we have treated Poland more favorably. We will even allow her to receive strategic goods, if they are found to be reasonable and necessary to the Polish civilian economy. In the summer of 1964 we entered into negotiations with Rumania, as a result of which we gave Rumania a preferred status, much as we have for Poland. In accord with that preferred status, we allow Rumania to receive certain strategic items, if they can be found necessary and reasonable to the Rumanian civilian economy. Each of the other East European countries must be looked at on its own footing, in the light of our foreign policy and strategic interests relating to the particular country.

What do we mean when we say that we use export controls to screen the strategic from the nonstrategic? If we merely were to limit ourselves to direct military items, and be concerned only with tanks, guns, planes, bombs, etc., that would be one thing. There are controls over items of those kinds maintained by the Department of State Office of Munitions Control. Some of you may have had experiences with that office in connection with applications to export to free world countries items which you know or have reason to believe are going to be used there for military purposes. However, the U.S. Export Control Act, which is administered by the Commerce Department, goes beyond the narrow concept of direct military use. It embraces a concept that might roughly be called a "military-industrial mobilization base." Some phrase of that sort would be aptly descriptive of the area of concern that is involved when the Commerce Department uses the term "strategic."

From that point of view, then—using that concept as our yardstick—we have in the past approved licenses to export some kinds of computers to Communist countries. We approve them now to the East European Communist countries. We also approve components for computers which you may send to your West European affiliates or other free world firms to be used in the making of computers to go to East European Communist countries. We also approve peripheral equipment to go into computers to be made abroad and sold to the East European Communist countries. There is no U.S. embargo on all computers, components, and peripherals for the East European Communist countries.

There has been some misunderstanding about this. It

may have been because, in addition to concern about building up the military potential of the East European Communist countries, the Congress required us in 1962 to be also concerned about contributing to the buildup of the economic potential of the East European Communist countries. However, this does not mean, has never been interpreted to mean, that every computer, because it contributes inherently to the economic potential of an East European Communist country, is also necessarily detrimental to our security and welfare. We have publicly interpreted this amendment as not requiring us to conclude that an item is detrimental to our security and welfare, from the economic potential standpoint, if a comparable item is readily available to Eastern Europe from other free world countries. We have, therefore, a basis on which, even applying the economic potential criterion, we can and do approve licenses to export computers, computer components, and peripheral equipment to East European Communist countries.

I turn now to the kinds of computers, components, and peripheral equipment, that we deny, that we are concerned about, that we ask you many questions about, when you come in to us with your license applications. When you tell us that you are interested in trying to sell these, we ask you, "How advanced is this computer over what is available from the free world without the use of U.S. components, without U.S. technology?" "How advanced is this over any that would be available to the East European Communist countries from their own resources?" We are here concerned with the more advanced types of computers. We are concerned with these because we have been told by the technical experts from Defense and other government agencies, that it is in this advanced area that the United States has a significant lead over the East European Communist countries and that there is more likelihood that such computers will be used in part and substantially for military-industrial, mobilization-base activities than the less advanced types of computers.

You may argue that this isn't necessarily so, and that is quite true. One could use the most advanced computer in the world for peaceful purposes. If we could be certain about that, it would be fine. I believe, however, that one would have a good deal of difficulty in finding a basis for even a reasonable degree of probability of such limited usage. For example, if company X gets an order for one of the most advanced types of computers presently known from a department store chain in one of the East European Communist countries, we may find ourselves having to ask the question: How many hours a week will this computer be needed to do the work of the chain? It may turn out to be that only a few hours a week are needed. What will be done then with that very expensive monster the rest of the week? We know, and it was a thrill for me to see this kind of operation yesterday, how peripheral equipment hundreds of miles away from a giant computer can be so connected to it as to work out very advanced problems, and how many pieces of peripheral equipment can be connected up and operating at the same time and still using only a small fraction of the capabilities of this computer.

Now it doesn't follow, even from what I have said, that there is anything that requires denial on the basis of this or some other single factor. We need full information from you in those cases. We need specific knowledge to be able to meet the requirements of the law and the national policy. We need to be more sure with respect to those types of computers than with respect to the others.

Next, it isn't enough just to tell us that "comparable" technology is available from computers that are obtainable from West European countries. The technology in a wide range of computers may be comparable, as technology goes. But the computers themselves may vary quite widely, and some may be far more advanced than others, even though all use comparable technology.

Of course, we are concerned to prevent our advanced technology from going without authorization to the East European Communist countries. We are also concerned not to have our most advanced computers go. As one of the people here at this conference told me yesterday, the East European Communist countries recognize that by getting our most advanced computers they have the best chance of overcoming most swiftly our technological lead. We have been queried whether this technological lead they want to overcome is only in the economic area, not related to a military-industrial mobilization base. Well, we're not sure. We're therefore concerned. So we will ask you many questions when you come in with such cases.

I would now like to talk a bit about one problem area, outside the East European Communist area, with which some of you have had experience. I refer to the problem that has arisen as a result of the Nuclear Test Ban Treaty, and especially as a result of certain activities of a friendly country that is not a signatory of the Nuclear Test Ban Treaty. We have obligations under the Test Ban Treaty not to aid any country in the development of nuclear weapons or in conducting nuclear explosions and nuclear weapons tests. As a result, the Commerce Department and the State Department's Office of Munitions Control have adopted regulations of a complementary nature, designed to regulate by special licensing procedures, items that

may be used for such purposes, even if going to free world countries, and whether or not specifically designed or modified for use in nuclear weapons testing. As some of you may know, we have expressed considerable concern here about very advanced computers, components, and peripheral equipment.

There is much more I could tell you about this new area of export controls, but time is running out and perhaps in some of your questions you may ask something about it.

A Panel Discussion

The future of electromechanical mass storage

Chairman:
WILLIAM A. FARRAND, *Autonetics*

Representatives of Large Installations:
ROBERT M. FRANKLIN, *Chrysler Corporation*
NORMAN HARDY, *Lawrence Radiation Laboratories*
ROBERT M. GRAHAM, *Project MAC at Massachusetts Institute of Technology*
MARVIN EYSTER, *Woodward Governor Company*

Equipment Manufacturers:
WILLIAM J. BRODERICK, *General Electric Company*
FRANK J. LOHAN, *Bryant Computer Products*
DONALD K. SAMPSON, *Control Data Corporation*
ALAN F. SHUGART, *IBM Corporation*
IRVING L. WIESELMAN, *Data Products Corporation*

INTRODUCTION*

The field of mass storage under consideration by the panel is that which communicates at machine speed in machine form. The areas of concern are both software and hardware from total system to details within the components.

The amount of mass storage presently installed in the U.S. is above 10^{15} bits in an on-line or pseudo on-line manner. Some organizations have made sincere requests for this much data capacity in a single store. Over eight years ago Art Angel described the perfect auxiliary store as an "infinite size memory instantaneously accessible and economically feasible." Ideals like this are seldom met, but the future lies between the present and the ideal. The concern for the future is not what can be out there but rather what is going to be out there. What determines the future is present and future problems in light of the technical and economical feasibility of them. The future, when you get there, should be the past—not still the future. This is our frame of reference.

In order to address ourselves to the future of mass stores, it is necessary to determine their present position. From their shortcomings and inefficiencies, it may be possible to determine their immediate path. The distant future (over 10 years off) is nearly impossible to forecast and is very dangerous for we'll probably live to find our predictions false. It is our intent to discuss the "needs and the possibilities" of certain specific characteristics. By having system users detail their problems and equipment manufacturers discuss their solutions, it is possible to see the cost and utility of equipment and, therefore, to judge the worthwhileness.

THE CHRYSLER 50,000-MILE WARRANTY (RACE) SYSTEM†

The Chrysler Corporation has 40 to 50 random access systems in operation. The 1.2 billion-character on-line RCA 3301-RACE system located in Highland Park, Michigan, is used to support the Chrysler five-

*William A. Farrand, Assistant to the Chief Engineer, Data Systems Division, Autonetics, set the stage for the session (summarized).

†Robert M. Franklin, General Supervisor of Information Systems Planning and Research for Chrysler, reported on the Chrysler system (summarized).

year, 50,000-mile warranty program. It is connected via commercial communication lines to video data terminals installed in regional offices in 23 states around the United States. Each of the 14-inch video data screen and teletype keyboard terminals is capable of getting into the file to find the history on a vehicle. Within 2½ to 3 seconds after a request of a vehicle by serial number, the operation has up to 480 characters of information displayed on the history of that vehicle from the day it was manufactured. A page-turning operation is available which permits request of additional information in a particular vehicle file when 480 characters is insufficient for the complete history. There are 1.2 billion characters on file now, and the program is only three years old. A firm requirement exists for 9 billion characters of on-line storage. An order of magnitude increase from this size is foreseen on this application alone. Each RACE stores 400 million characters on a stock of 256 random access magnetic cards with an access time (random average) of 333 milliseconds. With an increase to 9 billion characters, this average access time will not increase. This access rate poses no problems for the remote inquiry system but poses considerable difficulty when updating the file (present updating time, 20 hours, caused in part by the large block size—650 characters). Quality Control and Engineering people at Chrysler are beginning to appreciate the data which are stored in the file, but present organization of the file makes it nearly impossible to extract information to fill their requests as the file was structured to service the remote inquiry stations.

The file was installed in February with very few problems which were not solved by component selection. The commanding reason for choosing this particular equipment for this particular installation was file cost. At the selection time this was one-third its nearest competitor.

THE LAWRENCE RADIATION LABORATORY COMPLEX*

Current equipment includes two 7094's, two 3600's, a LARC, a STRETCH, and a 6600. An additional 6600 is expected in April and a 6800 in early 1968. Current accessory equipment includes three large Bryant disk files with the 6600, a 30,000 line-per-minute printer, miscellaneous input/output stations for individual researchers' use, and the 12 drums associated with LARC. An IBM 2321 with 3×10^9 bits capacity is expected momentarily and a 10^{12}-bit (equivalent to approximately 10,000 reels of high density tape) storage unit is on order from IBM. In general, the operation of the mass storage elements is repetitious in nature and requires only simple programming to permit simultaneous reading, writing, and computing on different elements. Operations tend to be run in a time-sharing mode during the daytime with long physics problems handled at night when personnel demands are not so great.

At the present time the computers and mass storage equipment is scattered in a number of adjacent buildings. It is planned to move them into a single large room. Ultimately it is planned to have teletype units and TV-type display systems scattered widely over the facility for the convenience of scientific personnel. The majority of these will be within a thousand feet of the computer center. Thus the entire operation is comparatively compact relative to most time-shared systems.

It was noted that the special printer produces about 10 pages a second and eats up several hundred dollars' worth of paper an hour. Consequently, one of the objectives of the organizational planning is to provide more direct access to computer output so that the need for large-scale printout can be reduced. At the present time one microsecond memory having a capacity of 128,000 36-bit words is available connected to six independent memory busses. One of these busses will be used solely by the General Precision disk file which has a 10-megacycle bit rate and 8×10^8-bit capacity. This disk file consists of two units, each having 5,000 fixed heads, of which nine are active at one time. The interconnecting channels currently operate at the 10 megacycle bit rate, but Norm Hardy considers this extremely slow for use with machines such as the 6800 and anticipates that in the 1970's equipment will be available for operation up to 1,000 megacycles.

A PDP-6 unit is used at Livermore exclusively as an executive controller to allocate work to the other units. To obtain maximum efficiency, allocation of disk storage space is crucial. Current thinking is to allocate disk space by pages of 32 sectors of 1,024 words, separated by one-sector gaps. Pages will be identifed by one of 18 standard page angles, and tasks will be queued by page angle. When none of the 18 queues are empty, 95% of theoretical transfer rate will be possible (equivalent to 250 accesses per second if each page is considered an independent access). Methods allow for accessing any arbitrary page or pages from a complete data file without following the entire file through serially. It is expected that the major data flow will be between the disk and the other devices in the system.

*Norman Hardy of Lawrence Radiation Laboratories outlined the Livermore installation and the operations contemplated (summarized).

PROJECT MAC*

The MAC system is basically the first prototype operation of a computer utility on a 24-hour basis. The next version, described elsewhere in this conference, is called MULTICS. The basic objective of these facilities is to provide service to a large number of terminals scattered over a large geographic area. This service effectively permits independent use of the central computer and storage facility by each terminal and permits each terminal to expect that all information related to its activities would be retained in the facility memory until explicitly deleted by that terminal. At the present time Project MAC uses a 7094 computer to service approximately 160 terminals in the New England area of which 30 may be active at any one time. For the type of operation contemplated, the ideal memory is obviously unattainable since it would have infinite capacity and instantaneous access and transfer at zero cost. The compromise approach used to approximate this in the M.I.T. program is to provide a hierarchy of existing types of memory of decreasing speed and increasing capacity within the limitations of available funding and space. Thus, the highest-speed units are core memory followed by drums, disks, and tapes in the present system. It is conceivable that some detachable memory may replace the tape units in the future. In addition to the 7094, Project MAC currently is equipped with one 1302-disk unit, two 720-A drums and one 720. Programs are given core space sequentially, with at most one user control program ready for execution at a time. The general intent of the facility is to develop operating and programming routines to relieve the user at the terminal from any consideration of the type of auxiliary memory equipment with which he is operating and to relieve the management of the detailed format of files at the Center. Thus the Computer Center is undertaking to handle the interconnection between terminals and the actual operating equipment in such a fashion that Center equipment can be changed from time to time without influencing the user's methods of operation. In terms of thinking on the management of the hierarchical memory, it is planned that programs will be transferred to slower and slower storage media when they are inactive, and that active programs will be maintained at the highest level of accessibility consistent with system capabilities and overall user demands. Present thinking is that high-speed drums and/or disks will continue to be the major units required for readily accessible storage in the foreseeable future. The 720-A and 1302 used on the MAC system are both underpowered, as the former has a quarter-second 32,000-word swap time, and the latter has effectively only a single access path. For such service this kind of equipment needs a multiplicity of access paths in particular in order to be of maximum utility for a service center. Obviously the faster the access and transfer times, the better, as it approaches more nearly the capability of an all-core memory.

*Robert M. Graham, Program Coordinator at MAC Center of M.I.T., reported on the Center's system and plans (summarized).

THE WOODWARD GOVERNOR MANUFACTURING CONTROL WITH CRAM*

This company has a rather specialized requirement since they produce and maintain a stock of some 55,000 component elements from which they custom-assemble governors to meet the requirements specified in customers' orders. To provide inventory control, update engineering specifications, coordinate requirements of customer orders, schedule the use of machine tools, and consolidate cost accounting work with the accounting department, this company makes use of an NCR 315 CRAM system. This includes a 10K core memory, four CRAM units, and equipment for reading and punching cards and paper tapes. This equipment took over for prior punched card equipment for the same applications. In addition to the manufacturing control operations, some forecasting and projecting of requirements for individual components and generation of appropriate manufacturing orders is carried on. By using random order in file organization, the access time in general is about 235 milliseconds, and in the special case where information happens to be on the card on capstan, the access time is about 45 milliseconds. At the present time approximately 88% of the total capacity of the file is in use and operating in good order. All records in the system are stored in a single, random-ordered file. All accesses to the file are made for the operating programs by a common executive program. Since the CRAM cartridges are removable, duplicate copies have been made for use during debugging operations to prevent danger of destroying the good files of information by programmer error. Improvements which Marvin would like to see in equipment include faster access time and a write lock-out on the file under program control, but only if the improvements can be made with complete compatibility between the improved devices and the current ones to avoid reprogramming difficulties.

*Marvin Eyster, Manager of Data Processing for Woodward Governor Company of Rockford, Illinois, described their use of data processing and storage equipment (summarized).

QUESTIONS AND COMMENTS FROM THE PANEL*

Bob Franklin pleaded for techniques to extract information from a file based on requirements not defined when the file is originally structured, and he postulated that this was a hardware problem. Irving L. Wieselman† answered that he considered this a software problem. The hardware manufacturer is concerned with building better hardware. More capacity, faster transfer rates, and lower costs are his goal. Bob Graham supported the position that this is a software problem, and pleaded for general-purpose programs which are hardware-independent. He further strengthened his belief that all the special wrinkles should be in the software and that the hardware should be as general purpose as possible.

Franklin's rebuttal brought out his disbelief that the electromechanics of mass files are not strongly influenced by data structures (e.g., the rigid formats characteristic of most present machines). Don Sampson‡ stated his belief that manufacturers build what sells, and overly specialized equipment does not sell. Graham pointed up the difficulty when the difference between the logical structure of the data and the physical structure of its representation are confused. He contended that the two are completely independent but are all too often made dependent and that this makes a special-purpose device. One of the objectives of the MULTICS system is to keep the users' logical data structure completely divorced from the physical representation. This facilitates the updating of hardware.

Bob Franklin displayed a RCA RACE card and pointed up the wear problems with it. There was no apparent wear on the magnetic recording surface. The wear showed up in the ejection area and on the picker studs.

Wieselman noted that the panel represents users which require exceedingly large data bases, and he requested comment on the value of a very small file used for a small data base for a small business. His question concerned the usefulness to a small business of privately-owned EDP vs public utility EDP.

Farrand noted that there are a very, very large number of disk packs around and the ability to get the data and store it on these seems to be a very well accepted practice.

Someone not identified from the floor suggested that address by content is the answer to Bob Franklin's query. Bob agreed that updating of their file is an address by content operation but that the method is exceedingly inefficient for they must "page" the data into core to address by content, and for the 1.2 billion characters in their system, this takes 20 hours. Additional data were revealed at this point concerning the Chrysler file. It has 6 million vehicle records and each record is quite large. This needs to be a large record for, in just the Plymouth division, there are 13 million buildable models. He then pleaded that there must be some way of looking right into the information within a random access file and saying, "This is the way it should be," and out it comes—even though it was not set up to have this request made in its original formatting. Bill Farrand noted that there is a dearth of algorithms concerned with adaptive storage but that address and extract by certain tags is quite another story and is quite practical in some cases. He further commented that at present this is circuitry-wise feasible in small files and is truly useful in huge ones.

*This is a summary of a tape of the panel's actual comments.

†Vice President of Product Management, Data Products Corporation.

‡Director of Engineering, Mass Memory Operations, Control Data Corporation.

QUESTIONS AND COMMENTS FROM THE AUDIENCE AND PANEL

Byron Smith* directed a question to Alan F. Shugart† concerning IBM's interest in devices of the CRAM and RACE character and made specific reference to the 2314, sometimes called a wall-to-wall disk file. He asked a specific question: "Is the access time of the card file a basic detriment, and is that why you are not using it?" Shugart answered that IBM is in the card file business with their Data Cell Drive. Shugart acknowledged the competition between the 2314 and the 2321 and stated that he thought there was a reasonable chance that the 2314 might replace the large disk files but not the Data Cell. Al then showed a movie of the first installation of a 2321 Data Cell Drive outside of IBM, a film on the installation of their machine at Allstate Insurance Company. This is used in conjunction with a Model 40 system. This installation has two 2311 disk packs, a 2321 Data Cell Drive, together with a 2841 control unit. These films showed the Data Cell Drive being installed and running. The Data Cell Drive was shown in operation and an operator was shown removing cells from the drive unit and reinserting same. An unidentified voice from the audience asked the question concerning expected life in terms of number of passes of a card in tape strip files. Shugart stated that they do not track the life of their cards in numbers of passes but that their specifications call for 4 to 5 million picks. He emphasized that this is not

*Analex.

†Technical Manager, Random Access Memory Programs, IBM Corporation.

necessarily addresses because once you pick a strip, you may have several addresses on that same strip. And he emphasized that 4 or 5 million picks per data cell should be the life of that cell but that revolution life around the capstan is not expected to be a limiting factor. Marvin Eyster commented that their installation seems to get about 200,000 revolutions on the drum per CRAM card but that they seldom revolve a card more than one or two revolutions and card life is limited by the number of times they drop it, and they are certainly experiencing in excess of 10,000 drops per card. However, they do not keep log sheets in this manner on the data.

Bob Franklin noted that the card stock on the RACE units is manufactured by 3M and that they have made some significant improvements in the material. He further noted that oxide life was not a determinant on the card life. William J. Broderick* commented on Byron Smith's point concerning the multitude of seeks available with a 2314 disk complex when used in parallel seeking, and noted that this could effectively reduce access time. He then asked the questions: "What does overlap access mean to us? What is the value of parallel read/write? What would it mean for the Chrysler RACE system to have multiple read/write?" He further asked Franklin if they have parallel read/write control. Bob answered by stating that they are not able to operate all four units simultaneously.

Frank J. Lohan† mentioned that there are several pieces of equipment on the market with simultaneous read and write but that the File Drum is the only current piece of equipment on the market in which several independent actions can be taking place in a given band of data simultaneously. Don Dittberner‡ expressed concern over the panel's apparent emphasis on present systems and lack of emphasis on future systems. He passed out a query concerning what is detrimental to having more control associated directly with disks so that the experience with serial associative search mechanisms could be utilized. A second query concerned the feasibility of a single unit incorporating a combination of optical recording and magnetic recording. Also why hadn't the question of packing density been brought up? He tendered the belief that 3,000 to 5,000 bits per lineal inch isn't going to be an unusual density in the near future. Bob Franklin remarked that the four users on the panel had pointed out rather graphically their problem areas and what they will demand in the near future. He emphasized that multiple access, high speed, high reliability, and low cost are certainly in our immediate future. Bill Farrand commented that the point brought out by Don Dittberner concerning the possibility of making the file a most autonomous entity by having closely associated with it certain logical editing and manipulative functions is a very good one for some people but is in definite deference to the attitude portrayed by Bob Graham. It was further mentioned that the 6600 and 6800 systems were structured in just this manner with peripheral control.

Bill Farrand then pointed up the controls on the future as being economic value and technical reliability. Files can be structured so that they do not take up main frame master control program steps. But are these added features worth the price? Next, reference was made to the 10^{12}-bit file ordered by LRL. As the large present disk files are about 1 billion bits in max size and larger than 4 feet in case length, using the electromechanics from these to form a 10^{12} file takes 1,000 units or over 4,000 feet. This demand for nearly a lineal mile of floor space might make the unit unfeasible.

The exchanges which we hoped for between the panel and the audience were thought to temper the question, "What will be available in the future?" by answering questions beginning, "My systems needs are . . ." by: "The following technology will answer this with the following characteristics at the following cost. Are you willing to pay for it? If so, we'll build it." The questions are, where are we going, what are we going to have when we get there, and what do we really need if we're to get there. The only way to get hardware in use is to have specifications on equipment which can be met by the manufacturers and which are useful to a large number of users.

Alan Spahr* asked for Al Shugart's opinion on the place of fixed vs removable media in disk files, and Al answered by stating that it is his opinion that a replaceable file is here to stay. He reinforced his opinion with considerations of reliability, noting that you can move the storage media from one machine to another if the machine breaks down. Don Sampson made the distinction between removable disk files and large fixed disk files by calling the fixed files large precise disk files. He further expressed the belief that there is a definite reliability edge in favor of the precise disk file. His belief is that the disk file business is going to settle down and that the bit density and cost per bit will decrease drastically on precise disk files. He, however, emphasized that though the race is on between the two types of units, neither will die for want of customers. Bill Broderick requested some discussion concerning

*Product Planning, General Electric Company.
†Disc File Product Manager, Bryant Computer Products.
‡Center for Management Technology.

*Analex.

the reliability difference between flexible media and media in storage subsystems. He specifically inquired if there was anything inherently better in flexible media other than price over fixed or removable disk units. Al Shugart answered by stating that the 2321 was developed strictly on a cost-per-bit basis. Bill Farrand then made some comments concerning the excellent volumetric efficiency of a strip file at least in principle. He also alluded to the value of the removable media feature usually built into strip files. Bill Broderick wondered if the use price to the customer were the same for a mass storage device that did not employ flexible media but was removable at the same price per character, would not the operational characteristics pay for the flexible media? Irv Wieselman summed up by asking if Broderick's real question was, "Are disk files more reliable than strip files?" Al Shugart answered that in his opinion disk files are more reliable than strip files. Al reinforced this opinion by stating that we have a large amount of experience with fixed media files and that this position may change when we have comparable experience with flexible media files.

Farrand commented that the removable media files would always be plagued with dust and dirt to a greater extent than fixed media files. An unidentified person from the audience then questioned whether removable media and fixed media files could be compared, since, in the hierarchy of stores as he saw it, they were on different levels, and his point was we were comparing apples with oranges.

Bill Farrand then commented that our experience in strip files isn't as sparse as one might think for nearly a thousand such files exist today. Bill Broderick then showed a slide, which gave cost per character (normalized to six-bit characters) vs record accesses per second for various types of files (Fig. 1). The prices depicted are from total subsystem on-line. The systems have been categorized by function: A, being core extensions; B, recall processing; and, C, very large capacity reference file units. The dotted outlined area A is "announced future large solid-state storage." The solid outlined area A consists of drums and fixed head disk files. The pricing data predictions are obtained from AFIPS and IFIPS publications.

SUMMARY*

What light have we thrown on the equipment and systems we have with respect to ordering of data, interrecord gaps, deskewing, and structure? Is the choice proper—as often used—of taking parallel data, turning the corner with it, and streaming? Is it an error to not stream the data more since, on the average, this is such an efficient operation for long records? We have attempted to state that if customers need it and technology can produce it, "it will be," if it is economically feasible.

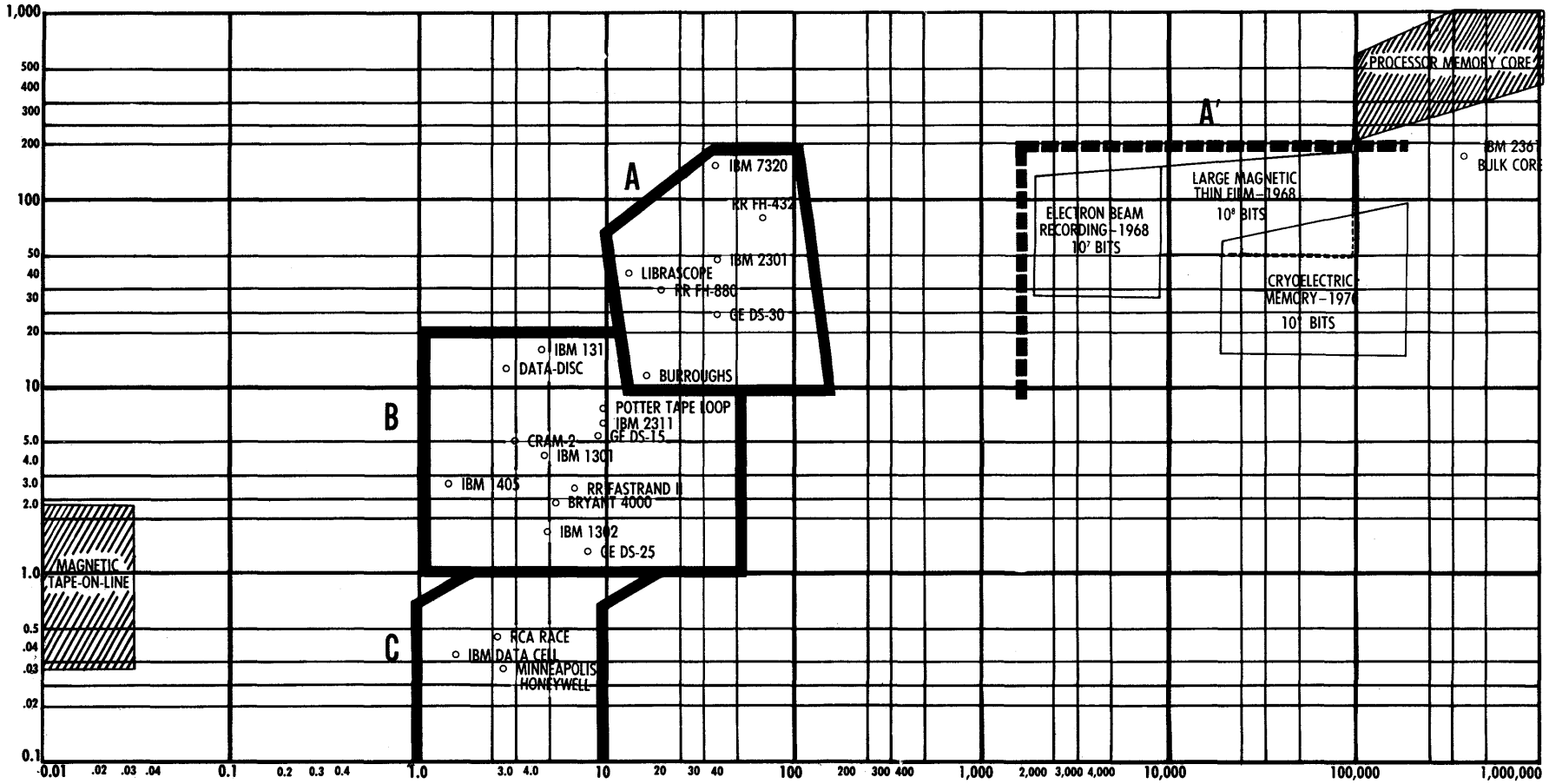
The view taken of the future is what will be—not what might be. If the needs of the industry are distinct, individual, all specifically different, then the economic feasibility is to be questioned. But if the need is universal and technically feasible, then the need will be answered by equipment. This is only prevented when poorly designed equipment is installed in systems and their failure to work costs so much in time and money that the situation is leapfrogged.

Many believe that the perfect organization is similar to the way a random access core is organized. This is often not an optimum organization due to the difficulties of locating the data by programming. The block organization of data in many electromechanical memories is of significance in many installations as is the sequential flow of data in a record. Electromechanical mass memories of many kinds and forms exist with excellent up-times and very low installation cost.

The questions are: "What is really wanted? What is really feasible? How much does it cost? Does that make it unfeasible?" Technology is leaping forward. One characteristic brought out by the panel members is that software is such a large investment that it is absolutely necessary that any future system be able to work—no matter how crudely, no matter how poorly, no matter how inefficiently—with the data in the available form using the available bases.

*By the session chairman (summarized).

MILS PER CHARACTER OF ON-LINE STORAGE
PRICE PER CHARACTER (NORMALIZED TO 6 BITS): Purchase price of controller plus first full storage unit, divided by formatted capacity in 6-bit characters.



ACCESSES PER SECOND: Data-dependent (non-simultaneous) access rate determined by average positioning time (if any) plus 1/2 latency plus full latency (minimum to initiate an update). This time in milliseconds is then divided into 1.0 second to determine a "minimum access rate." potential of the subsystem.

William J. Broderick
General Electric Company
Phoenix, Arizona
1965

Figure I

A Panel Discussion

Promising avenues for computer research

REX RICE, *Fairchild Semiconductor, Palo Alto, California*
KEITH UNCAPHER, *The RAND Corporation, Santa Monica, California*
TOM STEEL, *System Development Corporation, Santa Monica, California*
L. C. HOBBS, *Hobbs Associates, Corona del Mar, California*

PART I

Part I of the session is a presentation by the four panelists. We would like to give you our views on computing as it can be in the year 1970. In Part II, we invite the audience to comment. We'll be very pleased to have you air controversial views. We on the panel agree we do not know all the answers. First, we want to isolate promising avenues of research related to the potential payoff, secondly, we want to emphasize either existing research that should be encouraged or research that is not now active and is needed.

FUTURE COMPONENTS
Rex Rice

SYSTEMS GENERATION

Figure 1 is a simplified presentation of two generations of research, development, and manufacturing cycles of data processing systems. We are now in the production phase of the integrated circuit generation. A large amount of research effort is presently being expended to develop large batch fabricated arrays of components. As shown, we are now in the early re-

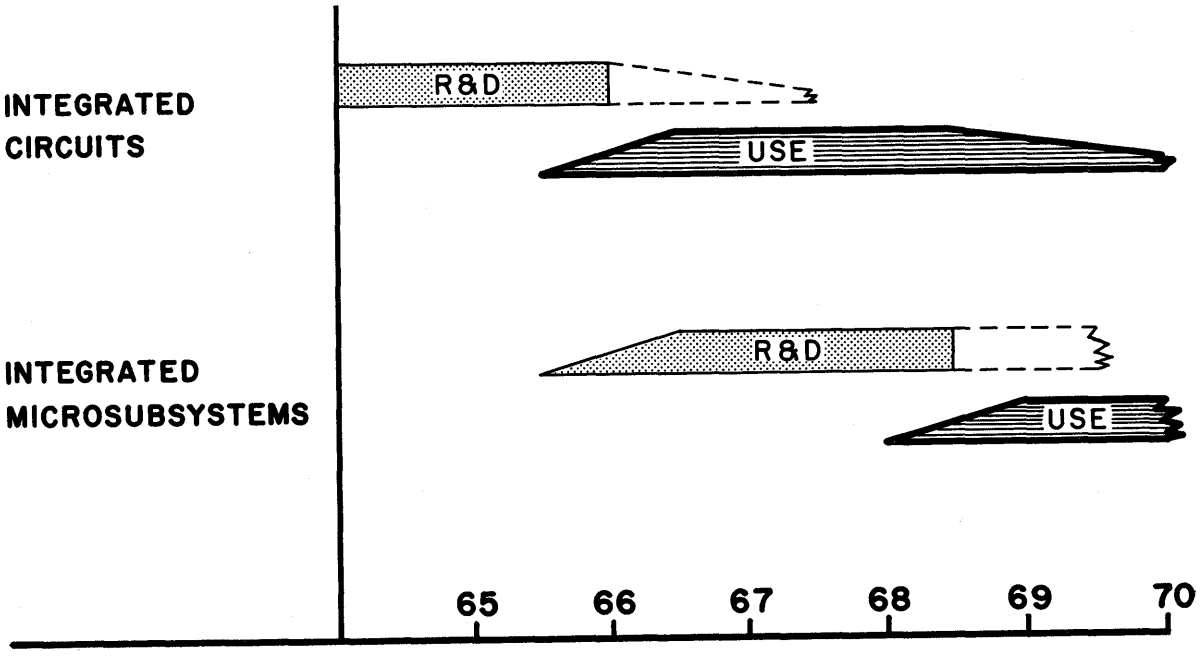


Figure 1.

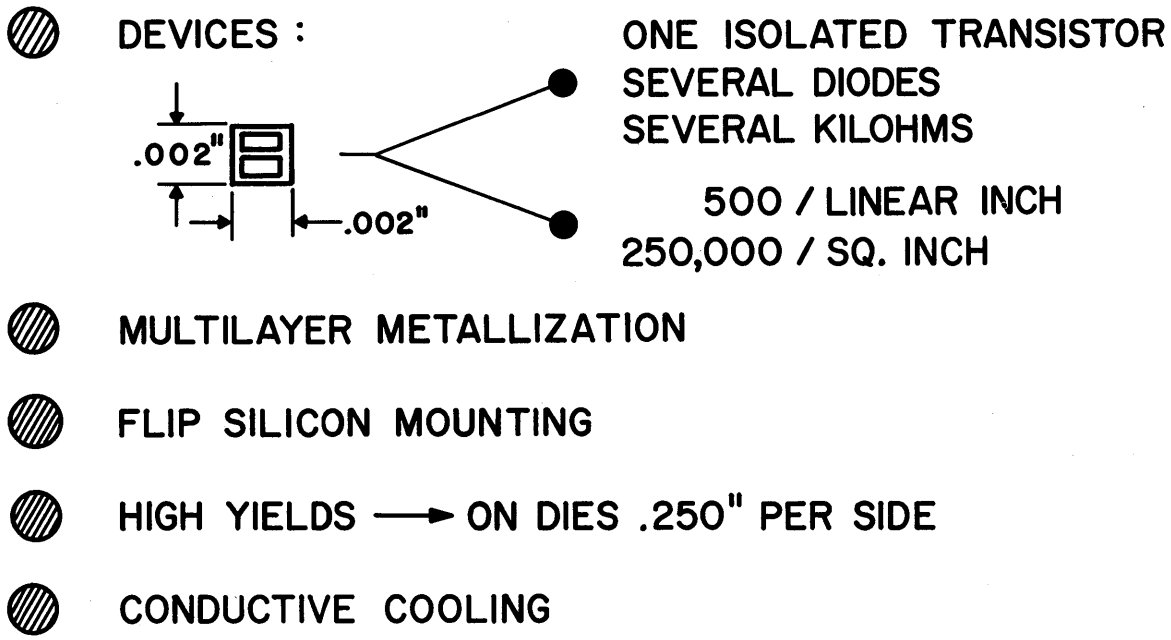


Figure 2.

search and development phases of the "Microsubsystem" generation and can anticipate such systems as being in operational use by 1969 or 1970. The balance of the discussion herein is concerned with the next generation and not our present "integrated circuit" generation.

SILICON TECHNOLOGY AND ECONOMICS

The rapid advances in integrated circuit technology and manufacturing capabilities have already outdistanced the predictions made several years ago. As illustrated by Fig. 2, we are now making devices which are only 2 mils on a side. Within this area one can obtain a transistor, several diodes, or several kilohms of resistance. This density represents 500 components in a linear inch. It also provides the amazing number of 250,000 components in a square inch.

This component density when considered alone, has no meaning. It must be coupled with other processes presently being developed in several research laboratories. First, multilayer metalization must be provided over the top of these devices in order to interconnect them into a useful microsubsystem. Secondly, a desirable but not mandatory process is the flip (inverted) mounting of silicon. Thirdly, research and manufacturing development is being vigorously pursued to obtain large devices at high yields. One can anticipate that a substantial improvement in yield will be obtained in the near future. We can then seriously consider a high-yield device which has a dimension of a quarter of an inch on a side. Finally, techniques are already available which allow heat developed in the silicon to be conducted out and dissipated elsewhere (see G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics,* Apr. 1965).

The economics of micro-systems resulting from this research can be summarized as follows:

Circuit Assumptions:
65,000 devices per quarter sq. inch die
20 devices per circuit
storage bit
flip-flop
gate, etc.
3,250 circuits per die
Cost Assumptions:
$50 silicon
10 package
40 testing
$100 for 3,250 circuits
Less than 3c *per circuit.*

There are approximately 65,000 devices in one quarter-inch die. Let us assume the average of 20 devices for each circuit. Each circuit could represent a storage bit, a flip-flop, a multiple input gate, etc. This provides 3,250 circuits on a die. For high-volume production on a few types of dies, one may assume silicon costs approximating $50, packaging $10 and testing of completed components $40. This totals approximately $100

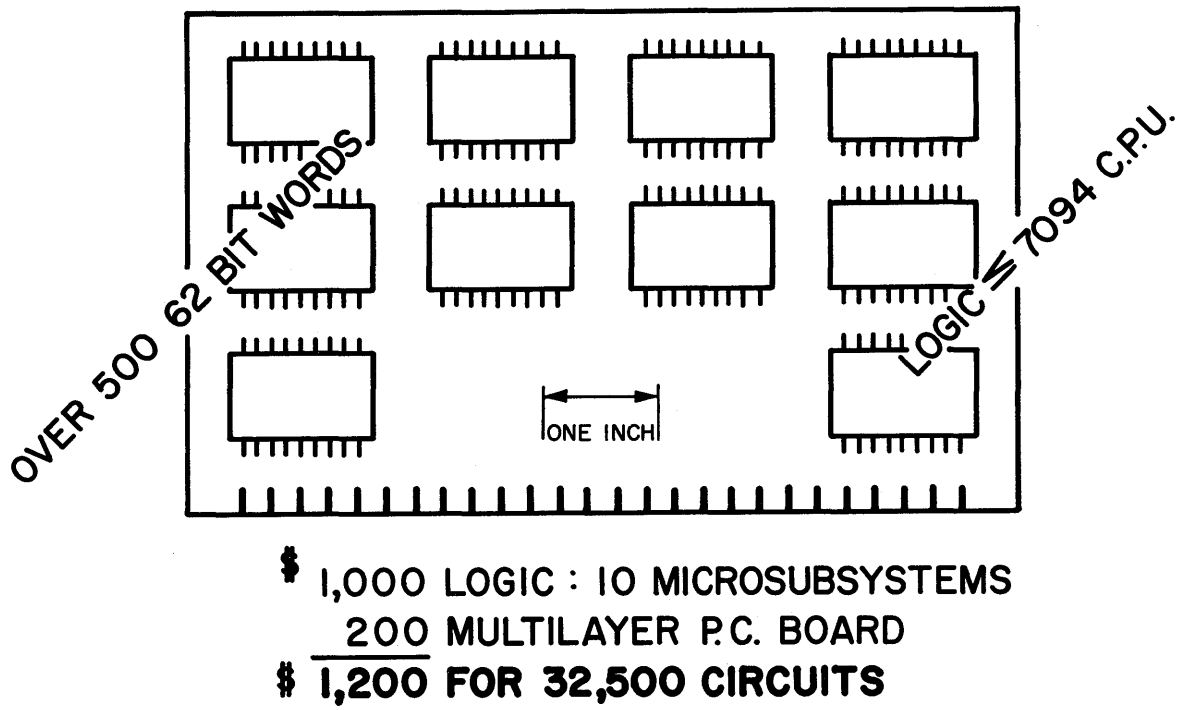


Figure 3.

for 3,250 circuits. It is technically feasible to achieve high-speed integrated microfunctions with circuits approximating 3 cents per circuit by 1970. It should be emphasized this will require some restraint on the part of circuit and logic designers so we can develop standard microsubsystems to achieve high-volume production on each type.

In Fig. 3 a hypothetical system function is shown. This function contains 10 microsubsystems mounted on a multilayer printed circuit board. Assuming a $200 cost for the multilayer circuit board, the total function cost would be $1200 for 32,500 circuits. For example, this single card could represent 500 words of 62 bits of high-speed storage. Alternatively, it represents more logic than is found in 7094 central process unit and its instruction controls.

BULK MEMORY ECONOMICS

Bulk memory will undoubtedly take many forms by 1970. For the purpose of this discussion, it is only necessary to illustrate one bulk memory with a reasonable cost so desk-side, real-time, stand-alone computing may be performed. A simple extrapolation on disc pack type memories now in development, as discussed at the Batch Fabrication Conference, follows:

Development now:
Discs—12.5 $\times$ 10^6 characters on-line
20 accesses per second
10,000 characters per second flow
$12,500 drive
500 pack
$13,000

Research now:
BORAM—4 $\times$ 10^6 characters on-line
1 microsecond access
1.5 $\times$ 10^6 characters per second flow
$???? read-write unit
500 pack
$????

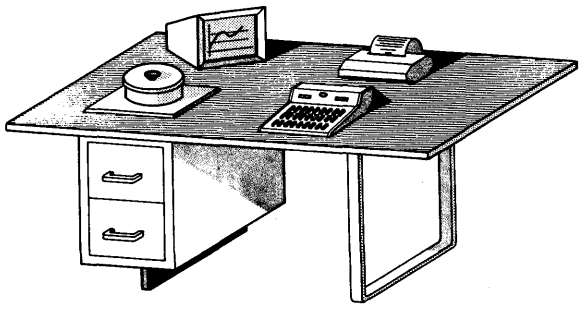
It was stated that for $13,000 the system would provide 12,500,000 characters on-line. At 20 accesses per second and 500 characters for an average access, this system provides a data flow rate of 10,000 characters per second. This flow rate will be adequate for some purposes; however, we should anticipate that with proper research higher flow rates will be achieved.

At the Batch Fabrication Conference, BORAM was described. BORAM is a block oriented random access memory of an electronic nature. At the present time a research program is funded to do work on this system. The specifications call for 4 million characters on-line

in a removable pack which costs no more than $500.00. The specifications also call for 1-microsecond access time and a 1,500,000 plus character per second flow rate. Systems designers certainly hope that this type of memory will be developed. The two memories discussed here illustrate what can surely be expected and what we may hope to achieve by 1970 with proper research.

Hypothetical System

A hypothetical system, utilizing components previously described, is presented in Fig. 4. Here we see a desk which contains the logic and disc pack storage drive in the drawers on the left. On top of the desk is



CPU (≈7094)	$ 1,200.	
FAST STORAGE	8,500.	21,500
BULK STORAGE	13,000.	
KEYBOARD	100.	750
PRINTER	400.	
DISPLAY	250.	
TOTAL	$23,450.	

Figure 4.

an alphanumeric keyboard and a hard-copy line printer. Also included is a graphic display which is capable of producing analog as well as alphanumeric displays. This system can do both commercial and scientific problems ranging from small to medium large. This local data processor will also tie into a telecommunications network so library routines and data may be swapped. We envision the system doing its own computation for all applications except for a few types of large problems. This hypothetical system is not proposed to be used for bulk file processing such as social security data processing, large insurance company files, accounting information for large businesses, etc. It is presumed large file processing will be handled in a batched manner in large computational centers.

The cost of such a system is made up of the following elements: $1,200 is for central process unit logic; $8,500 to provide 4,000 words of very-high-speed storage. This Memory requires 80 of the microsubsystems previously described mounted on two cards. It provides access time in the order of 100 nanoseconds. $13,000 is included for disc drive and one on-line disc pack. The interface with a human operator is through a keyboard costing $100. A medium-speed line printer costing $400 and a display assumed to cost $250 are assumed. The proposed display will make high usage of special microsubsystems to achieve low cost. The total cost for this system is estimated to be $23,450.

HARDWARE CONCLUSIONS

The conclusions one may draw relative to the hardware portion of such a system are:

- Hardware costs insignificant ($23,450 is less than the salary of one professional by 1970).
- Need high volume.
- Need effective problem solving tools (language).
- Need new software (systems).
- Need better man-machine communication (equipment).

First, $23,450 is apt to be less than the salary of one professional person by 1970, even excluding overhead. The computation facility such a system will provide, will make the hardware cost insignificant. To a first order approximation, one may assume hardware costs are zero and then concentrate on the *use* of such a system *and how to obtain software.* To achieve this cost, this type of system must be in high-volume production. In order to reach such a production, two items must become paramount. First, this system must give the problem originator effective computation aids, in his own natural language and at his desk, in order to boost the demand. Second, we must develop an adequate software-hardware combination to provide real computational assistance. If this means lavish use of low-cost hardware to make software development simple, then we should certainly start thinking in this frame of reference. Having postulated this technically achievable objective by 1970, one may ask these questions, "What will it do to help the problem originator? How can we possibly develop the programming necessary to make it run?"

THE MAN-MACHINE INTERFACE

Keith Uncapher

The purpose of this panel is to encourage creative use of $.02 logic elements, to stimulate rapid growth of computer technology, and to provide low-cost computational assistance to anyone who can use it.

This paper discusses assistance to a particular class

of computer users—namely, the practicing 1,000,000 engineers in the United States. Though not necessarily the most important class of users, it is one which many of us know a great deal about. Throughout this paper, "the user" or "the casual user" generally means the noncomputer-specialist practicing engineer. One promising avenue of research is to consider how to give engineers (and others) low-cost, on-line, on-demand, computational assistance.

First, let's examine the *practicing engineer's* viewpoint of the computer equipment and systems provided for him during the last 16 years—that is, the viewpoint of the casual user, the noncomputer-specialist, nonprogrammer engineer interested only in the tools available to help him solve his problems.

In 1949, the engineer could be on-line to an analog computer: he could sit at a console and interact with the machine. The graphic input/output, dials, meters, and gadgets helped him formulate and solve his problem. More important, the engineer problem-solver at the console of an analog computer was part of a very tight feedback loop including himself, his problem, and the machine.

These were all advantages for the engineer, but the situation had negative aspects too. The user saw too much unfamiliar hardware, which often impeded the solution of his problem. He could easily touch +100 volts while altering wires on a plugboard. Further, he had to know the layout of plugboards and the nature of the instruments, bells, and whistles which surrounded him. To be an effective user, he even had to understand the guts of the analog computer—drift rates, ground loops, how an operational amplifier could integrate, etc. —because a direct relationship often existed between the problem being solved and the stability of the overall analog system. That is, the stability of the computer could greatly alter the accuracy of the solution.

Thus, the user was forced to pay close attention to the system itself rather than to just solving his problem. Figure 5 depicts, in a gross sense, the user's view

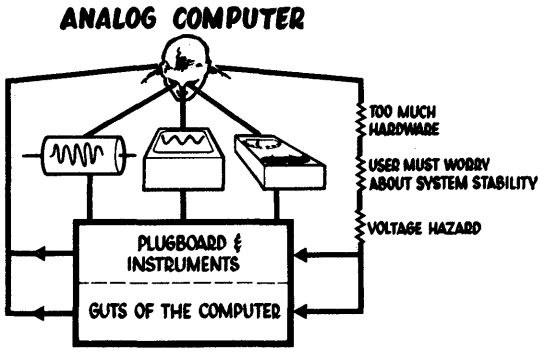


Figure 5.

of an analog computer. Note that while he faced some "impedances" (i.e., too much hardware, system stability problems, etc.), he was in fact totally and continually coupled into the feedback loop including the machine and his problem. Graphic aids allowed him to continuously monitor the behavior of his problem. But in spite of all the positive advantages of a well-designed analog computer, few engineers used them. Many engineers in the late forties apparently were unwilling to learn enough about analog computers and computing to solve their problems.

In the early fifties, the advent of 701's and 1103's tended to push the analog computer into the background. Many installations attempted to give each user a turn at the console of a 701. For the skilled programmer, the 701 was indeed a blessing—but not for the casual user interested only in solving a problem. Fig. 6

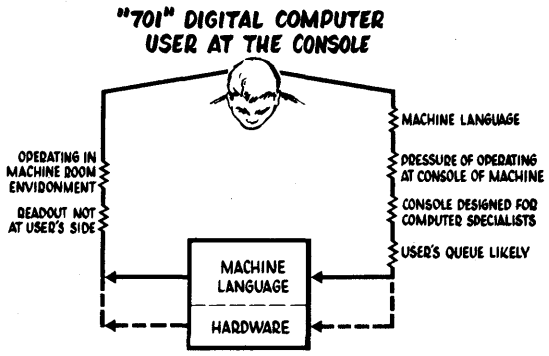


Figure 6.

depicts his probable reaction. Such an operation required the user to deal with machine language and a very high-priced machine; therefore, time at the console was precious and costly, creating pressure on the user. Moreover, he often had to wait in a long queue. *To effectively use the machine,* he again had to know something about the hardware with which he was dealing. The fact that the word length was 36 bits, the read-around ratio, the expected reliability of tapes, etc., were factors which determined the kinds of problems that the system could solve effectively and reliably. Again, much of this distracted from the major reason for being at the console—namely, the solution of a problem. Operating at the console of a 701 was extremely attractive to the computer specialist or to the skilled programmer, but not to the casual user.

Although many engineers shunned early digital computers, those who did use them clamored for more and more console time and computing. The need to serve more users stimulated the development of batch processing.

Batch processing brought higher-level machine languages which both benefited and hindered the casual

user. Again, the long queue is present, along with new "impedances," such as long turn-around time and occasionally dealing with a nasty operator. The user needed to understand less of the machine's internal hardware but, to be effective, still had to know the word length and many other machine parameters. He had to memorize books of special rules for higher-level languages. Again, for the casual user (see Fig. 7), we

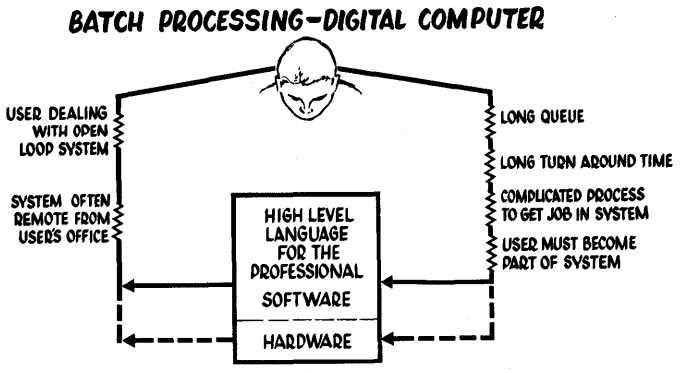


Figure 7.

surely didn't do very well.

During the past three years, the rediscovery of being on-line has given new hope to the casual user. Once again he can work from a console, as in the former analog and 701 days. Although the emerging on-line, time-shared systems are exciting, promising, and productive, few cater to the needs of the casual user. Many are retrofits of languages designed for relatively skilled programmers. Most do buffer the user from much of the hardware and some of the software of the machine with which he is dealing, but many still require him to know a great deal about both hardware and software (Fig. 8). For the computer specialist and

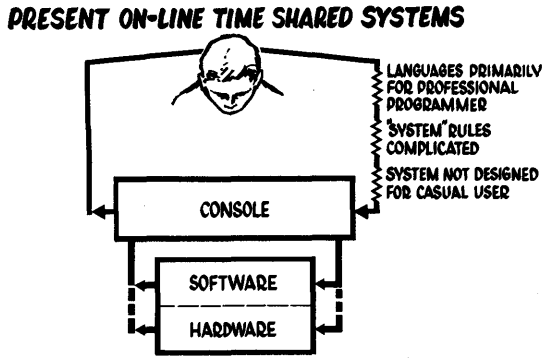


Figure 8.

the skilled programmer, it's no problem. For the casual user, it is a level which many will never bother to attain. Just being on-line is not enough to attract most casual users.

The rest of this talk defines promising avenues of research which, hopefully, will attract those currently unwilling to use computers. One such avenue is to design systems tailored to the needs of the *casual user* rather than computer professionals. One of the greatest potential markets for "free" hardware is low-cost, casual-user-oriented systems which can solve a variety of problems for many classes of users. Present computer technology is probably sufficiently mature to be able to build a few such systems. In considering their design, keep in mind that casual-user-oriented systems should:

1. Allow bilateral conversation in a language of the user's choice.
2. Require little or no training.
3. Provide the user with adequate and constant feedback regarding the formulation and solution of his problem.
4. Be more helpful to the user than such alternatives as FORTRAN, slide rule, desk calculator, service programmer, computing aide, etc.
5. Not be fragile or inflexible in the hands of the user but, rather, resilient and responsive.
6. Allow the user to interact via a "personal" console in his office, his home, or both—and probably with graphic capability.
7. Must allow user-system interaction data to be stored on-line with nearly instant access.
8. Assist the user in the formulation as well as the solution of his problem, and permit him to concentrate on that solution rather than system mechanics.
9. Cost so little that long periods of console inactivity are acceptable.
10. As much as possible, "seal off" the user from the messy details of the computer hardware and software.

In a total sense, the user should find the system a *"helpful assistant."* Figure 9 depicts a user's view of

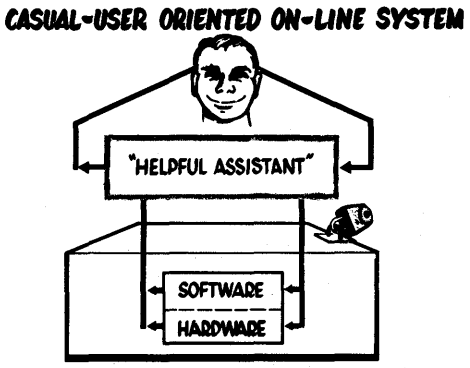


Figure 9.

such a system. It shows that, although some of the "impedances" of other systems are not eradicable, at least the messy technical aspects of the software and hardware should and can be masked: *The user should see only what helps him solve his problem.*

Let's consider other promising areas of research. Certainly it is important for the casual user to be on-line in the familiar surroundings of his own office. All of us have experienced the trauma of trying to learn to use a computer in the presence of skilled, professional programmers. The novice is more at ease—and thus more likely to not give up—in the privacy of his office, learning via a "friendly" console, without an expert looking over his shoulder. Therefore we must provide the user a "personal," *low-cost,* reliable console in his office or home. It must cost so little that it becomes another piece of office furniture—not removed from his office if he goes on vacation. The user should be able to justify the console and his use of the system on the basis of a few hours of use per week.

A typewriter is initially an adequate input device for many users. Eventually, sophisticated languages will allow the casual user to solve more difficult problems, creating a demand for graphical input/output devices and graphically oriented languages. To be helpful, the level of interaction through graphics must be so intimate that many users will need a sophisticated, high-speed (expensive by today's standards) graphic capability. Another promising avenue of research is to build low-cost, highly interactive displays—say less than $20,000 per console. Perhaps the broad application of digital techniques in the design of displays could lower the cost and improve performance.

Certainly "free" hardware will permit significant processor capability on the back of a typewriter or display. The equivalent of a CDC 3200 on the back of a typewriter will be possible by 1974. If research provides what the casual user really needs, he will probably not be satisfied with just the ability to communicate directly with a modest computer. The engineer at the console will probably want to communicate with his friends at other consoles, with data bases, and with other larger and faster machines. Thus, low-cost, cheap, reliable communications are necessary to connect stations, to transmit anywhere quickly and cheaply—in fact, it is required to further the use of existing on-line systems. To put it another way, present on-line system use is severely hampered by the lack of low-cost suitable communications. Present common carriers employ slow switching techniques. Moreover, if the user wants to buy some time, he must buy at least three minutes' worth, which is enough to transmit 126,000 bits (over a slow-data-rate line), though he may want to pay for and transmit only 3 bits. We can do better.

Skilled digital computer specialists consider a promising area of research to be the design of an all-digital, nationwide communication system required to connect users, consoles, computers, and data bases in any way required—cheaply and quickly. The concept is sensible, and such a network is a prerequisite to full utilization of on-line, time-shared computers.

In summary, there is plenty to get excited about. Every engineer, programmer, businessman and student *can* be on-line. However, we should move slowly until we are sure we know *how* to design and implement systems for the casual user. We can learn his needs and requirements, but not easily. Once we do, the technical progress of the past 15 years will permit us to design and build new systems far surpassing our accomplishments to date.

PROGRAMMING RESEARCH
Tom Steel

The promise of essentially "free" hardware presents an important challenge to those who produce software. The point has been made that the cost of hardware will become insignificant in terms of the total systems cost.

The cost of programming, by today's standards, is enormous in contrast to "$ .02 logic." In fact, it already appears to be true of many big new systems that the cost of preparing the software perhaps equals that of the hardware (at today's relatively expensive hardware prices). If we are to make this "helpful assistant" work, it will be necessary to get repetitious software costs down to some reasonable figure. It is therefore important that we of the programming community forget many of the constraints that we have placed on ourselves for one reason or another. For example, in today's batch processing environment many people spend a great deal of time and effort attempting to make their programs "short" in terms of the storage space they require. All they *need* to do is make them short enough to fit into the high-speed memory of the machine.

One of the reasons that computer systems do not now have some of the features that a "helpful assistant" should have is that (in terms of our present view of the way machines operate) these features cost too much in machine time to run. But suppose machine time itself were negligible in cost. We could then parallel operations sufficiently to get real time down to a reasonable point, and so, if overhead didn't cost anything, who would care if it were 20%, or 50%, or 95% of total running time—if it served a useful pur-

pose. I think a reasonable guess might be that it would be necessary to spend about 50% of the machine's time and capacity on overhead functions. That may sound like a lot, but the situation today is almost that bad in many cases.

What can be done to provide the forgiving, friendly, interactive, "helpful assistant?" What are the sorts of things we need? One thing we don't need is the FORTRAN compiler or the PL/I compiler. We must have some much simpler, problem-oriented language—more like the kinds of languages, notations, etc., that the user ordinarily employs. If these users are to be "a million engineers," this language looks something like mathematics, but you will also find a great deal about format, loops, and other things that the casual user doesn't understand, won't bother to find out about, and shouldn't be asked to. It is necessary to eliminate this kind of thing, and it is important to recognize that much of this elimination can be done today. We haven't done so yet, and there are some good reasons—mainly related to cost. We haven't attempted to integrate the various techniques that are available for solving problems. We do not have systems that have a simple, quick calculation mechanism coupled with a data file retrieval system. We do not have a formula manipulation program together with something like an analytic differentiator where the interfaces are compatible. But there is no really good reason why, with careful design and thinking ahead, it isn't possible to put these things together into a comprehensive package.

We would need a set of basic mathematical functions, some sort of generalized array processor, facilities for the generation of subroutines that would compute special functions, and algorithms for going from a series definition of a function to a rational function which could have the coefficients optimized. Further, it's perfectly feasible to go from the last-named item to the actual generation of the machine code to compute the function given the arguments.

It is then necessary to have all of this automatic. You can't ask that the whole proceeding come to a halt while the user puts in a few cards that perform links between one set of programs and another. The interfaces between the subsystems have to be designed smoothly.

In addition to this, one needs some kind of reasonable display and printing program. One should be able to describe formats in a very simple, probably pictorial way. Then, of course, one needs some kind of monitoring system on top of all this to tie it together. One of the things that engineers would like to have available to them is some kind of reasonable fact retrieval system so that they don't have to rush off to the library to get the handbook they didn't happen to have on their desks. This ought to be available through the console. One should be able to approach this retrieval system through the use of natural language questions; so the system should provide, additionally, the facility for natural language inquiry. Again, we know pretty much how to do this today. There are some ambiguity problems in handling natural language inquiry, but most of the ambiguity problems can be easily resolved provided a dialogue exists. If there is a possible ambiguity, the system comes back and says, "Did you mean this or that?" and it doesn't do it in a very demanding way. Undemandingness is part of this matter of being "forgiving." If you try to intimidate the user by harsh, cryptic, or telegraphic messages, he is not going to be very happy about it. He may not know why this bothers him, but it will, as a good deal of evidence shows.

What does all this take in terms of today's programming? A rough guess is about 500,000 instructions of the sort that are in the machines we use today. That is a lot of programs. I looked at the question of how much effort it would be to design, implement, and check out such a system in terms of the programming alone. It is a job that could quite likely be done in about 125 man-years of effort and could probably be done right if you gave it five years and didn't try to meet some irrelevant deadline. That may seem like a huge effort, but it is quite clearly less than the effort that IBM is currently putting into the production of 360 software, and it would be much more useful.

So far I have discussed what we need and how much we have at present. What is required in the way of research? Most of the things I have discussed already exist; it is a matter of putting them together. It is at these interfaces between the programming subsystems that we need the research. Also, we need a great deal of study at the man-machine interface. We need to human engineer the programs—not just the console. It may be important whether a console display is angled at 30° or 60°, but it is also important how many characters there are in a message and how many different characters are available.

There is much work here that needs to be done, and so far it is not getting adequate attention, although I begin to detect signs that people are recognizing this need and beginning to think about studying it. Given five years, we ought to have most of the answers.

POSTULATED SYSTEMS

L. C. Hobbs

The title of this portion of the panel is somewhat misleading since I will be discussing postulated improvements in systems rather than postulated systems. Emphasis will be placed in promising avenues of research and development from the systems standpoint.

Two promising avenues that offer significant system improvements—programming research and man-machine interaction—have already been covered by Tom Steel and Keith Uncapher. The low-cost storage and logic elements resulting from batch fabrication described by Rex Rice are certainly essential to both of these. Low-cost batch-fabricated memories will make it feasible to store the large program libraries called for by Tom Steel's postulated programming system. Small low-cost local stores will make it possible to store frequently used programs or program segments locally, thus requiring relatively infrequent access to a large centralized program storage. In this regard I think programmers must be very careful to assure that new software for systems implemented with new technologies is not adversely influenced by past experience with old technologies.

To a large extent these low-cost batch-fabricated elements will also make possible the low-cost displays necessary to effect the close man-machine interaction called for by Keith Uncapher. At least the storage and logical functions of the man-machine interaction console can benefit significantly. If we do not place too stringent requirements on the visual image generation portion of the console and if input techniques such as the RAND Tablet are effectively utilized, a low cost display console should be feasible. After all, a television set that utilizes no batch-fabricated electronics and that includes RF, IF, and audio circuits can now be purchased for under $100. In more sophisticated man-machine consoles, communications costs will probably force inclusion of a general-purpose stored-program computer in the console.

First, I would like to discuss very briefly some areas of research and development related to computers and central processors that offer promise in the short range and then some much more significant areas for the long range in which the computer and central processor are negligible factors. It should be emphasized that avenues of research and development discussed here are promising in the sense of potential payoff if the problems can be successfully solved, rather than promising from the standpoint of likelihood of success.

SHORT-RANGE PROMISE

Utilization of Large Arrays from Fabrication and Interconnection Standpoints

Utilization of very large functional arrays of interconnected circuits is essential to achieving the very low cost potentials discussed by Rex Rice. This is also essential from the standpoint of improved reliability and maintainability. However, one is faced with two major problems in considering large arrays:

> The possible need for eliminating bad or substandard circuits from the array to achieve a reasonable yield.
>
> The lack of flexibility resulting from large arrays which tends to make each array within a system unique.

There are three major approaches to the utilization of large interconnected arrays from the systems standpoint that are under consideration. The first is cellular logic in which large arrays of identical circuits are fabricated with a standard interconnection pattern (e.g., connecting each circuit only to its four adjacent neighbors) with the ability to modify the function of the circuit by changing something in the circuit subsequent to fabrication. For example, one approach of this type uses a circuit with four cut-points which can be cut in different combinations to alter the function of the circuit.

In the second approach, a large array of circuits is fabricated and each circuit is individually tested. The test results are put in a computer which is also storing the logical equations of the function to be implemented. The computer then generates the proper interconnection pattern to interconnect available good elements (skipping the bad ones) to perform the required logical function. In this approach, a separate mask must be prepared for each array fabricated; hence, this is an expensive operation unless cheap methods can be developed for producing interconnection masks under computer control. Several such mask fabrication techniques are under development. On the other hand this approach offers a major advantage since it is easy to vary the function performed by the array by changing the logical equations supplied to the computer. If each interconnection mask for each array is generated individually, there is little incentive for rigidly standardized functions.

The third approach is advocated by those who believe that in the future it will be technically feasible to achieve high yields of large integrated circuit arrays in which all circuits are good. This would permit a standardized interconnect pattern to be used for each specific

logical function. This has the advantage that only one mask need be made for a particular function. This mask can be used to interconnect the circuits in many arrays of that type. On the other hand, it is more difficult to change the function to be performed by the interconnected circuit array since this requires making a different mask.

Since I believe that the integrated circuit industry will achieve success in either the second or third approach, the cellular logic approach appears to be of only interim interest as a solution to the problem of achieving a reasonable yield in large arrays of interconnected circuits. If cellular logic approaches to machine organization find a place in the computer industry, it will be for other reasons. When the second and third approaches prove feasible, we will probably find fixed interconnect patterns used for arrays produced in relatively large quantities where flexibility is not as important—i.e. highly repetitive functions such as storage arrays, registers and adders, etc. Computer-controlled variable interconnect techniques will be used for low-volume applications such as the control portions of a computer where each function tends to be unique.

Highly Functional Organizations

In order to facilitate the use of large arrays and to minimize the interconnections between batch-fabricated units, extensive research and development in new machine organization and system design techniques is needed. Logical design and machine organization approaches must be developed that will permit a machine to be organized along much more highly functional lines than at present so that a computer can be assembled from a limited number of relatively large functional blocks. These functional blocks should be self-contained to the greatest possible extent with a minimum number of signals passed from one functional block to another.

With the low-cost elements postulated by Rex Rice, such functional organizations can be achieved at the expense of quite inefficient logical design within each block. The number of logical elements within each functional block will be relatively unimportant. The number of interconnections between functional blocks will be of major importance. Hence, logical designers need to concentrate on minimizing interconnections between batch-fabricated arrays rather than on minimizing flip-flops and gates as in the past. The criteria for machine organization and system design should be adaptability to batch-fabrication technologies rather than logical efficiency or minimization of logical components.

Modular Computers and Multi-computer Systems

The development of modular computers and multi-computer systems offers other approaches to the standardization of relatively large modules within the system. If the machine is organized along highly functional lines as discussed previously in order to use very large arrays, there will be a strong tendency for each functional block to be unique in a single computer.

Two different approaches may offer promise in this area. One is the design of highly modular computers in which a given module that performs some specific function is relatively slow but is repeated many times in the system to provide a high over-all systems speed. The Holland and Solomon machines are examples of this concept although I am not suggesting those specific designs as the proper solution to the need for highly modular organizations from this standpoint.

Another approach, multicomputer systems, has been under active investigation for a number of years and several successful systems have been designed on this basis. The Navy Tactical Data System, which provides for up to three computers working together on a common problem with direct data interchange, is a primary example of this type system. However, multicomputer systems to date have been limited to a relatively small number of computers in each system. To aid in the problem of utilization of large quantities of identical batch-fabricated arrays, it will be necessary to think in terms of multicomputer systems utilizing a large number of very small standardized modular computers. This not only raises machine design problems but also severe programming problems requiring software research and development to permit the effective utilization of large numbers of small standard modular computers in a multicomputer system without prohibitive overhead costs for executive control routines.

LONG-RANGE PROMISE

The avenues for research and development discussed above are considered short range since it is believed that reasonable solutions will be found within the next few years. From the long-range standpoint, these areas will not present significant problems. If we accept Rex Rice's postulate of essentially zero cost logic and storage elements for the future, it is a reasonable next step to postulate essentially zero cost computers and internal storage.

This brings us face to face with the hard fact that reducing the costs of the central processor and internal memory to zero would probably reduce the total system cost by less than 50%. This ratio will vary of course

for different type systems, but looking at the broad range of all types of information processing systems 40 to 50% is probably a good average estimate. Reducing costs of the low-level digital logic and storage portions of peripheral equipments to zero might result in another 10% reduction in systems cost. We are still left with roughly half the systems cost that is not affected by the advances in electronic and magnetic batch-fabrication technologies and the zero cost elements that Rex Rice postulated. Hence, from the long-range standpoint, this half of the system presents the promising areas of research and development with respect to the potential payoff if successful. I have no real solutions to offer but would like to outline three major areas of this type and suggest some possible approaches.

Input/Output Equipment

The most significant problems in future systems will be encountered in the input/output area. There are three major approaches to improving the performance of future systems with respect to input/output equipment. These are:

Improvements in the performance of present types of input/output equipment.

Development of new types of input/output equipment that are not in widespread use at present.

System organization approaches that minimize the need for conventional input/output equipment.

Each of these approaches will play a part in performance improvements in future systems. However, unless much greater effort is placed upon the development of nonmechanical input/output equipment, the best hope for future systems probably lies in developing system techniques that minimize the need for input/output equipment.

Improvements in Conventional Types of Input/Output Equipment. Almost all present types of input/output equipment involve electromechanical techniques and components to a large extent. Many also involve high-voltage or high-powered electronics which are not amenable to batch-fabrication technologies. This imposes limitations on the improvements that can be achieved and on the ability to utilize the benefits of batch-fabrication of electronic and magnetic components. Although these electromechanical input/output equipments will limit systems performance, the effect on systems cost and reliability is even more serious. The performance limitations could be overcome to some extent by using a larger number of input/output units, but this further accentuates the cost and reliability imbalance with respect to the central processor and memory. Performance improvements of less than one order of magnitude, and in most cases of less than two-to-one, over equipment commercially available today are anticipated.

New Types of Input/Output Equipment. Several new types of input/output equipment are under development that offer promise for performance improvements in future systems. These include:

Character recognition and print readers
Voice recognition and voice output
Nonmechanical keyboards
Graphic input and output
Solid-state replacements for magnetic tape equipment

Some of these, such as optical character readers, are in limited use at present, while others, such as voice recognition equipment, are probably 10 years or more away. Advances in integrated circuit logic components and batch-fabricated memories will provide significant reductions in cost of flexible character recognition and voice recognition since the implementation of equipment of this type involves complex logical functions.

The important role that keyboards have always played as a man to machine-language transducer will be facilitated by the development of new types of keyboards that do not involve mechanically moving parts and that may permit more design freedom from the human factors standpoint. New graphic input devices, such as the RAND Tablet, coupled with low-cost displays will further facilitate man-machine communication and interaction.

Solid-state replacements for magnetic tape may improve the speed and reliability available for this type of input/output function, but cost competition with magnetic tapes is questionable. If solid-state storage modules that can be plugged into read-write electronics in a manner somewhat equivalent to placing a reel of tape on a tape unit prove feasible and economical, the input/output and off-line storage functions presently provided by magnetic tape could be provided by high-speed high-reliability devices and media with no moving parts. A BORAM device of this type, providing random access to a block of data in the storage module, could also be used as a replacement for electromechanical on-line mass memories.

System Organization to Minimize Input/Output. The greatest improvement in the performance of input/output equipment can be achieved by avoiding input/output operations wherever possible. By keeping the data within the system and by capturing data at the

source, much of the need for conventional types of input/output equipment can be eliminated. For example, the need for voluminous printed reports can be reduced sharply if the user is operating on-line with the processor through an efficient console. When any part of the data base within the system is rapidly available to the user upon request, he will have little need for large reports that are used for occasionally looking up printed results—particularly since these may be out of date by the time they are used. In general, any effort to increase the extent to which systems are "on-line" will tend to reduce the amount of conventional input/output equipment in the system. Achieving the improvements possible in this area will require a combined effort of users, programmers, hardware engineers, and systems planners and designers.

Very Large Capacity Mass Memories

Very large capacity mass memories represent another major problem area for future systems, which is closely related to the input/output problem discussed previously, since similar techniques and mechanisms are used at present. Batch-fabricated electronic and magnetic technologies will provide solid-state on-line auxiliary storage with reasonably large capacities in the order of 10^8 bits. Cryogenic techniques may push this up to 10^9 to 10^{10} bits. However, for very large auxiliary storage requirements in excess of 10^9 or 10^{10} bits, electromechanical devices will be required for the foreseeable future to keep the costs within reason. Of course, this somewhat contradicts the points made previously under the concept of zero costs storage elements, but almost zero cost per bit multiplied by 10^9 bits may still be an undesirably large dollar figure. Even if solid-state random-access on-line auxiliary storage costs drop to something in the order of 0.1¢ per bit, electromechanical auxiliary storage will still be required to achieve costs in the order of .001 to .01¢ per bit for very large capacity storage. Further research and development in block-oriented random-access memories (BORAM) may eventually provide a solution to this problem that will eliminate the necessity for electromechanical mechanisms with consequent improvements in cost, size, and reliability.

Communications

As the costs of digital elements and equipment continue to drop significantly, we will quickly reach a point where communications costs become a major problem unless improved communications techniques are developed. Keith Uncapher has pointed out one approach to this. The communications problems takes on greater importance as we tie together remote computers into a geographically dispersed data processing network.

Those proposing "computer utilities" appear to sweep this problem under the rug a little too easily. If you give the average housewife a free terminal, she probably cannot afford the telephone costs to tie into the computer utility in Chicago or Kansas City. The only really incontestable argument for the computer utility concept is the need for a common data base (where the "data base" includes both data and programs). All other arguments for a computer utility are really based on economic decisions that will vary as technology changes.

In the past, the cost of a large high-speed computer has been much less per operation than that for a small computer. As logic and storage elements become very cheap, this argument falls by the wayside. When it becomes cheaper to provide a stand-alone computer that permits occasional access to a large centralized data base than to provide more frequent communications for a terminal that depends upon a public utility for its computing capability, we will find small stand-alone computers in widespread use. If the local drug store or market can have a competent stand-alone computer for $10,000, it will not be economic for them to tie up a telephone line for continuous communication with a centralized computing utility unless significant improvements are made in communications techniques.

Fortunately, the concept of zero cost logic and storage elements offers some promise for a partial solution to the communication problem. Keith Uncapher has already pointed out the effects of more extensive use of digital communication techniques. In addition, the relative cost between computing and communication will change to the point that small users can be provided their own relatively sophisticated stand-alone computer and internal memory capable of accessing a remote large-scale system when necessary. With this approach, it will not be necessary to provide frequent or high-speed data communications between the central system and a typewriter-like terminal for each step in the process. Instead, the user's own stand-alone computer will perform most of his work. Access to the central large-scale system will be required only for certain data or programs and some types of calculations that may be too complex for the small computer. Computing and storage capability in the terminal will minimize the data exchange required with the central system.

Conclusion

In closing, I would like to summarize some specific avenues of research and development from the systems standpoint that appear promising with respect to potential payoff.

- Low-cost BORAM or other types of large capacity alterable on-line storage.
- Low-cost and more useful display consoles compatible with integrated circuit and batch-fabrication technologies.
- Low-cost "true" input/output (e.g., printers, keyboards, graphic input, optical character readers, voice input, etc.).
- System design and application concepts that minimize input/output operations at the expense of more internal logic and memory.
- Modular computers and multicomputer systems.
- Improved concepts of increased functional modularity.
- Machine languages that facilitate compiling operations.
- Improved automatic fault isolation and maintenance.
- Techniques for minimizing communications requirements.

I am sure that each of you can compile a longer list in your respective areas. The promising avenues of research and development from the standpoint of potential payoff are those directed toward the problems that will limit the capabilities of future systems. Unfortunately, these are the difficult problems rather than the glamorous ones.

PART II

DISCUSSION

Note: The space available has limited the published discussion by the panel as well as by the audience. The taped audience discussion has been condensed and edited. The panel has tried to include the principle thoughts of the audience but has had to reduce verbage. We apologize in advance if we have unintentionally misinterpreted or misrepresented the remarks by members of the audience.

IVAN FLORIS (*Stevens Institute*): . . . I don't think we can solve a universal language for engineers and scientists. I think that we should ask more of them to try to structure their problem in a language which is not a natural language but rather a formal one. If we try to make it too easy for them, we are going to find out that we have intellectual machines dealing with more professionals and I don't like that idea.

TOM STEEL: I think perhaps the best response to that business about getting trained professionals to all use a formal language is that (I agree) it isn't hard for us to learn FORTRAN, it is just a damn nuisance. It probably isn't hard to learn Swahili either but I don't want to bother. It is all well and good to talk about requiring that the professional learn precision, learn how to state his problem properly and to be formal and precise when it is necessary, but there is such a thing as being formal just for the sake of being formal and the kind of languages that we have now put an excessive burden on the user to take care of grubby details associated with solving his problem that he ought not to have to bother with. That is, one of the reasons these machines were designed and built was to relieve the man who was solving a problem from fussing around with all the grubby details of adding 2 and 2 and getting 4. Now we'll just carry it one step further. There's a lot of mechanical organizational elements in sequencing a set of calculations that the user shouldn't have to fuss with. He has better things to think about.

CARL BROOKS (*Northrop Corporation*): I would like to make a comment based on that—a question which I think perhaps is addressed most directly to Keith Uncapher. In trying to build the locked box (Fig. 9), for which there is no need for the user to glimpse inside, I don't disagree with this as a goal—I think this is a fine goal but I would like to draw one of the many parallels that one can make with current analog and digital machines and from there sort of point out a paradox with this situation. The range of one common breed of analog machine such as Keith mentioned is from —100 volts to +100 volts. One has the parallel situation in a digital machine of a given six-bit word length. Scaling is always a problem and no matter what engineer you are trying to turn this problem over to, he can always find a problem that is going to blow your scaling out no matter how long you make your word or how many volts you make your range on the analog machine. Hence, when you get to that problem (and we in the research business are faced currently with that problem as much on a digital machine as on an analog machine), you come to the point where you have to look inside the box. You have to find out what are your eventual hardware limitations. Nobody has built a software package yet that will overcome this. Now I would like to ask Keith what prospects he

sees for the development of a new outlook in software or a new outlook in hardware—a complete breakthrough into a new kind, an entirely new concept in computing that would overcome these problems which run parallel in both analog and digital machines.

KEITH UPCAPHER: I am not sure that I see any breakthrough, but what I do see though is the ability for us to understand the needs, let's say, for the engineer and to provide for him the appropriate feedback and limited formatting of information. The total system design and philosophy should be such that the user is alerted if he attempts to exceed the bounds of the system. But within the bounds, he should constantly see precisely 9-place accuracy, just as if he were going to an engineers handbook. This is really difficult to come by at present. That is, it's difficult to design a system which takes care of the messy round-off problems, yet seals off the user from the messy details of the computer. I think Cliff Shaw did a beautiful job with JOSS in this regard. Cliff arranged the language so that there is no payoff for it and really there is no way to look into the box and decide if it is a 36-bit machine, whether it is binary, or whether it is anything.

JEAN SAMMET (IBM): My comment and implied question is directed to Tom Steel. While I agree with almost everything that he said, I have to very seriously question the numbers that he used because I think they are all off an order of magnitude and unfortunately in the wrong way. I am perfectly willing to grant some deviations, but Tom used the figure of approximately 500,000 instructions, today's instructions, for the package that he was talking about. I will point out that I believe the amount of sfotware that is in the current project MAC system, not the users' program, but those things which are in some reasonable sense considered part of the system itself comprise approximately 500,000 instructions right now. I think that just about everything in there will be needed if this integrated system and probably at least as much more. I think it would probably be not too unreasonable to say that you are off by a factor of about 10. I would be much more inclined to believe that the number of instructions would be in the neighborhood of five million rather than half a million.

REX RICE: During the panel session, I think Tom Steel brought up a point concerning the environment that was required as well as the requirement for the program.

TOM STEEL: . . . I think the direct answer to the implied question you ask is, "I need a check for 10 million dollars." I think you need about 20 people, programmers, and need to give them 5 years and detach them from having to do 97 other things so that they can concentrate on doing this job. You can't totally isolate them because in order to do this right, they are going to have to have access to guinea pig engineers to try out some of their ideas. I think a very important aspect of this is that the people who design the programs also write and check out the programs. I have seen systems, that have been designed by one set of people, coded by another set of people. First of all, the results didn't meet the design specifications and secondly, the resulting program was about five or six times as big as it needed to be and I think that is where perhaps we are having some disagreement on the size of this package. To go straight to the heart of it, I think MAC is badly done and it could be cut down considerably.

JOHN REYNOLDS (Stanford University): I can't resist interjecting one remark that 5 million words of sloppy code may be less expensive than 500,000 words of well-written code. But more seriously, I am bothered that there hasn't been a distinction here between languages which are easy to learn and languages which are easy to use. I think it is perfectly reasonable to expect a professional engineer to spend as much time learning to program as to learn to do drafting. . . . What I am afraid of is that there is going to be an emphasis on avoiding powerful languages just to make it simple to keep the languages easy to learn. What is really important, particularly with the vital class of problems which are not put on machines, is not that the people who want to do them are too lazy, but simply that in present-day languages the coding effort is too enormous to develop more powerful languages, rather than languages that are easier to learn. And by power I want to stress two things: first simple conciseness—the ability to write code rapidly—and secondly, generality. Too many of these problems run across the rather arbitrary boundary lines that are set up in the so-called problem-oriented languages so that you may have to orient a language to every problem differently unless you set up much wider boundaries.

NED CHAPIN (Consultant): . . . I think it is implicit in the panel discussion that they must have considered the tradeoff in hardware and software, but I have heard them be very quiet on the subject. I wonder if they could make a little noise.

CHARLIE HOBBS: I think the vote on that is, with the kinds of hardware costs that we are looking at for the future, the tradeoff comes in favor of the hard-

ware. The hardware costs for central process and internal memory will be so cheap that we can afford to pay a very high price in terms of number of components to achieve economies elsewhere, such as programming for the user.

BILL HUGEL (Stewart-Warner): Perhaps I have to defend my own projections, which perhaps in 1970 don't show 3 cents—maybe only a dime—I am not sure. But in any case, some of the difficulties you face with this large array having 60,000 devices are due, I believe, to the fact that you are talking about something maybe of the order of 90% yield or maybe even 99% yield for a good company like Fairchild. I am not sure, but if you raise that point nine or point nine nine to the sixty thousands power, the yield comes down rather drastically. Then if you take the ten to the tenth circuits that are required and apply the new yield factor, I believe that the capacity outstrips even Fairchild and our own put together.

REX RICE: We are not now using redundancy to get around the yield problem because of the low level of the circuit sophistication. We are postulating that this will in fact be required. . . .

CHARLIE HOBBS: There are three approaches that I am aware of that are being worked on for overcoming the problem of yields with respect to large arrays. One is the use of cellular logic . . . in which a large array is interconnected with essentially a simple and standardized pattern and the price is paid in terms of efficiency or effectiveness of a circuit. To achieve this, each circuit has the ability to be modified physically by cutting or in some way breaking some contacts to change its function to implement the logical function. The second one is the approach of using a computer control fabrication of the mask. That is, you test the individual circuits, enter these test results into a computer that also has a logical equation, generate the program interconnections, then make the mask to connect the good circuits and leave out the bad ones. The third approach is for those who are optimistic, and I think there are a number in the semiconductor industry that feel this way—that yields will be so good that none of this will be necessary. . . .

WALTER DOUGHERTY (IBM)): I would just like to suggest that there may be a point that has been overlooked. I am suggesting that you have a central data base and perhaps a nationwide central data base or some large regionwide central data base and a reasonably powerful stand-alone computer which might sometimes access this central data base. I think this is very reasonable for the engineer and for the problem solver but I think that one of the major things we face is handling of information from the point of view of education and from the point of view of simply having access to information for the drugstore owner, for the local grocer, etc. I suggest that maybe a further level should be interjected between the large national data base and the powerful terminal or powerful local computer. This other level might be sort of a regional system. I am not suggesting a public utility necessarily but I do think there is a need for a regional-wide data base and a regional-wide computing facility, where you would have the possibility of cheap communication.

KEITH UNCAPHER: I don't think I would want to argue with you. I think either implies a communication system which has the general properties of those I tried to describe and something that doesn't exist and I don't think is going to exist without some fairly prompt action and a lot of dedication, perhaps by means of us in this room. The building of a network which will fulfill the requirements offered by you and others all seem to suggest that a low cost (per user) nationwide user-user computer-computer network must exist and I vote for an all-digital approach.

HAROLD CANTNOR: As one of the many million potential casual users, I got a bit confused and I wonder if the panel can orient me in saying whether they plan to do something for the casual user, or to the casual user, or with the casual user.

KEITH UNCAPHER: I suggest we do something for the casual user. . . . I think for the airline reservation clerk, the industry has provided a fairly good on-line system which doesn't require the user to know anything about the system. . . . I hope we can do the same for the engineer. . . . I think we are beginning to see in the environment at RAND, with JOSS, that there is a significant new class of JOSS users who have never found FORTRAN, etc., worth the challenge. One could perhaps argue for good or bad reasons that these users should have used other alternatives, but they haven't—even in a one-hour turn-around time environment.

HAROLD CANTNOR: Do you believe that a human being can truly carry on a dialogue with a machine, or it is through the technological devices that he carries on discussions with his fellows?

KEITH UNCAPHER: I think he can carry on a conversation with respect to the solution of a problem with a computer system—absolutely.

BRUCE MCKEEVOR (General Electric Computer Labs): . . . The original statement of the panel's objective

is that they are worried about expanding the market for computers to handle those 80 to 90% of the potential users who have stayed away from computers in droves. Those are just the people that we experts don't have any, or very little experience with. The airline example was mentioned. It was possible for the experts to learn enough about some particular user to develop a system for him. I am worried about the generality of this being true and I would like to throw two alternatives out to the panel to let them chew on it for awhile. One: Stick fairly close to what you are trying to develop but instead of developing a system, develop some hardware and a set of languages for the user, develop some collection of software supplemented by suitable hardware —that the user can shape into his own system. At the time he is using it, he can have an inconsistent language and use it where it doesn't get him in trouble. Where it does get him in trouble, he can patch his own way around it on his own terms. Now, an even further out alternative, perhaps one that is practical gets back to the question of the free hardware. . . . I am wondering if perhaps what we can do is apply this idea indirectly. . . . Go ahead and build a batch processing system, a national one that is huge or a gigantic time sharing thing. . . . However, let the system be as complicated and sophisticated, as powerful as is necessary, and require that the system be exercised by a set of experts, each of whom now is, in effect, a broker. The customer comes in, formulates his problem and the user pays the company owning the machine some fee and the expert (broker), whether he be a programmer or an applications engineer, gets a percentage of the take. . . .

TOM STEEL: Well, it seems to me that to a not inconsiderable extent that is precisely the situation we are in now. . . . In the first place, most of these people for very good reasons don't want to have to be bothered to sit down and explain what their problem is to somebody else and make him understand it. We have already discussed the fact that he probably has not clearly formulated his problem and that is part of what he does when he interacts with the machine directly. It helps him formulate his problem. He can do this in a two-stage process by talking to this paragon of virtue, the programmer who satisfies everybody, and maybe he can formulate his problem by interacting with that guy, but then there is the other step of having to put it on the machine. It doesn't seem to make much sense from anybody's point of view.

KEITH UNCAPHER: . . . First of all, I agree with Tom. I think what was proposed is about what we have today, namely a service programmer of sorts who also happens to be a broker. If I try to answer for the engineer for instance, I realize that there are some exceptions but the engineer, or at least many that I know of, looks at the service programmer as a bossy, bilateral, filter and we would like to get rid of him so that there could be direct interaction. . . .

REX RICE: On that high ending note, I thank the audience for their very provocative remarks and invite them to come up and have at us. Thank you.

Late Papers Not

Appearing in Volume 27, Part 1

A high-speed thin-film memory: Its design and development

Q. W. SIMKINS

International Business Machines Corporation Systems Development Division, Poughkeepsie, New York

INTRODUCTION

Memories in today's high-performance systems are typically made up of memory modules of capacity comparable to the new memory to be described here. Cycle times are 500 nsec to 1 μsec with access times of 300 to 500 nsec. This paper presents the design of a thin-film main memory with a capacity of 8,192 words of 72 bits each. The cycle time of this memory is 120 nsec with an access time of 60 nsec. Thus, this memory design represents a 4-to-8-times improvement in main memory performance over the present state of the art.

The first section of the paper is a general description of the physical layout of the memory. The memory array is then discussed with particular emphasis on noise problems in the array. Finally, the memory circuits and packaging are described.

GENERAL SYSTEM DESCRIPTION

The memory layout is shown in Fig. 1. The approximate dimensions are 68" high by 42" deep by 7" thick. The memory array consists of two 1,024-word by 288-bit assemblies which are driven and sensed in parallel. 72 of the 288 bits read in parallel are gated out. Each assembly contains 72 3" $\times$ 3" bit plates arranged in a back to back 6 $\times$ 6 configuration. The dimensions of each assembly are 20" $\times$ 20" $\times$ 4". The drive and sense circuits and associated logic are packaged on cards mounted on multilayer boards which surround the array.

Array Design

The memory array is made up of copper bit plates on which the films are deposited, striplines of Mylar*

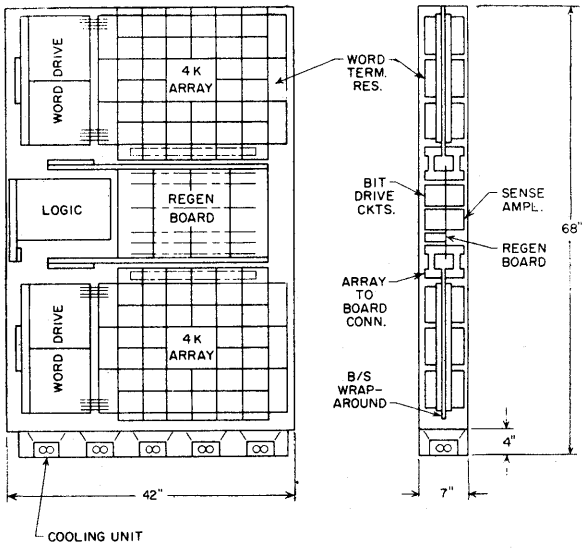


Figure 1. Memory layout—8K gate.

foils, and a ferrite keeper which partially closes the return flux path of the film memory elements around the striplines.

The thin-film elements utilized in this memory are 25 $\times$ 30 mil rectangles on 30 $\times$ 54 mil centers. The permalloy films are about 800 Å thick. They are switched in ~5 nsec by means of orthogonal drive fields produced by fast-rising current pulses in two sets of striplines. The word lines are closest to the film and are parallel to the easy axis of the film as shown in

*Mylar is a registered trademark of the E. I. du Pont de Nemours & Company.

Fig. 2. The bit lines are perpendicular to the word lines and separated from them by a thin dielectric foil. The sense line is parallel and coplanar with the bit line.

Registration of the striplines to the bit plates is achieved by mounting the striplines on rigid frames with appropriate mechanical registration fixtures. The line pattern is etched with the copper-Mylar laminate in tension, and a bit plate-stripline registration tolerance of ± 3 mils is maintained.

Array Noise Considerations

The signal energy from a thin-film memory array is low. In this memory the signal level is 4 millivolts, signal duration about 5 nsec, and the sense line impedance 50 ohms. At this signal level, array noise must be minimized if reliable memory operation is to be achieved. Noise results from unwanted energy coupling into the array at read time and at write time. The read noise results from capacitive coupling between the word line and the sense line. Since this noise is coincident with the signal, it directly affects memory operating margins. Write noise results primarily from coupling of energy from the bit line to the sense line. Since the sense system must recover from the write noise before the next read operation, the read-write cycle time is directly affected.

Read noise in an array in which all lines are terminated in their characteristic impedance is given by the following equation:

$$V_{rn} = \frac{C_{sw} Z_{ow} Z_{os}}{2} \times \frac{dIw}{dt}$$

where C_{sw} = the capacitance between word and sense lines,

Z_{ow} = the characteristic impedance of the word line,

Z_{os} = the characteristic impedance of the sense line, and

$\frac{dIw}{dt}$ = the rate of rise of the word line current.

This noise, unless canceled, is comparable to the signal level. First order cancellation is obtained by using a dummy sense line which runs between the bits on the bit plate. This is illustrated by Fig. 3. The outputs of the sense and dummy sense lines are sensed differentially using a balanced pulse transformer.

Since the bit and sense lines are closely spaced and long, the coupling between them gives rise to a large write noise (30 mv). The reactive component of write noise results from inductive and capacitive coupling between the bit and sense lines. This occurs during the rise and fall of the bit current. It determines the recovery time for the array and sense amplifier. The capacitively coupled noise from the bit line divides and is propagated in opposite directions on the sense line. Thus, the polarity of the capacitively coupled bit noise at the two ends of the sense line is the same. The inductive coupling, on the other hand, results in noise of opposite polarities at the ends of the sense line. Thus, on one end of a properly terminated sense line inductive and capacitive noise will add, while on the other end

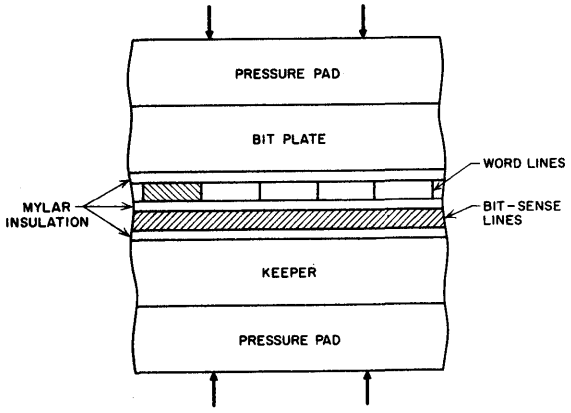


Figure 2. Array sandwich cross section.

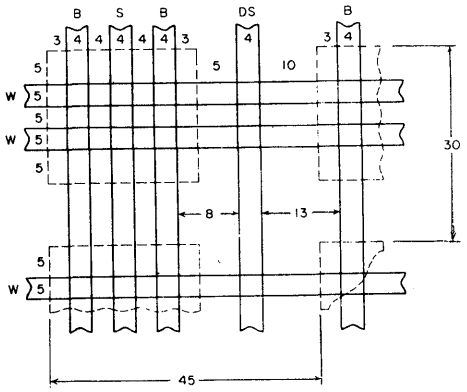


Figure 3. Array plan view—bit and line.

it will subtract. We take advantage of this directional coupling[1] by locating the sense amplifier and the bit driver at opposite ends of the array. The noise reduction thus gained may be as high as a factor of five. A second component of the bit noise is due to resistive coupling in the ground plane. This noise has approximately the same pulse shape as the bit pulse and is controlled by the magnitude of the ground impedance. An additional source of write noise is energy coupling to the sense line from the circulating ground plane currents. This noise is low in amplitude, but since the time constant for ground plane current spreading may be as

long as 500 nsec, this noise may be present during the read portion of the next cycle and so must be considered in the design of the sensing system.

Write noise may be controlled by a number of techniques. Directional coupling has already been described. In addition, several balancing techniques are available. A symmetrical arrangement of bit, sense and dummy sense lines may be employed in the bit sense lines to achieve first order cancellation of self-induced bit noise. This arrangement, however, is wasteful of bit current, and not effective in eliminating coupling between adjacent bit lines. Both self-induced and adjacent bit line noise can be canceled by driving and sensing two arrays in parallel. This arrangement, used in this memory, is illustrated in Fig. 4.

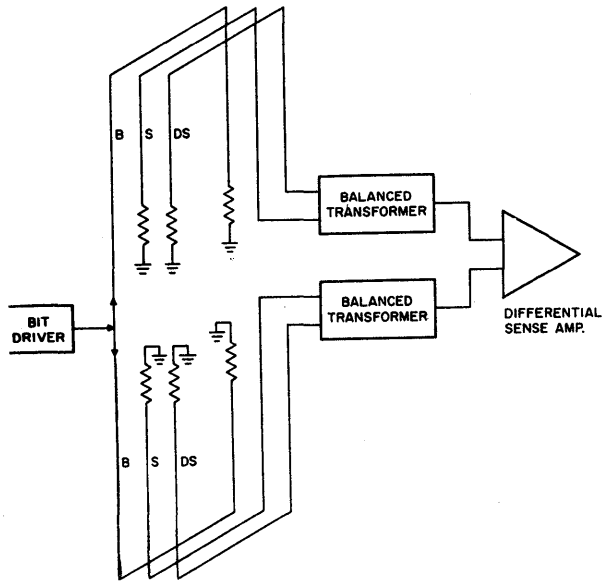


Figure 4. Bit-sense system.

Memory Circuits

The word selection system employs a transistor matrix (one transistor per word line). The nominal word current is 500 ma with a rise time of 7 nsec. The matrix transistors and word and gate drivers are packaged in a card-on-card approach similar to that employed in Solid Logic Technology (SLT).[2] All line impedances are controlled to provide line matching and prevent troublesome reflections. The word lines are terminated in approximately their characteristic impedance (i.e., 17 ohms). Interconnections between the array and circuit boards are made by means of coaxial cables.

The bit and sense circuits are packaged between the two halves on the array. The bit driver circuit provides 100-ma pulses into each of two parallel bit lines with a rise time of 7 nsec. These pulses are positive or negative depending on the information to be stored. The sense amplifier has a matched impedance input fed from a balanced pulse transformer. The output of the sense amplifier is gated into a tunnel diode detector, which feeds a data-register latch circuit. The sense amplifier, detector, data-register and bit driver are all packaged on the same board to minimize delay in the regeneration loop.

The films are vacuum deposited on a $3'' \times 3''$ highly polished copper substrate about 80 mils thick. The substrate is precoated with a micron or more of silicon monoxide to smooth the surface and improve the film properties. This deposition is made in the presence of a strong, uniform magnetic field and at a precisely controlled substrate temperature to produce the desired anisotropy in the films. An initial screening of the bit plates is done by checking magnetic properties using Kerr magneto-optic measurements prior to photoetching into the desired bit pattern. Typical properties are $H_c = 4$ oe, $H_k = 5$ oe, with skew and dispersion less than 2 degrees.

After etching, the bit plate is pulse tested under worst-case operating conditions. Each of the 4,128 bits on the plate is tested with a pulse pattern that corresponds to worst-case conditions, including information pattern, magnetic history, ground plane currents, trapped flux effects, disturb conditions, and marginal currents. Worst-case conditions involve simultaneous word and bit disturb, including disturb effects from adjacent lines, as well as those from lines directly over the bit. A bit plate is accepted only if all bits on the plate meet the minimum specifications for the bipolar one and zero signals.

The copper bit plate serves as the ground plane for the striplines. Interconnection of the bit plate ground planes is achieved with pressure connections around the bit plane periphery. To improve reliability of the ground system, the edges of the bit-plate are rhodium plated.

A metallic substrate, rather than glass or mica, was selected for this application for the following reasons:

1. Close proximity of the ground plane to the sense line reduces noise and permits higher-speed operation.
2. Lower line impedances reduce power requirements of drive circuits.
3. The ground plane reduces field spreading which permits closer spacing of the bits.
4. Metal substrate has more uniform temperature during film deposition resulting in more uniform film properties.

Disadvantages of the metal substrate, which include ground plane current and trapped flux effects,[3] are minimized by the keeper. The memory design includes etched discrete bits rather than a continuous magnetic film for these reasons:

1. Disturb effects (creep) are less severe with etched bits, permitting higher packing density.
2. A lower noise-balanced sense system with a dummy sense line running between the bits can be used.

Conclusion

This paper has described briefly the design of a large, high-speed film memory which represents a substantial speed improvement over the current state of the art in main memories. Key features and design choices highlighted here include the magnetic film element design, the metal ground plane substrate, the use of discrete bits, and the electrical design of the array to minimize noise effects.

Appendix

FILM MEMORY ARRAY PROPERTIES

Word lines

Z_o	=	17 Ω
length	=	24″
attenuation	=	<5%

Bit lines

Z_o	=	31 Ω
length	=	46″
attenuation	=	<5%

Sense lines

Z_o	=	60 Ω
length	=	46″
attenuation	=	25%

Signal = 4 mv (at the film)

Noise

Read Noise	≈	2.4 mv (uncanceled)
	≤	0.4 mv (canceled)
Bit noise	≈	65 mv (uncanceled)
	≤	30 mv (canceled)

REFERENCES

1. G. F. BLAND, "Directional Coupling and Its Use for Memory Noise Reduction, *"IBM Journal of Research and Development,* vol. 7, no. 3 (July 1963).
2. J. J. Corning et al, "Solid Logic Technology: Versatile, High-Performance Microelectronics," ibid, vol. 8, no. 2 (Apr. 1964).
3. V. T. Shahan, and C. J. Townsend, "Measurement of Trapped Flux and Ground Plane Current Effects," 1964 Intermag. Conference.

Efficiency and management of a computing system

H. D. HUSKEY

University of California
Berkeley, California
and Universidad Católica de Chile

INTRODUCTION

The study of efficiency in computing systems can be subdivided in the following way: (1) efficiency of hardware, (2) efficiency of software, and (3) efficiency of operating systems. Efficiency in this context will be considered as performance for the average customer divided by potential performance. It is granted that both "average customer" and "potential performance" are concepts about which it is difficult to be precise and rigorous.

Every computing laboratory should use a certain amount of "instrumentation" so as to be able to define the average customer and to have some information about customer distribution. Potential performance depends upon the customer distribution. Care should be exercised in collecting such information because of "feedback effects." For example, if it is known that the use of tapes is being monitored, the users will be more conservative.

Priority systems enhance the performance for some customers at the expense of performance for others. Although some aspects of priorities may encourage more efficient use of the computing system, there will be no attempt in this presentation to consider their effect.

HARDWARE

The design of computer hardware systems has been difficult because of the tremendous differences between data rates in the parts of the system. Data, in many cases, originate at manual rates of a few characters per second. If punched in cards then the cards may be read at rates of 250 to 1000 per minute. If cards average 30 characters each (a bit high), the rate is 500 characters per second. If transferred to magnetic tapes or disc files, the data rates may be from 100,000 to 1 million characters per second into the computer processor. Automatic operations may be done at close to a million per second. Then the output process perhaps involves the transfer of information to a line printer capable of printing 1,000 to 2,000 120-character lines per minute. Again, only rarely will all 120 characters be significant.

From the above discussion it is obvious that throughput, at least for small problems, depends more on input/output capability than on arithmetic speeds.

In the past, word length in a computer was more related to the number of core planes that could be driven, and to command length considerations, than to accuracy requirements in computation. In a university, most computation is for educational purposes and almost any number of significant figures is satisfactory. For example, illustrations of loss of accuracy in numerical analysis can be taught as well with four significant digits as with eight.

The input data for much of the computation are relatively rough, so that computing with eight significant figures may have no meaning.

Doing multiple precision arithmetic slows most computers by substantial factors. However, if six significant digits were sufficient for 80% of the problem load (and I think it is), then it might be better to put the incremental cost of more digits of precision into input/output hardware.

Some computers have been built with inadequate round-off facilities. Many problems (particularly differential equations) involve millions of arithmetic operations in sequence, so a slight bias in round-off may accumulate with disastrous effects.

Almost all computers have index registers which facilitate the processing of arrays. The real choice here lies between hardware index registers and index registers in memory. Hardware registers are fast and expensive and complicate the interruption process (since they must be saved). Memory index registers are cheap, permitting much larger numbers. They may be incorporated in a paging system which minimizes the cost of interruption.

Scientific and engineering computation typically involves processing integers and real (floating point) numbers. Some computers have been built so that integer and real arithmetic are done the same way. This permits the mixing of types of operands in an arithmetic expression with no extra commands.

In computational processes such as the solution of systems of linear equations by elimination, the majority of the commands in the program are involved with the modification and testing of indices. The inner loops of such programs (those commands executed the most times) can be simpler (and faster) if flags on the data can be used to "interrupt" the computing at the end of a row or column.

SOFTWARE

Most scientific and engineering problems are done using a problem-oriented language such as FORTRAN. It has been said that it required 18 man-years of effort to produce the first FORTRAN translator. At this level of effort, just any computation laboratory is not about to modify the FORTRAN translator. Consequently, one settles for that FORTRAN system which the computer company produces. Naturally, such a processor has been designed so as to supply the needs of all customers in an optimal way. If the majority of the users can run their problems on a simpler one-pass system (one-pass indicates that the source language deck is read once, object code generated, and perhaps executed in place in memory), then the remainder can run using the full system. Thus, the majority of users are not paying the overhead represented by the unused portions of the complete system. For example, how many FORTRAN users use the complex number capability?

Monitors, or operating systems, may not utilize all the hardware capability of a given computer. This may be because of size limitations, i.e., the larger and more sophisticated the monitor, the less space there is left for the customer in the high speed memory.

Techniques in translating are well developed so that it is no longer a major task to write a compiler. Consequently, computer laboratories should custom design their operating systems and perhaps write translators for languages suitable for the majority of their customers.

OPERATIONAL PROCEDURES

Most computer centers suffer from a lack of information about what is being done by the customers. On the other hand, it is easy to collect too much information and increase the overhead costs of doing computing.

Batch processing implies a minimum turn-around usually of some hours. Unless there are restraints, the user (1) tries alternative forms of the program—in case he doesn't understand formats, for example, (2) runs more cases than necessary just to be sure of spanning the area of interest, and (3) takes extensive memory dumps for debugging purposes.

In undergraduate classroom use, one has to choose between restrictive measures, which minimize extra runs or extra input and output, or less restrictive measures permitting the student to do essentially whatever he likes. This last option is not necessarily bad since it permits the student to explore alternate approaches to solving the problem.

Magnetic tape is used inefficiently in many computer laboratories. Many times reels of magnetic tape have only a few feet with recorded information.

At 500 characters per inch an 80 column card corresponds to about one sixth of an inch of tape. This is comparable to the start-stop space needed on the tape. Therefore, if the customer reads cards it is easy for him to establish 80 character records on tape and lose 50% in efficiency both in terms of quantity and speed.

One may think that disc storage solves these problems. However, disc space may be handled by system subroutines and these may use buffer sizes which lead to similar inefficiencies.

ON-LINE SYSTEMS

The use of consoles on-line to a computer will change the character of the inefficiencies. Perhaps the most serious problem is that the user is now encouraged more than ever to "play around" at the console. With high traffic usage leading to sign up for time at consoles this can be controlled some. However, it will be only user self-discipline in the end that produces efficient use.

The use of Culler-Freed methods where the user calls for evaluation of expressions for 100 to 200 values of an argument will use up computer time at a tremendous rate. Again the real premium will be on adequate planning so that those minutes at the console will be well used.

In on-line systems a great advantage will come from the fact that the user need not ask for extra information so the relative printing load will decline markedly. Since he can sustain attention on his problem he can complete the task in minutes instead of days and eliminate reorientation time.

A proper on-line system has to supply bulk storage for keeping all users' data (programs, numbers, text, etc.) available on demand. Soon the system becomes loaded with junk. Perhaps user priorities should be developed which are inversely proportional to the amount of bulk storage used by the customer.

Except for the use of bulk storage most of the actions of a console user will have no effect on the efficiency of use at other consoles. Thus, in contrast to batch processing, only the individual suffers from lack of self-discipline.

Use of digital computers in basic mathematics courses

WILLIAM D. MARSLAND, JR.

The Frank J. Seiler Research Laboratory, U.S. Air Force Academy
Colorado Springs, Colorado

A digital computer is now available at most colleges and universities in the United States. In those where there is none today, the advent of time-sharing systems in the next several years should result in a computer capability for all at a nominal cost.

Much effort has been expended and much has been written on the use of computers in engineering education. To a much lesser extent, efforts are under way to introduce the use of computers in the behavioral and social sciences. It seems to me that one very important and basic area where a digital computer could be very helpful is being overlooked, namely, to provide increased insight and understanding of many basic areas of algebra, trigonometry, analytic geometry and calculus, or introductory modern analysis.

Let us consider several examples from the aforementioned areas and see how the computer could be used as an aid in giving the student a greater insight and understanding.

In the development of Naperian logarithms one defines:

$$\lim_{x \to 0} (1 + x)^{1/x} = e$$

A rough calculation of this limit is usually given for three or four easily calculated points showing that it does indeed gradually increase above 2.7. The comparatively simple computer program presented in Fig. 1 (written in Burroughs B5500 Extended ALGOL) can bring out the approach to the limit much more vividly.

As more display devices become available, the above results may well be shown even more effectively in the form of graphical output. At the ACM 1965 National Conference, G. Culler of the University of California described a research project of this type that is being investigated this fall in mathematics classes.

```
          BEGIN
REAL          X,E;
FORMAT        F(F10.6,F20.10);
FILE          LINE 4(2,15);
              FOR X ← .00050 STEP -.00005 UNTIL .00010, .00009 STEP
              -.00001 UNTIL .000005 DO
          BEGIN
              E ← (1 + X)↑(1/X);
              WRITE(LINE,F,X,E);
          END;
          END.

                    0.000500          2.7176025403
                    0.000450          2.7176705084
                    0.000400          2.7177384016
                    0.000350          2.7178062976
                    0.000300          2.7178741955
                    0.000250          2.7179420919
                    0.000200          2.7180099813
                    0.000150          2.7180781143
                    0.000100          2.7181460557
                    0.000090          2.7181595571
                    0.000080          2.7181730365
                    0.000070          2.7181864848
                    0.000060          2.7182005454
                    0.000050          2.7182140037
                    0.000040          2.7182273962
                    0.000030          2.7182406571
                    0.000020          2.7182555664
                    0.000010          2.7182691575
```

Figure 1.

In studies of probability one is rarely able to generate empirical data using present-day classroom-teaching techniques. For example, consider tossing an honest coin and counting the heads and tails. A teacher might make an assignment for each student to toss a coin (assumed to be honest) 100 times and to count the number of occurrences of heads. Then in class the results could be collected and the overall results discussed. The teachers would then point out what one might expect if the number of tosses were increased. With the aid of a digital computer and a pseudo-random number generator, data for 50 sets of 10, 100, 1,000, and 10,000 tosses can be generated in less than 5 minutes providing much more data and the opportunity for observation of what happens as the number of tosses is increased, thus providing the student with the data needed to gain insight and to draw his own conclusions. The program given in Fig. 2 assumes the existence of

the pseudo-random number generator. This approach offers the opportunity for a side discussion of what is meant by a random number and the problems of generating one.

In calculus we define the Riemann integral as the limit of a summation process. The approach of the sum to a limit for a few specific examples can be demonstrated quite readily using the computer. Consider:

$$\int_0^2 x^3 dx = \frac{x^4}{4}\Big]_0^2 = 4$$

A computer program using rectangles to approximate the area under the curve is given in Fig. 3 for the cases using 2, 4, 10, 100 and 1,000 rectangles.

The latter example also serves as a natural introduction to methods of numerical integration, a topic frequently overlooked in calculus courses, but essential in many practical problems.

Two of the examples given above have been related to the convergence of some process. In many calculus courses the convergence of series is discussed, but the student is normally left with the impression that if one can prove that a series converges, the goal has been achieved and everything will work out all right. Unfortunately, even with the fastest computers presently available, one frequently cannot afford the luxury of using enough terms of the series to compute desired results. Rates of convergence should also be discussed in calculus and can be nicely demonstrated with the aid of the computer.

```
          BEGIN
FILE          LINE 4 (2,15);
INTEGER       I, J, K, COUNT, X, SUM ;
REAL          MEAN ;
FORMAT        F(X60,17,116), 8(/X60, "MEAN = ", F20.10), H(X60,
                "NO. OF TOSSES", X3, "NO. OF HEADS"/);
COMMENT   IT IS ASSUMED THAT THE COMPUTER CENTER HAS A RANDOM NUMBER
GENERATOR AVAILABLE. THE CODE FOR SUCH A GENERATOR SHOULD BE
INSERTED HERE.  SEE ALGORITHM NUMBER 133 IN THE COMMUNICATIONS OF
THE ACM FOR AN EXAMPLE OF A RANDOM NUMBER GENERATOR.  ;
          FOR J = 10, 100, 1000, 10000 DO
          BEGIN
          WRITE(LINE, H);
          SUM = 0;
          FOR I = 1 STEP 1 UNTIL 50 DO
          BEGIN
COUNT = 0;
          FOR K = 1 STEP 1 UNTIL J DO
          BEGIN
          X = RANDOM(0,1,0);
          IF X = 0 THEN COUNT = COUNT + 1 ;
          END;
          WRITE (LINE, F, J, COUNT);
          SUM = SUM + COUNT ;
          END;
          MEAN = SUM/50 ;
          WRITE(LINE, G, MEAN);
          WRITE(LINE(PAGE));
          END;
          END.
```

NO. OF TOSSES	NO. OF HEADS	NO. OF TOSSES	NO. OF HEADS	NO. OF TOSSES	NO. OF HEADS	NO. OF TOSSES	NO. OF HEADS
10	3	100	44	1000	522	10000	5001
10	7	100	52	1000	514	10000	4970
10	5	100	52	1000	481	10000	5071
10	7	100	51	1000	497	10000	4984
10	4	100	55	1000	518	10000	4951
10	5	100	47	1000	503	10000	4997
10	7	100	40	1000	493	10000	5007
10	5	100	50	1000	525	10000	4866
10	4	100	45	1000	492	10000	4986
10	5	100	51	1000	503	10000	4943
10	6	100	52	1000	505	10000	5000
10	7	100	47	1000	498	10000	5098
10	8	100	47	1000	512	10000	5019
10	5	100	45	1000	511	10000	5100
10	4	100	40	1000	479	10000	5026
10	5	100	44	1000	498	10000	5005
10	5	100	40	1000	481	10000	5051
10	7	100	49	1000	498	10000	4973
10	3	100	56	1000	507	10000	4977
10	3	100	50	1000	494	10000	4969
10	8	100	54	1000	445	10000	4985
10	7	100	46	1000	495	10000	5022
10	6	100	46	1000	507	10000	4992
10	5	100	49	1000	490	10000	5048
10	8	100	59	1000	448	10000	4961
10	4	100	53	1000	493	10000	5005
10	5	100	47	1000	511	10000	5015
10	7	100	44	1000	507	10000	4991
10	7	100	41	1000	497	10000	5057
10	2	100	46	1000	499	10000	5093
10	4	100	58	1000	513	10000	5063
10	6	100	56	1000	480	10000	4930
10	4	100	54	1000	450	10000	5057
10	5	100	52	1000	497	10000	5032
10	9	100	49	1000	484	10000	5011
10	3	100	56	1000	487	10000	5095
10	3	100	52	1000	511	10000	5051
10	6	100	43	1000	504	10000	4976
10	6	100	46	1000	500	10000	5055
10	6	100	49	1000	494	10000	5004
10	5	100	55	1000	499	10000	4995
10	4	100	51	1000	499	10000	5018
10	5	100	61	1000	492	10000	4957
10	4	100	50	1000	500	10000	4927
10	7	100	52	1000	482	10000	5059
10	4	100	45	1000	491	10000	4965
10	7	100	45	1000	494	10000	5027
10	7	100	52	1000	496	10000	5094
10	3	100	58	1000	480	10000	4948
10	4	100	38	1000	509	10000	4994
MEAN =	5.3200000000	MEAN =	49.6000000000	MEAN =	495.7000000000	MEAN =	5007.6600000300

Figure 2.

```
          BEGIN
FILE          LINE A (2,15);
FORMAT        F(X61, I4, X5, F15.5), G(X55, "NO. OF RECTANGLES",
                  X2, "AREA OF RECTANGLES"/);
INTEGER       I, J;
ARRAY         X, HEIGHT, AREA [0:1022];
REAL          TOTALAREA, STEPSIZE;
              WRITE(LINE,G);
              FOR I := 2, 4, 10, 100, 1000 DO
          BEGIN
              TOTALAREA := 0;
              STEPSIZE := 2/I;
              X[0] := 0;
              FOR J := 1 STEP 1 UNTIL I DO
          BEGIN
              X[J] := X[J-1] + STEPSIZE;
              HEIGHT[J] := X[J]**3;
              AREA[J] := HEIGHT[J] * STEPSIZE;
              TOTALAREA := TOTALAREA + AREA[J];
          END;
              WRITE(LINE, F, I, TOTALAREA);
          END;
          END.

 NO. OF RECTANGLES   AREA OF RECTANGLES

                 2            9.00000
                 4            6.25000
                10            4.84000
               100            4.08040
              1000            4.00800
```

Figure 3.

The examples selected have been purposely so simple that with an ample number of time-sharing consoles around a campus, an instructor might easily assign one for students to program and reasonably expect the students to have the results at the time of the next class meeting. There are many other examples which are equally simple. By asking the student to explore the solutions for himself, I believe he will gain much greater insight and understanding of the processes which he now accepts frequently on faith and without much appreciation of the basic concepts.

With the advent of time-sharing systems I expect to see at least one very simple language or a very simple subset of a more sophisticated language on most systems. I believe it will be very commonplace to find that the basic computer courses will gradually migrate to the freshman year. I feel the time is at hand for mathematics departments to consider how the computer can serve them in providing increased insight in the topics they are presenting. However, mathematics departments frequently make little or no use of computers and it may often be necessary for other computer-oriented faculty members to guide and assist the mathematics faculty in the use of computers.

Some other areas where I feel a computer could be a useful tool are given below. The list is not intended to be all inclusive and you are invited to augment it with your own examples.

Demonstration of the solution of equations representing radioactive decay.

Approximation of the zeros of a function.

Demonstration of finding the delta for a given epsilon in continuity discussions.

Proving trigonometric identities (this would be a very sophisticated application).

Solution of selected transcendental equations using iteration techniques.

Numerical computations using De Moivre's theorem.

Tests for parallelograms, straight lines, and various special kinds of triangles and polygons.

Generating empirical data for various probability studies.

Empirical data on arithmetic and geometric progressions, nth term of a progression, etc.

Demonstration of why one cannot prove theorems by example instead of by mathematical induction.

Discussion of the need for a hierarchy of operations.

Generation of binomial coefficients, Pascal's triangle.

Demonstration of various limit theorems; some false conjectures could be thrown in for testing using the computer.

Demonstration of convergent and divergent sequences and series.

Determination of coincident and parallel lines.

Solution of simultaneous equations, including cases where the coefficients are measured quantities with the possibility of limitations on the measuring equipment.

Synthetic division techniques for roots of polynomials.

Checking for rational roots of an equation.

Slope of the tangent to a curve at a point as a limit process.

Evaluation of a function near a maximum or minimum point.

Demonstration of amplitude period, and phase shifts for various forms of trigonometric functions.

Demonstration of limit processes arising out of applications of L'Hospital's rule, e.g.,

$$\lim_{x \to 0} \frac{\sin x.}{x}$$

Verifying definitions of ellipse and parabola as locus problems.

Demonstration of the derivative of a function as a limit process.

Using graphical output devices, generate envelopes of curves by use of derivatives to show usefulness in curve sketching.

Demonstration of the conclusions of Rolle's theorem and the mean value theorem.

Integration by the trapezoidal process and various numerical methods.

Development of an algorithm for evaluating an

$n \times n$ determinant.

Evaluation and sketches of hyperbolic functions.

Generating data for polar curves and other parametric representations of curves.

Three-dimensional representations of vectors (sophisticated).

Least square and other curve fitting methods for realistic amounts of data.

Examination of maxima and minima of functions of several variables.

Multiple integrals as limits of double summations.

Demonstration of Fourier series approximations of functions for varying numbers of terms in the series.

ERRATA: Final Version of Papers Appearing in Preliminary Draft Form in Volume 27, Part 1

MATHLAB: A program for on-line machine assistance in symbolic computations*

CARL ENGELMAN

The MITRE Corporation
Bedford, Massachusetts

THE PURPOSE OF MATHLAB

A mathematical scientist experiments. Today, his test tube and his breadboard are blackboard and paper. He may, it is true, have available a computer, but its role is numerical and its results are delivered not today or tomorrow but the day after the final programming bug is corrected. The computer is not present during the most creative phases of the scientist's labor. The purpose of MATHLAB is to provide the scientist with computational aid of a much more intimate and liberating nature.

What sort of aid? The basic goal is to provide facilities for those operations which are mechanical. Among the most common of these are the addition of expressions and equations, the substitution of subexpressions into a larger expression, differentiation, integration, Laplace Transforms, multiplication of matrices, and the solution of simple equations. Although the greater part of a scientist's time is spent on these mechanical pursuits (in fact, an appreciable portion is probably spent in simply checking answers and in the eternal bookkeeping problems of getting minus signs and 2 π's right), we must keep in mind that most of the tedious computations associated with the creative aspects of his work are of a symbolic, rather than a numerical, nature. If we are to free the scientist from his routine mathematical chores and conserve his energies for the more properly human activities of interpretation, analysis, planning and conjecture, then we must mechanize the passage from r^2/r to r in addition to that from $1 + 1$ to 2.

REQUISITES FOR A MATHEMATICAL LABORATORY

I should like to outline here the properties I feel are required of a mathematical laboratory, not in terms of the range of mathematical operations available, but rather in terms of its spirit and feel.

1. It should be capable of ordinary numerical computation. This implies the ability to perform arithmetic, to compute functions or to look up their values in tables, and to draw graphs.
2. It should be capable of a wide spectrum of symbolic computations.
3. It should respond to simple user commands. MATHLAB is intended for the physicist, not the programmer. The commands should be no more complicated than his thoughts. If he wishes to enter an equation into the computer, he should need only to type the equation in a notation like that of ordinary mathematics. If he should then wish to differentiate that equation with respect to x, he should have to give a command no more complicated than "differentiate (x)."
4. It must be expandable by the expert. The language, functions, and subroutines of the laboratory must be such that it will grow as an organism. If today we write programs for symbolic differentiation, we should expect, tomorrow, to employ them in programs for power series expansions. The opportunity to expand the pro-

*This project was supported by The MITRE Corporation Independent Research Program. It was also supported in part, through access to its computer facilities, by Project MAC, a M.I.T. Research Program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract No. NONR-4102(01).

grams should be open to anyone who masters a well-defined and common computer language.

5. It should be extensible by the user. Although the ability of the physicist to augment the existing programs will no doubt be severely limited compared to that of the programming expert, he should be provided tools for doing certain simple things for himself, such as changing notational conventions or teaching the machine the derivatives of his favorite functions.
6. The computer, as viewed by the user, must be intimate and immediate. The user should have next to his desk a console consisting of a typewriter or, preferably, a typewriter and a scope. Economy might, in some cases, dictate the substitution of a plotter for the scope. These are connected to a large, fast, on-line, time-shared digital computer. He communicates with that computer by typing messages on his typewriter or by means of a light-pen on the scope. The computer replies by means of the same machines. It types both messages and equations. On the scope it displays both equations and graphs. Above all, the response time to the user's requests must be short.

BRIEF SUMMARY OF EXISTING PROGRAMS FOR SYMBOLIC MATHEMATICS

The first program we should mention was written for Whirlwind I by J. H. Laning and N. Zierler[1] and was not really a program for symbolic computation. But, even though the program was capable only of numerical computation, it could accept programs written as simple symbolic mathematical expressions and perform them for a user with no machine language experience.

A later program that could accept instructions in the form of symbolic expressions, but which was also limited to numerical computations, was, of course, FORTRAN.[2]

The ALPAK system[3,4,5] at the Bell Laboratories, designed by W. S. Brown, J. P. Hyde, and B. A. Tague, is a program for the manipulation (multiplication, division, differentiation, etc.) of polynomials and rational functions. It is not a system of symbolic computation in the sense we are using that expression. Although its instructions are FORTRAN-like, its input/output, as well as its internal representations of polynomials and rational functions are lists of numbers representing the relevant coefficients and exponents. Owing to its entirely numerical nature, the program is quite rapid.

Another interesting development in the field of numerical computations is the STL On-Line Computer[6] designed by G. Culler and B. D. Fried at the Space Technology Laboratories. Although the programming facilities are similar to FORTRAN, the instructions are entered, in an on-line environment, through the pushing of buttons labeled "+," "sqrt," etc. After subroutines are debugged, it is possible, by pushing buttons, to define the effect of a short sequence of button pushes to be the same as any sequence of first-level pushes. These subroutines of higher-level button pushes can themselves be incorporated in the definition of button pushes at a still higher level.

Probably the most fundamental development, to date, for the adaptation of computers to symbolic computation is the design by J. McCarthy, for just such purposes, of a language called LISP[7,8] (for **LIS**t **P**rocessor). A great majority of the symbolic computation programs extant are written in this language.

The foremost problem of content facing the construction of a mathematical laboratory today is probably the writing of a satisfactory program for the simplification of mathematical expressions. Simplification is the unconscious of mathematics. We all simplify expressions every day, making choices appropriate to the occasion but of which we are almost totaly unaware. For this reason, it is much simpler to program a more advanced formal computation, such as differentiation, which has exact rules of which we are conscious, than it is to have the machine simplify the answer after it completes the differentiation. We cannot afford to enter into a detailed discussion of the current attempts to solve this problem, but we would like to mention, in chronological order, the authors of three LISP programs for the simplification of mathematical expressions: T. Hart,[9] D. Wooldridge, Jr.,[10] and W. A. Martin.[11]

A LISP program for symbolic differentiation was written by K. Maling.[12] Although it suffered from some weakness of simplification and from its input/output being restricted to well-formed LISP expressions (which are not nearly as legible as those written in ordinary mathematical notation), it was certainly a dramatic early demonstration of the ability of LISP to handle formal mathematical computation.

In my opinion, the most impressive example of symbolic mathematics yet performed by machine is the formal (or indefinite) integration program of J. Slagle, written as a doctoral thesis under Prof. M. Minsky at M.I.T.[13] Perhaps the most interesting aspect of this program is that it was heuristic rather than algorithmic. It possessed a small table of integrals and tried to re-

duce the problem to one or several found in that table by the same bag of tricks possessed by a good college freshman.

The remaining programs as we shall discuss do not, like those above, represent attempts to perform particular symbolic processes (simplification, differentiation, integration) on a computer. They are, rather, attacks on the whole problem of building a system of symbolic computations.

The first such program we should like to mention is the mathematical laboratory project of W. A. Martin, a work-in-progress as a doctoral thesis at M.I.T. under Prof. M. Minsky. This work is by far the closest known relative to our MATHLAB. At present it appears that the main difference of emphasis will be that Martin's work will stress very broad input/output capabilities. He is, for example, working on input from scopes achieved by signifying with a light-pen an interesting subexpression of a previously "printed" expression,[14] as well as anticipating using the scopes for handwritten input. The emphasis in our program is more in the direction of continually increasing mathematical powers. Since both programs are written in LISP, a good deal of exchange should be possible.

Another approach to the mathematical laboratory problem is a project, under the direction of L. C. Clapp,[15] while he was employed at Bolt, Beranek, and Newman. The programs are not written in LISP but in a much simpler and weaker list-processing language of Bolt, Beranek, and Newman's own invention, called SIMPLIST. While this language and program might serve well for rapid performance of simple computations, e.g., adding two symbolic equations or evaluating a function entered symbolically, it seems unlikely that they will be capable of difficult symbolic procedures, such as a powerful simplification program or symbolic integrations. These limitations stem primarily from the fact that their work is married to the PDP-1, a small computer.

The next system we should like to discuss is the IBM FORMAC system. This is an extension of FORTRAN which provides it with the ability to perform a certain amount of symbolic computation. It is under development in Cambridge under a group headed by J. E. Sammet.[16] It is quite capable, possessing such abilities as simplification, substitution, expansion of products of sums, factoring out powers of a given variable, and differentiation. For a number of reasons, however, it does not seem well suited to a mathematical laboratory. It is program oriented. One does not give a simple command, such as "differentiate" but rather one writes a program. True, the language, like FORTRAN, is easy for a nonprogrammer to learn, but it will not have the universal accessibility of, say, the simple commands of our MATHLAB. The FORMAC programs can, of course, be run on-line, but they cannot be run line by line, nor were they intended to be. It is necessary to write an entire program, for the program has to be compiled as an entity, as does a FORTRAN program (in fact, it employs the FORTRAN IV compiler). Thus, although a useful tool for symbolic computation when you know at the outset what you want to do, FORMAC would be quite inconvenient for experimentation.

The final program we would like to discuss before turning to our own MATHLAB is the FORMULA ALGOL of Perlis and Iturriaga. In one sense, this language is similar to FORMAC; namely, the ability to manipulate formulas, i.e., mathematical expressions, is embedded in what was previously a purely numerical compiler, in this case ALGOL rather than FORTRAN. Viewed this way, its built-in, as opposed to programmable, facilities are quite meager: restricted mainly to formula substitution and some rudimentary simplification. However, this view[17] of FORMULA ALGOL has been superseded by the creation of a powerful programming language[18] which combines the numerical facilities of ALGOL not only with some ability to manipulate mathematical expressions but also with the ability to create and manipulate list structures. In this role of combined algebraic and list-structure compiler, FORMULA ALGOL stands as a predecessor of LISP 2,[19] the type of language in which programs such as MATHLAB will probably be written in the future.

MATHLAB—ON THE SURFACE

MATHLAB is our current attempt at realizing a mathematical laboratory of the sort we have been discussing. The program, which has been developed on the time-shared system of Project MAC at M.I.T. and on the IBM 7030 at The MITRE Corporation, is continually growing, and the following description is accurate as of June 30, 1965. We do not feel that MATHLAB has, as yet, sufficient mathematical powers to be of aid to a general user, except with respect to special and occasional problems. It does, however, possess most of the qualities postulated in the previous section as requisites for a mathematical laboratory.

1. *Numerical computations.* It is very weak in this department because we decided at first to study symbolic computation as it represented the crux of our problem. It cannot draw graphs or evalu-

ate common transcendental functions. We can evaluate algebraic expressions with numerical arguments in a variety of ways.

2. *Symbolic computations.* Here we can perform many common tasks. We can simplify, substitute, add equations, differentiate, integrate a little, solve equations, etc.
3. *The user commands are simple.* If the user has stored an equation called "e1" and wishes to differentiate both sides of it with respect to "x" and call the resulting equation "e2", he need only type: differentiate (e1 x e2).
4. *The program can be expanded by any LISP programmer.* In fact, we are doing this all the time.
5. *MATHLAB can be extended a little by the user.* He can teach the machine the derivatives of functions and change the names of system commands.
6. *It is intimate.* The user types in some initial equations; the computer acknowledges them. The user requests certain symbolic manipulations; the computer performs them and types back the answer. The user types in some expressions or numbers and requests the computer to substitute these for certain variables in a previous equation; the computer types back the answer, etc.

Mathematical Notation

In this section and the next we wish to describe MATHLAB as it appears to the user. There are some minor differences between MATHLAB as it exists at Project MAC and on the STRETCH at MITRE. Where such differences occur, we shall describe the situation at MITRE. First, what sort of expressions may a user type to denote mathematical quantities? The answer is: those expressions, composed in the ordinary way, of the following entities:

1. Numbers: 1, 5/2, 2.5
2. Words, representing symbolic variables, composed of strings of letters and digits, the first of which is not a digit:
   x, distance, x1, x1sub2.
3. Operation symbols:
   + (addition)
   — (subtraction or minus)
   * (multiplication)
   / (division)
   ↑(exponentiation)
4. Parentheses: (and). These have two functions. The first is to ensure the desired interpretation of certain expressions, e.g., to distinguish 5*(x + y) from 5*x + y. The second use of parentheses is for functional notation, e.g., sin(x).
5. Comma: The comma, besides its end-of-message role to be discussed shortly, serves to separate the arguments of a function of several variables, e.g., f(x, y, z).

All blanks are ignored. The rules of precedence of the operational symbols are conventional (FORTRAN) except that, in the absence of parentheses, e↑x↑2 denotes e↑(x↑2), not (e↑x)↑2 = e↑(2*x).

For example, if the user wishes to enter the mathematical expression (in conventional notation): $\sin(5x + y^2)$, he need only type (in MATHLAB's notation): sin(5*x + y 2).

The System Commands

The program as we have developed it accepts three types of symbolic quantities, called *expressions, functions,* and *equations,* which are stored in the computer and which can be referred to and manipulated by *name.* An expression is a mathematical quantity or expression referred to by its (one word) name. A function is a mathematical expression and an ordered list of *bound* or *dummy* arguments and is also referred to by a (one word) name. An equation is really two mathematical quantities, one for the left and the other for the right side of the equation, referred to by a (one word) name. The initial expressions, functions, and equations in an experiment are entered into the computer by a function called "denote". For example, one might type:

```
denote nil
d = 1/2*a*t↑2,
e1 == r↑2 = x↑2 + y↑2,
f (x,y) = x + y,
,
```

The first three commas signify the end of individual definitions and the fourth comma tells the computer that this is all the information we choose to give it for the time being. The word "nil" is a vestige of the hidden fact that "denote nil" is really a couple for the LISP evalquote operator. The effect of this input is to store in the computer an expression whose name is d and whose value is 1/2*a*t↑2, an equation whose name is e1, whose life side is r↑2, and whose right side is x↑2 + y↑2 and a function whose name is f, whose list of dummy arguments is (x y), and whose value is x + y. From this point on the user never again has to type in r↑2 = x↑2 + y↑2 but can simply refer to e1.

Incidentally, the response to the above instruction (we shall, from now on, use the convention that we speak in lower case and the computer in capitals) is:

THANKS FOR THE EXPRESSION D, THE EQUATION E1, AND THE FUNCTION F.

In terms of these basic constructs of *expression, function,* and *equation,* we shall describe the various system commands.

repeat (*x*)

This command repeats x to the user. x may be the name of an expression, a function, or an equation. The format for an expression or a function is similar to that of denote. For an equation, if x were e1 above, then "repeat" would print:

(E1) R↑2 = X↑2 + Y↑2

This same format is used when any of the succeeding commands is called upon to print an equation.

Pleasesimplify(*x y*)

This command simplifies x and names it y. In this, as in the following commands, the name "y" may be the same as the name "x". In that case, the old x is lost. If "x" is the name of an equation, both sides are simplified independently. For a detailed discussion of the scope of simplification, the reader is referred to Ref. 10, which discusses the simplification program we employ.

forget(*x*)

If x is a complicated expression, function or equation, its storage might be burdensome. The command "forget" allows the user to retrieve the space when it is no longer needed.

substitute((*v*1 *v*2. . .*vn*) *x y*).

The first argument "(v1. . .vn)" must be a list of names of expressions and functions in any order. The *value* of each "vi" is substituted in x at each occurrence of the name "vi". The new equation, expression, or function (depending on x) is named "y".

At this point we should like to state more precisely the meaning of the denote and substitute instructions. Should we give the command:

```
denote nil
z = x + y,
```

x and z would be quantities of quite different natures. We shall refer to "x" as a *formal symbol;* it is without meaning. "z", on the other hand, is the name of an official expression; its meaning, which we normally refer to as its "value", is the quantity x + y constructed of the formal symbols x and y.

If we now type the instructions:

```
denote nil
x = 5*t,
,
```

an expression whose name is x and whose value is 5*t would be created, but this would in no way affect the status of x in the value x + y of the expression whose name is z. That x remains a formal symbol. The fundamental connection that can be established between these two occurrences (of different types) of the character "x" is through the instruction "substitute".

If we now type:

```
substitute( (x) z w)
```

the program will look for an expression whose name is x, find that its value is 5*t, look for an expression whose name is z, discover that its value is x + y, substitute the quantity 5*t (containing the formal symbol t) for all occasions of the formal symbol x in x + y (obtaining the quantity 5*t + y), simplify this to 5*t + y, and create a new expression whose name is w and whose value is 5*t + y.

Substitutions take place only upon command, never automatically. This is as it should be. The user may have previously informed the computer that x = r*cos(t), but he might like to type x without having it automatically changed to r*cos(t). Automatic substitution schemes are not only undesirable, but are also prone to interminable loops.

The substitution of a function in a function is defined recursively so that it will operate to any depth. For example, if

g(u, v) = f(f(u, v),f(u, v))

and

f(x, y) = x + y,

then the command:

```
substitute( (f) g h)
```

would yield:

H(U, V) = 2*U + 2*V .

It is probably better to think of *substitute* in this circumstance as a command to unwind the functional definitions.

There is an important restriction on the substitution of expressions within functions. One may only substitute for the unbound variables or parameters of a function. For example, if f(x) = x + t, one may substitute for t but not for x. It is expected that the casual user will attempt to violate this rule and instructive error messages have been prepared. This restriction is dictated by the meaning of a dummy variable of a function, but

any desired result can surely be achieved by a short sequence of commands. In this case, we could, for example, denote h(y) = f(x) and then substitute first f and then x in h.

This description may seem too detailed, but an understanding of the distinction between an expression and a formal symbol as well as the function of the substitute instruction is fundamental to an understanding of MATHLAB. We shall presume the extension of these concepts to equations and to other commands, e.g., differentiate, is apparent.

add((q1 q2... qn)name)

The q's can be equations, functions, expressions, or numbers in any order. If there is at least one equation among them, name is an equation; if not and some q is a function, name is a function; otherwise it is an expression. Equations are added by adding left sides and adding right sides independently. Expressions, functions, and numbers are added to an equation by adding their values to both sides of the equation. If functions are added to form a new function, the list of dummy variables of the new function is the union of the lists of dummy variables of the old functions.

multiply((q1... qn)name)...

Similar to above.

subtract(x y name)...

Similar to above, but only two equations, expressions, functions, or numbers are subtracted instead of an indefinite number as in add and multiply.

division(x y name)...

Similar to above.

raise(x y name)...

Similar to above. name = x^y.

negative(x y)...

y = — x.

invert(x y)...

y = 1/x.

flip(x y)...

x must be the name of an equation. y becomes the name of that equation with the left and right sides interchanged. This is useful if we wish, say, to add the left side of one equation to the right side of another.

makeequation(x y)...

x must be the name of an expression or function. If x is an expression, an equation is formed whose name is "y", whose left side is the name "x", and whose right side is the value of x. For example, using the expression "d" of *denote* above, if the instruction: "makeequation (d e2)" were given, the computer would respond: (E2) D = 1/2*A*T↑2. Should x be a function, the effect is best described by an example. Employing the function "f" of *denote* above, "makeequation (f e3)" would yield: (E3) F(X, Y) = X + Y.

makeexpression(e)...

e must be an equation whose left side is a single word. Then an expression is formed whose name is the left side of e and whose value is the right side of e. For example, "makeexpression(e2)" would now produce: D = 1/2*A*T↑2, which is where we started.

makefunction(e)...

e must be an equation whose left side looks like "f(x1, x2, ... , xn)" with all the xi simple (one word) formal symbols. We get a new function "f" whose dummy variable list is "(x1 x2 . . . xn)" and whose dummy variable list is "(x1 x2 . . . xn)" and whose value is the right side of e.

The purpose of these three "make" commands is twofold. The first purpose is to guarantee the legality of certain succeeding commands. In the example occurring in the description of "makeexpression" above, e2 could not be substituted in another equation but d could. The second is to ensure the accessibility of certain results. E.g., taking e2 and f as above, compare the effect of the two following sequences of commands:

First:

```
add((e2 f) e)
(E) D + X + Y = 1/2*A*T↑2 + X + Y
```

Second:

```
makeequation (f e101)
(E101) F(X, Y) = X + Y
add((e2 e101) e)
(E) D + F(X, Y) = 1/2*A*T↑2 + X + Y
```

expand((x1 x2... xn))...

This produces no immediate result but affects all succeeding simplifications. Whenever one of the x's (which are formal symbols) occurs in a product of sums, that product is multiplied out. The following dialogue will clarify this.

```
denote nil
e3 == y↑3 + y*y = (x + z) * (u + v),
THANKS FOR THE EQUATION E3
pleasesimplify(e3 e4)
(E4) Y↑3 + Y↑2 = (X + Z)*(U + V)
expand((x u))
YES
pleasesimplify(e3 e5)
(E5) Y↑ + Y↑2 = X*U + X*V + Z*U + Z*V
```

The "expand" command affects not only the command "pleasesimplify" but other commands, such as

"substitute" or "add", which always simplify their answers.

factor((*x y*. . .)). . .

Like "expand", this produces no immediate result but affects all future simplifications. It causes the collection of all terms containing, as a factor, a power of x and similarly for y. The order of formal symbols in the list "(x y. . .)" implies precedence. In this case, the term "x*y" will count as y occurrences of x rather than x occurrences of y.

We give an example:

```
denote nil
mitre = a*x + 2*x + x*y + y + b*y + 4*x
  ↑2 + c*x↑2,
,
THANKS FOR THE EXPRESSION MITRE.
pleasesimplify(mitre bedford)
BEDFORD = A*X + 2*X + X*Y + Y + B*Y
  + 4*X↑2 + C*X↑2
factor ((x y))
YES
pleasesimplify(mitre bedford)
BEDFORD = (2 + A + Y)*X + (4 + C)*
  X↑2 + (1 + B)*Y
```

By calling the functions expand and factor at different times with different arguments, the user can maintain control over the form of his answers.

differentiate(*y x yprime*). . .

Differentiates y with respect to x and calls the resulting expression or equation (depending on y) yprime. At present, all differentiation is explicit, i.e., a term is considered as constant in x, unless x appears in it explicitly.

learnderivative. . .

Allows the user to teach the computer the derivative of a new function. The format of this command is best explained by an example.

```
learnderivative nil
arctan,
x,
1/(1 + x↑2),
```

The need for this command has, to some extent, been obviated by an improvement in the differentiation program. If the computer is asked to differentiate arctan (x)↑2, it will decide that the answer is twice arctan(x) times the derivative of arctan(x). If it should then discover that it does not know the derivative of arctan, it will try to obtain it from the typewriter. If it succeeds, it will complete the differentiation and remember the derivative of arctan for future use.

integrate(*v x w*). . .

v must be an expression which is a rational function of x with (rational) numerical coefficients. w is its indefinite integral. For a more precise discussion, see the following section. We give an example. If:

v = (x + 1)/((x↑2 + 1)*(x↑2 + x + 1)↑3),

then the command

```
integrate(v x w)
```

yields the result:

```
W = (2/3*X↑3 + 1/2*X↑2 + 1/3*X  — 1/2)/
    (X↑4 + 2*X↑3 +3*X↑2 + 2*X + 1)
    + 1/2*LOG(X↑2 + 1  — ARCTAN(X) —
    1/2*LOG(X↑2 + X + 1)
    + 7/(3*SQRT(3))
    *ARCTAN((2*X + 1)/SQRT(3))
```

solve(*e x*). . .

e must be an equation that is equivalent to a rational function (with symbolic coefficients) of x. At present "solve" can handle only those equations which are really (although not necessarily explicitly) quadratic or linear in x. The reply to this command, excepting those cases which the program cannot solve, takes one of three forms depending on whether the computer analyzes the equation to be linear, quadratic with distinct roots, or quadratic with a double root. The following three examples illustrate these different responses.

1. If e is the equation: y = a*x + b, then "solve (e x)" yields:

```
THIS EQUATION HAS A SINGLE ROOT.
X = (Y — B)/A
```

2. If e is the equation: a*x↑2 + b*x + c = 0, then "solve (e x)" yields:

```
THIS EQUATION HAS TWO ROOTS, NAMED
  FIRSTROOT AND SECONDROOT.
FIRSTROOT = 1/2*(— B
+ SQRT(B↑2 — 4*A*C))/A
SECONDROOT = 1/2*(— B
— SQRT(B↑2 — 4*A*C))/A
PLEASE RENAME THOSE WHOSE PRESER-
  VATION YOU WISH TO ENSURE.
```

3. If e is the equation: x↑2 — 2*b*x + b↑2 = 0, then "solve (e x)" yields:

```
THIS EQUATION HAS A DOUBLE ROOT.
X = B
```

We give a fourth example which, though simple, exhibits more of the meat of the program:

4. If e is the equation: $1/(x\uparrow 2 - 1) = 1/(x - 1)$, then "solve (e x)" yields:

```
THIS EQUATION HAS A SINGLE ROOT.
X = 0
```

There are two points here. First, the equation does not appear, at first glance, to be linear but the program analyzes it as such. Second, a naive attempt at solving the equation, e.g., inverting both sides, could yield what we feel is an extraneous root at x = 1. For an explanation of how the program avoids this trap, see the following section.

rename(x y)...

The expression, function, or equation that had the old name "x" obtains the new name "y."

newname(A B)...

This command differs from all previous commands in that it affects, not the data, but the system commands themselves. It creates a new system command B whose effect is identical to A and which exists in addition to A. For example, if the user tires of typing "differentiate," he can give the command:

```
newname (differentiate d)
```

after which the command:

```
d (y x yprime)
```

will have exactly the same effect as

```
differentiate(y x yprime).
```

Before digging beneath the surface of MATHLAB, it might help clarify some of the preceding if we give a very short sample session possible today.

```
denote nil
e1 == r↑2 = s*t,
s = x↑2 * y,
t = log(w)/x,
;
THANK YOU FOR THE EXPRESSIONS S T
  AND THE EQUATION E1.
substitute ((s t) e1 e2)
(E2) R↑2 = X*Y*LOG(W)
denote nil
w = sin(x↑2 + y↑2),
;
THANK YOU FOR THE EXPRESSION W.
substitute ((w) e2 e3)
(E3 (R↑2 = X*Y*LOG(SIN(X↑2 + Y↑2))
differentiate(e3 x e3prime)
(E3PRIME) 0 = Y*LOG(SIN(X↑2 + Y↑2))
  + 2*X↑2*Y*COS(X↑2 + Y↑2)
  /SIN(X↑2 + Y↑2)
denote nil
;
THANKS FOR NOTHING.
```

This last crack on the computer's part is indicative of the fact that most of our commands have very heavy error protection. If the user makes a mistake in constructing an expression or an equation or tries to give an expression a name already assigned to an equation, etc., he will receive an instructive error return.

MATHLAB—BELOW THE SURFACE

LISP

The entire MATHLAB program is written in LISP.[8] The system commands are all addressed to the LISP evalquote operator, e.g., the command "differentiate (h t dh)" presents the evalquote operator with a couple consisting of the function "differentiate" and the list of arguments "(h t dh)".

Numbers

The LISP system, written by R. Silver and P. Heckel[20] for the IBM 7030 (STRETCH) at MITRE, contains only one type of number, namely, rational numbers, i.e., ordered pairs of integers. If any of the numbers, 12/5, 24/10, or 2.4, is typed in, it is converted to the rational 12/5.

In addition to rationals, MATHLAB possesses another type of number: the rational power of a rational number. It will, for example, compute the value of $(81/16)\uparrow(2/3)$ to be $9/4*(3/2)\uparrow(2/3)$.

Internal Representation of Mathematical Expressions

The internal representation of any mathematical expression is a well-formed LISP S-Expression in a prefix notation. If the user types:

```
denote nil
v = t↑2 + sin(pi*t),
```

then there is stored on the property list of the atom "v" the property EXPRESSION followed by the S-Expression:

```
(PLUS (EXPT T 2)(SIN(TIMES PI T))).
```

Should the user then type:

```
differentiate(v t dv)
```

there is stored, upon completion of the differentiation and simplification, on the property list of the atom "dv" a pointer to the S-Expression:

```
(PLUS(TIMES 2 T)(TIMES PI
  (COS(TIMES PI T))).
```

This is translated back into the original infix notation and the typewriter prints:

DV = 2*T + PI*COS(PI*T)

Equations are stored similarly to expressions, except there are two pointers, one to the S-Expression representing the left side of the equation and one to the right. Functions are stored by having the indicator FUNCTION point to the listed pair consisting of the list of dummy variables and the value of the function represented by the corresponding S-Expression.

Besides the obvious need for well-formed LISP expressions, there are two reasons for our choice of this internal representation of mathematical expressions. First, this representation has become fairly standard and this allows us to exchange programs with other workers. Second, the prefix notation turns out to be well suited to our applications. Consider the differentiation we just discussed. Probably the easiest and fastest thing for LISP to tell us about any list is the first item on it: in this case, PLUS. But this is precisely the first thing our differentiation program would want to know so as to invoke the rule that the derivative of a sum is the sum of the derivatives. Both the input (infix→prefix) and the output (prefix→infix) translation programs are written in LISP, the former employing the character-reading functions.

Other internal representations of expressions also occur, e.g., polynomials as lists of coefficients and rational functions as dotted pairs of lists of coefficients. All such alternative representations have translation programs connecting them in both directions with the standard prefix representation.

Simplification and Differentiation

The internal programs for simplification and differentiation have been borrowed from the Stanford Simplify Program.[10] They have been modified in two ways.

Simplify has been enlarged to handle a family of simplifications typified by the transformation: (EXPT (MINUS X) 4/3→(EXPT X 4/3,, i.e., $(-x)^{4/3} \rightarrow x^{4/3}$. This simplification was impossible in the original Stanford system because 4/3 could only be represented by an approximating decimal, such as 1.3333333 and nothing can be done with $(-x)^{1.3333333}$.

Differentiation has been modified so as to look to the typewriter for the derivatives of new functions. If necessary, the program will demonstrate, by example, the correct format for teaching it derivatives.

Integration

The program for the integration of rational functions with numerical coefficients was written by M. Manove[21] at MITRE in the summer of 1964. It is based on a theorem of Hardy[22] that states that the integral of such a function is of the form $R_1 + \int R_2$ where R_1 and R_2 are rational and R_2 has only simple poles. The program always finds R_1 and does the best it can with $\int R_2$, that best depending on its ability to factor the denominator of R_2. It is sufficient to consider R_2 monic with integral coefficients. To factor R_2, "integrate" first calls a simple program written by the current author which, after finding all rational (hence, integral) roots and factoring them out, will, if left with a quartic, factor it into the product of quadratics with integral coefficients (should such a factorization exist). In addition to this factorization program, "integrate" memorizes the factors of the denominator of the original problem (if that denominator is presented in factored form) and uses these as trial divisors of R_2.

"Integrate" is powerful enough to have found several errors in published tables of integrals.

Solve

"Solve" first brings the equation it has to solve over on one side. It then combines the various terms into a single rational expression with one numerator and one denominator, employing greatest common divisor routines (for polynomials with symbolic coefficients) to eliminate common factors from numerators and denominators. The roots of the original equation are then the roots of the numerator of the constructed rational function. If that numerator is quadratic, its roots are found by the quadratic formula. The vanishing of the discriminant is the test for a double root. We are, of course, dependent on the simplification programs here and within the g.c.d. routines to tell us when complicated S-expressions are equivalent to zero.

CURRENT WORK—September 1, 1965

Current work involves new representations of polynomials and rational functions in several variables with applications to simplifcation, the factorization of polynomials in several variables, and the integration of rational functions with *symbolic* coefficients.

In addition, we are working on programs for the display of mathematical expressions on scopes and the adaptation of MATHLAB to the AN/FSQ-32 computer at the Systems Development Corporation, Santa Monica.

REFERENCES

1. J. H. LANING and N. ZIERLER, "A Program for Translation of Mathematical Equations for Whirlwind I," Instrumentation Laboratory, M.I.T. Engineering Memorandum E-364 (Jan. 1954).
2. INTERNATIONAL BUSINESS MACINES CORP. *Reference Manual* 709-7090 *FORTRAN Programming System,* Prentice-Hall, Englewood Cliffs, N.J. (1958).
3. W. S. BROWN, "The ALPAK System for Nonnumerical Algebra on a Digital Computer—I," *Bell System Tech. J.* (Sept. 1963).
4. W. S. BROWN, J. P. HYDE, and B. A. TAGUE, "The ALPAK System for Nonnumerical Algebra on a Digital Computer—II, ibid (Mar. 1963).
5. J. P. HYDE, The ALPAK System for Nonnumerical Algebra on a Digital Computer—III, ibid (July 1964).
6. B. D. FRIED, "STL On-Line Computer, Volume I —General Description," Space Technology Laboratories, 9824-6001-RU-000, Redondo Beach, Calif. (Dec. 28, 1964).
7. *J.* MCCARTY, "Recursive Functions of Symbolic Expressions and Their Computation by Machine," *Communications of the ACM* (Apr. 1960).
8. "LISP 1.5 Programmer's Manual," M.I.T. Computation Center and Research Laboratory of Electronics (Aug. 1962).
9. T. HART, "Simplify, Artificial Intelligence Project," M.I.T. Computation Center and Research Laboratory of Electronics, Memo 27.
10. DEAN WOOLRIDGE, JR., "An Algebraic Simplify Program in LISP," Stanford Artificial Intelligence Project, Memo No. 11 (Dec. 27, 1963).
11. WILLIAM A. MARTIN, "Hash-Coding Functions of a Complex Variable," M.I.T. Project MAC Memorandum MAC-M-165, Artificial Intelligence Project, Memo 70 (June 25, 1964).
12. K. MALING, "The LISP Differentiation Demonstration Program," Artificial Intelligence Project Research Laboratory of Electronics and M.I.T. Computation Center, Memo 10.
13. JAMES ROBERT SLAGLE, "A Heuristic Program that Solves Integration Problems in Freshman Calculus, Symbolic Automatic Integrator (SAINT)," Ph.D. thesis, Mathematics Department, Massachusetts Institute of Technology, 1961.
14. WILLIAM A. MARTIN, "Syntax and Display of Mathematical Expressions," M.I.T. Project MAC, Memorandum MAC-M-257, Artificial Intelligence Project, Memo 85(July 29, 1965).
15. LEWIS C. CLAPP and RICARD Y. KAIN, "A Computer Aid for Symbolic Mathematics", *Proceedings of the Fall Joint Computer Conference,* Spartan Books, Baltimore, 1963.
16. J. E. SAMMET and E. R. BOND, "Introduction to FORMAC," *IEEE Transactions on Electronic Computers* (Aug. 1964).
17. A. J. PERLIS and R. ITURRIAGA, "An Extension to ALGOL for Manipulating Formulae," *Communications of the ACM,* (Feb. 1964).
18. Aa nlJ. Perlis, Renato Iturriaga, and TOMAS A. STANDIS, "A Preliminary Sketch of Formula ALGOL," Internal Report, Carnegie Institute of Technology, Pittsburgh (Apr. 9, 1965).
19. R. W. MITCELL, "LISP 2 Specifications Proposal," Stanford Artificial Intelligence Project, Memo No. 21 (Aug. 19, 1964).
20. P. HECEL, forthcoming MITRE Aechnical Report (untitled).
21. M. MANOVE, "Integrate: A Program for the Machine Computation of the Indefinite Integral of Rational Functions," The MITRE Corporation, TM-04204 (Apr. 22, 1965).
22. G. H. HARDY, *The Integration of Functions of a Single Variable,* Cambridge University Press (1958).

A time- and memory-sharing executive program for quick-response, on-line applications

JAMES W. FORGIE

Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts*

BACKGROUND

The TX-2 Computer, an experimental facility at M.I.T. Lincoln Laboratory, has been in operation since 1960.[1] Never a service facility, the computer has been used principally in a number of long-term research projects that have taken advantage of the special input/output capabilities and the direct accessibility of the machine. These projects have included graphics,[2,3] waveform processing,[4,5] and pattern recognition.[6,7] Most of the work on the computer has involved real-time inputs, interaction with output displays, or both. The computer has always been used as an on-line facility with the bulk of its time allotted in sessions of several hours duration. Programming has been in machine language, augmented in the past by a number of personal macro languages and recently by a more general macro language for list processing (CORAL). An on-line macro assembler, MK 4, has been used both as an assembly program and on-line operating system by most users.

In the fall of 1963 it was decided to realize on TX-2 an experimental operation-oriented, on-line system in order to study man-machine interaction in problem solving. This system would allow the scientist or engineer to make use of the computer throughout his work on a data-analysis problem without concerning himself with many of the details ordinarily involved in programming a computer.[8] The system would be based on a library of computational and display routines that could be called directly by the user in an appropriate problem-oriented language. For a problem area in which library routines existed, it was expected that single library routines, or short combinations of routines, would suffice for a high percentage of the operations he would need. In order to handle the few remaining cases, the system would include special and general compilers which the user could utilize to create the occasional pieces of program that he might need to complete the solution to his problem.

It was felt that a person using a system of this sort would probably spend much more time looking at his displays and thinking about what to do next, than he would spend actually doing computations. Economic considerations then dictated that multiple consoles should be provided and the computer facilities shared among these consoles. An executive program which would handle such sharing of the computer facilities and related problems of storage allocation and communication became a central part of the system design.

In January of 1964 a commitment was made to realize such an operation-oriented on-line system on TX-2. Since experimentation with the new system would put pressure on the already full schedule of TX-2, a further requirement was placed on the design of the new system, namely, that its executive program should allow for the use of TX-2 in something approaching its accustomed style at the same time that the system was running. Thus the advantages of time-sharing could be made available to the projects already using the machine. This paper discusses the design of the executive system, called APEX, which has grown out of those decisions.

SYSTEM REQUIREMENTS

The design requirements for the APEX system may be briefly stated as follows:

1. *Time Sharing.* The system should time-share the

*Operated with support from the U.S. Air Force.

essential computing facilities among a small number of consoles (perhaps half a dozen) most of which would consist of an input keyboard and either an output typewriter or a display oscilloscope, or both.

2. *Fast Response.* The on-going activities in graphics, waveform processing, and pattern recognition all involved the use of interactive displays. It appeared that response times in excess of one second would seriously degrade the performance of already existing programs in these areas. In addition, the proposed experiments with the operation-oriented on-line system called for the ability to degrade response time in order to measure its effect on the user. Thus, all proposed applications of the system called for fast response under at least some circumstances.

3. *Retention of Results.* The executive should assume responsibility for the retention of all programs and data files whose destruction was not specifically ordered by the user or his program.

4. *Subroutine Autonomy.* The executive should allow any program, assuming it is written as a closed subroutine and follows certain conventions, to be run as an independent program making full use of core storage addresses and index registers. Routines to be run in this fashion should be precompiled and stored in absolute binary form. They should be completely independent of the routines which call them and thus might call themselves recursively. The executive should provide isolation and protection for such routines and facilitate the passing of parameters to them. This requirement for subroutine autonomy was intended to achieve speed in the running of library routines by eliminating the time for compilation or relocation and was intended to simplify the programming of such routines by minimizing the number of restrictions they would have to meet.

5. *Flexible Input/Output Services.* The executive should handle the details of all input/output operations. It should provide continuity for displays and keyboard-typewriter conversation. It should provide for the sharing of I/O devices like printers and magnetic tape that cannot be duplicated at every console. In so far as possible, it should leave formats and the interpretation of commands to the user's own programs.

DESIGN DECISIONS

Early in the design phase of the executive program a number of policy decisions were made which had a considerable effect on the character of the final program. In retrospect the most important were the following:

1. In order to meet the requirements for fast response, memory as well as time should be shared among the consoles. If the active part of the program for each console can be kept in core, the time required to switch between users is greatly reduced. With a small number of consoles, it appeared that the TX-2 memory was large enough so that it would often hold all of the active pieces of their programs, provided the pieces were of reasonable size. It was therefore decided that the executive should provide services that would encourage programmers to break large structures into small units. And to facilitate this sort of memory-sharing, it was decided that hardware for relocation and bounding should be added to the computer.

2. The system should provide program sharing so that memory sharing would operate efficiently. Large public routines, such as compilers, should be written as pure procedures so that they could be shared by all users. The TX-2 order code allows this kind of program to be written without any special difficulty. It was decided that the executive should incorporate features to facilitate the operation of pure procedures and that the hardware necessary to protect them should be installed.

3. The executive should simulate an apparent computer for each console. The requirements of the operation-oriented system could have been met by a highly specialized executive program, but such a design would not have satisfied the needs of the research projects that were already using TX-2. Their needs would, perhaps, have best been served by a time-sharing system that provided the entire facilities of the computer for each user in turn. Existing programs could then have been operated in the new system without significant changes. The large amount of time-dependent interaction between the TX-2 I/O system and the programs which use it would have required a very complex executive program in order to allow the existing I/O routines to operate. The simulation of an apparent computer similar, but not identical, to TX-2 seemed a reasonable compromise between these two requirements.

4. There should be no direct communication between the user and the executive. All input from the user should be passed through the executive to programs operating in his simulated computer and translated there. Commands to the executive would then be passed back from such a program to the executive. It appeared both unnecessary and undesirable to tie the system to any language conventions by building the conventions into the executive.

5. Insofar as possible, software features should be realized in programs operating in the simulated com-

puters. This decision allows the executive to be as simple as possible and permits expansion of the overall software structure without modifying the executive.

6. Compatibility between former TX-2 programs and programs that would operate in the simulated computer was not to be a requirement. The design of the simulated computer should be made to correspond to TX-2 whenever possible and reasonable. But it was expected that some changes would have to be made in all programs to accommodate the input/output characteristics of the executive and to take advantage of the storage allocation that it provides.

7. Changes in the hardware of the TX-2 computer were to be considered as legitimate variables in the design work. The computer engineering group was prepared to make reasonable modifications to the computer when such changes appeared to be the desirable and economical solutions to the software problems. Throughout the development of the executive program there was strong interaction between hardware and software designs and designers, and major changes were made in the computer to facilitate the APEX system. These include the addition of a file memory (a UNIVAC Fastrand Drum), hardware to trap the attempted execution of privileged instructions, and four memory-snatch channels to increase the efficiency of high speed I/O operations. The most significant change was the addition of a hardware system called SPAT (an acronym for **S**ymbolic **P**age **A**ddress **T**ransformation). SPAT, which has been in operation since January 1965, uses a 1,024 word thin-film memory[9] and high-speed transistor circuitry to make a three-level address transformation within a single TX-2 clock-pluse time (0.4μsec). This transformation makes available to the executive the advantages of paging, segmentation, and complete memory protection. It greatly reduces the overhead involved in sharing memory and programs.

BASIC CHARACTERISTICS OF THE APPARENT COMPUTER

As has already been said, the APEX executive program simulates an apparent computer for each console. The apparent computers may be viewed as somewhat restricted replicas of TX-2 augmented by special features provided through the executive program. The core storage for each apparent computer is bounded and segmented and is limited in total extent to approximately two-thirds of the TX-2 core capacity. The order code is that obtained by eliminating input/output and multiple-sequencing instructions from the TX-2 order code,[10,11] and then adding some executive calls to handle input/output, file maintenance, and allocation of storage in the apparent computer. The number of index registers is reduced to 15, and some restrictions are placed on the choice of "configurations" (which are used primarily to control operations on subwords). Unlike TX-2, the apparent computer is a single-sequence computer in the current version of the system (i.e., it has in effect only one program counter), but the hardware allows for future expansion to three sequences. In general, programs written for TX-2 will not operate in the apparent computer, and vice versa. However, programs which do not involve I/O operations may often be transferred with no change.

The storage structure of the apparent computer takes advantage of the SPAT address-transformation hardware that has been added to TX-2. The SPAT hardware (which is discussed in more detail in the last section of this paper) breaks core storage into pages of 256 registers, which are organized into books (segments) of up to 32 pages (8,192 registers). The 17-bit address of TX-2 allows 16 such books to be selected by the four highest order address bits. Since the total number of apparent addresses exceeds the available core, some of the books must always be incomplete or empty.

In the apparent computers realized by the APEX executive program, the user's programs and data are organized into files. A file is a contiguous group of registers, which must be some integral number of pages in length. It always has at least one name, which is known to the APEX file directory. Files may exceed one book in length, but they must begin at the start of a book, and no more than one file may occupy a book. Executive calls in the user's program determine which files are to appear in core at any one time. A file may be set up in a book specified by the directory, as is usually the case for program files, or it may be set up in an arbitrary book according to the requirements of a program which is to process it.

All files begin as working storage files with ephemeral names. When a program has finished writing information in such a file, it may give an executive call to assign a permanent name to the file. Once the file has been given a permanent name, it will remain in the file memory until it has been discarded by a call from the user's program. Thus, all data files that the user has had occasion to name will be retained from one session to the next. Files which receive only ephemeral names are discarded automatically according to simple rules. Executive calls are available that can give a file read-only protection, prohibit its use as a program file, and expand or shrink it by an integral number of pages.

A further feature, called auto-expandability, takes advantage of the potentialities for dynamic storage al-

location provided by the SPAT transformation. This feature allows files which tend to grow with time to make efficient use of core without putting the burden of boundary testing onto the programs that build them. Auto-expandability is accomplished by having the executive expand the file by one page whenever the program makes a reference to an address in the first page beyond the current boundary of the file.

The complex of files set up in the user's apparent memory at any one time is called a Map. A Map may be thought of pictorially as shown in Fig. 1 with a typical setup for a matrix routine. However, a Map may be equally well described as simply a list of names of files, together with the number of the book in which each is to appear in the Map. In Fig. 1, the dashed lines indicate the potential capacity of each book, while the solid lines indicate the actual core occupied by the

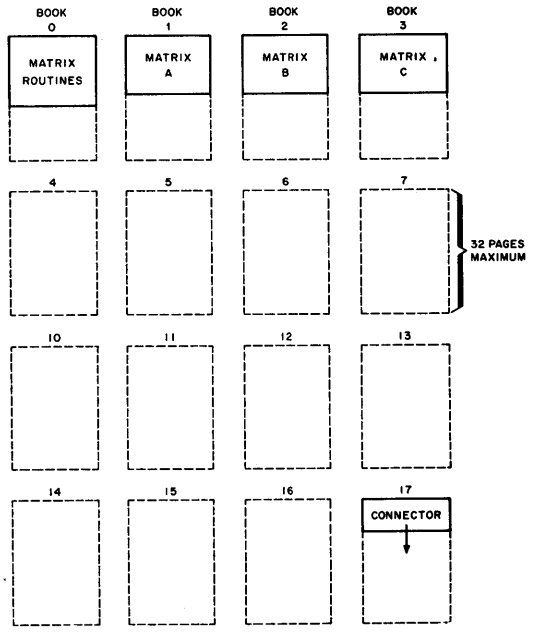


Figure 1. Map for matrix addition.

four files. More will be said about the contents of this Map after a short discussion of the way the APEX system handles library routines.

One of the principal design requirements for APEX was to provide a means whereby library routines (or arbitrary subroutines written by the user) could be called into core and operated without any conflict between the core-addressing requirements of the routine and the program that called it. This requirement is met by providing a fresh Map for the called routine. When a program wishes to call a library routine to be operated in a new Map it does so by issuing a Go Up call to the executive, passing along the name of the library routine as a parameter of the call. This process is called "going up" because the new Map is thought of as being put on top of the Map that contained the calling program. The reverse process, returning to a lower Map, is called "peeling back."

Since there are usually some parameters that must be passed to a library routine, a special file called the Connector is provided. It is common to all Maps and is used primarily for communication between them, but it may also be used for small amounts of working storage by the library routines, which are generally pure procedures. By convention, the first register of the Connector indicates the beginning of free storage in the file. Before issuing a Go Up call, the calling program normally stores a block of information into the free area of the Connector and moves the free storage pointer appropriately. This block contains a Peel Back call followed by the parameters to be passed to the library routine. The location of the Peel Back call and the name of the library routine are then given to the executive as parameters of the Go Up call. On receiving the Go Up call, the executive produces an entirely new Map that contains only the library routine and the Connector and then passes control to the library routine in the new Map. The routine is entered as though it had been called as a closed subroutine from a location just before the Peel Back call in the Connector. It then finds its parameters in the Connector, inspects them, and gives executive calls to set up such other files as it may need to carry out its mission.

Figure 1 shows the state of a Map in which a library routine for matrix addition has just finished its work. The Map began with only the file of matrix routines and Connector file set up. The addition routine found the names "A" and "B" as its input parameters, set up the files having those names in Books 1 and 2, set up an ephemeral file in Book 3 to receive the sum, computed the sum, and gave the output file the permanent name "C," which it found in the Connector as the name to be given to its output.

With its operations complete, the routine will now make a standard subroutine exit, transferring control to the Peel Back call, which will cause the executive to discard the new Map and return control to the calling program on the lower Map at the location just beyond the Go Up call. If the library routine was to have generated some output which was to be returned directly to the calling program, the calling parameters would have specified a location in the Connector into which the output would have been placed.

This way of handling library routines has a number of advantages. First, the library routine is written as an ordinary closed subroutine and is not itself concerned with going up or peeling back, unless it needs to call

another routine in the course of its operation. It may therefore be operated either by going up to a new Map or by setting it up in the same Map as the routine that calls it and then using it as an ordinary subroutine. The latter mode of operation has speed advantages, but is limited to situations where the programmer has determined that core assignments and index register usage are compatible. A second advantage comes about because the ability to change Maps is available not only to library routines, but also to arbitrary programs written by the user. The stack of Maps aids the programmer in putting together large, complicated structures, which may exceed both the real core and the core addressing capacities of the machine. However, he must keep in mind that changing Maps involves a bookkeeping overhead in the executive and will involve swapping memory (at disc speeds) if real core capacity is exceeded.

MORE SOPHISTICATED USES OF MAPS

Figure 1 showed a Map that resulted from the operation of a simple library routine. Now consider a more complex Map which illustrates the use of auto-expandable files and shows some advantages of segmentation in a large program. Figure 2 is a typical Map used by

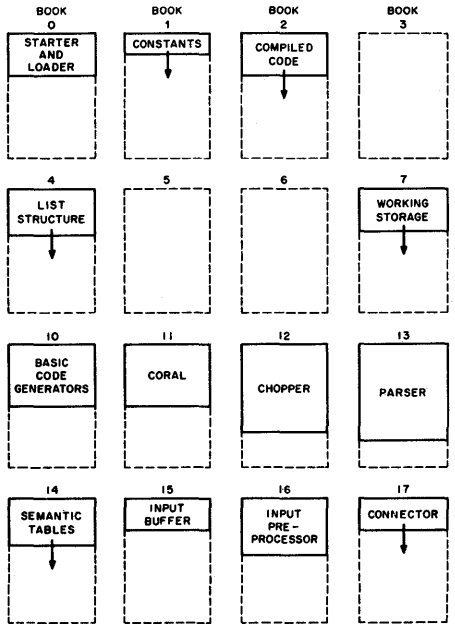


Figure 2. Map for a compiler.

a compiler realized within a compiler-compiler called VITAL* (Variable Initialized Translator for Algorithmic Languages), which is being developed to operate in the APEX system. The figure shows the Map for the compiler itself, which is made up of 12 files, five of which are auto-expandable, as indicated by the arrows extending downward from the rectangles that represent the files. The Map also contains the Connector, which is auto-expandable too. It is not in itself an essential part of some compilers; but it provides a way of entering and leaving the Map, and is therefore used extensively in translating and running programs in some languages.

In operation, the Input Pre-Processor (Book 16) builds a statement in the Working Storage file (Book 7) from characters obtained from the Input Buffer (Book 15). The Chopper (Book 12), using language definitions from the List Structure (Book 4), breaks up the statement into a string of operators, constants, and symbolic expressions. This string is stored in the List Structure using the CORAL subroutines in Book 11. The Parser (Book 13) then determines the grammatical structure of the statement string and prepares a generating string (also stored in the List Structure) which the Basic Code Generators (Book 10) and Loader (Book 0) can use to build the Compiled Code in Book 2.

The Input Pre-Processor, Chopped, Parser, and CORAL are independent of the particular language being translated. The definition of the language is determined by the Semantic Tables, Basic Code Generators, and List Structure. Since VITAL allows the definition of a language to be modified and augmented during the compilation, the complete VITAL structure includes another Map called the Meta-Compiler Map. This Map (not shown) is similar to the Compiler Map of Fig. 2, except that the Basic Code Generators and Semantic Tables are those for the meta-language. When the services of the Meta-Compiler are needed, it is entered by a Go Up call from the Compiler Map.

Figure 2 shows a program structure that exploits fully the segmentation of storage structure made available by the SPAT transformation in TX-2. While it would appear from the figure that there are three spare books in the Map these books are actually committed to providing for the larger files of Compiled Code and List Structure that would arise in the case of a larger program or a more complex language. If fewer than 16 books had been available, it would have been necessary to combine files, losing the advantages of auto-expandability in conserving core capacity and of flexibility in switching languages. If more than 16 books were available, greater flexibility could doubtless be achieved, but the bookkeeping costs to the executive would increase and could become burdensome in simple situations where only a few segments are needed.

It is often the case that when a program structure

*The VITAL compiler-compiler project was initiated by L. G. Roberts. It is now the responsibility of J. A. Feldman and C. D. Forgie.

pushes the limit on the number of books, it is also pushing the limit on available core. In that event the programmer has the choice of asking for a new Map or changing the contents of the current Map. VITAL uses the latter method when it must compile a program so large that the Compiled Code and the List Structure cannot be contained in the space shown in Fig. 2. Since the Compiled Code does not need to be in core while the List Structure is being built or modified, and since the Chopper, Parser, and CORAL do not need to be on hand during a simple compilation from a previously constructed List Structure, VITAL drops the unnecessary files from core when space is limited.

The APEX system also uses Maps to handle interrupts. The user's program may define special Maps called Ghost Maps, each of which is associated with a particular source of interrupts. These Maps are called Ghosts because most of the time they have no real existence. But when an interrupt occurs, the Ghost Map associated with the device that caused the interrupt appears on top of the stack of Maps and control is passed to it.

For interrupts caused by an illegal instruction, a boundary violation, or an I/O difficulty, a special HELP Ghost Map is provided; it automatically takes the user to a fixed, public routine that stabilizes his current I/O problems, if any, and then sets up a Basic Translator that allows him to call debugging or other routines to his aid. Note that in this situation, the HELP Ghost Map has suspended the operation of his program and has given him the full use of his apparent computer to work on his trouble. In addition, the stack of Maps has preserved all that was known about his program structure at the time the interrupt occurred. He may be able to fix the trouble and continue, discarding only the HELP structure at the top of his stack, or he may elect to start again at the bottom, forgetting everything about his old structure.

Another example of the use of Ghost Maps is in processing break-point interrupts. In each of its core memory words TX-2 has an extra bit called the metabit that is unaffected by ordinary instructions. A pseudo-I/O device in TX-2 will detect the occurrence of a set metabit on an instruction, a deferred address, or an operand and will produce an interrupt if the pseudo-device is turned on. This facility can be used for a variety of debugging purposes, and the APEX system provides executive services and library routines that make the facility available in the apparent computers. Figure 3 shows the Map stack that will appear in one such application.

The figure assumes that the user has logged in and

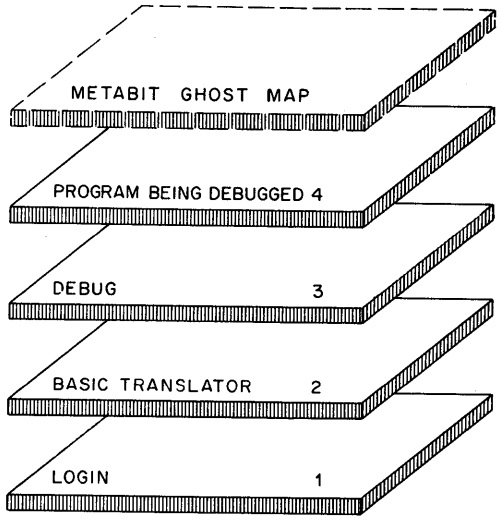


Figure 3. Map stack for break-point processing.

used the Basic Translator in Map 2 to call a program called DEBUG*, which will put break-point interrupts into his program. Map 3, the DEBUG Map, contains its own translator, which accepts commands from the user, allowing him to specify where he wishes to put break-points and what response he desires when they are detected. The DEBUG program calls into its own Map the program files in which the user wants to put break-points. DEBUG then tells the executive that it wants to use metabits in these files and says where they are to be placed. (The executive is involved in this operation because metabits do not exist on the file-memory drum and must be restored if the file is moved out of core and back in.) The program structure to be debugged may involve many files, but only those that are to have break-points need be set up at this time and given special metabit status.

DEBUG then gives an executive call to define a Metabit Ghost Map, to which control will be transferred when the pseudo-I/O device detects a set metabit. DEBUG defines this Map to contain the same files as the DEBUG Map itself, but specifies a different entrance point. DEBUG then issues a Go Up call that takes control to the program to be debugged (Map 4), which runs in its normal fashion until a set metabit is encountered. At that point the executive activates the Ghost Map and transfers control to it. DEBUG, running now in the Ghost Map, makes note of the break-point and responds to the user in the way he specified. It may take some action that he has previously specified, or it may halt operations and await his further commands.

*The DEBUG program is developing under the guidance of Prof. T. G. Stockham, Jr., of M.I.T.

In either case, a Peel Back call given in the Ghost Map can return control to the program being debugged, and let it continue to the next break-point. In Fig. 3 the program being debugged is shown as occupying a single Map (Map 4), but the system works equally well when a multiple-Map structure is being debugged.

INPUT AND OUTPUT FOR THE APPARENT COMPUTERS

The basic console that is available to the user of APEX has a keyboard, an output typewriter, a display scope of one sort or another, a light-pen, and a few push-button switches. Since APEX is an experimental system, there are differences in the makeup of the equipment available at the various consoles. Some consoles have no output typewriters, others have no display scope; a RAND tablet[12] will be available on one console, and some will have a connector with 36 output and 36 input wires to which a user can attach special equipment of his own.

The keyboard and the output typewriter operate as a fully duplexed system with the executive handling the typing directly. The executive places input characters into a buffer file accessible to the user's program. As each character passes through the executive it is checked against a table of terminators supplied by the user's program. If the user's program is inactive, or if a Ghost-Map interruption mode has been specified, the executive will take appropriate action when a terminator is encountered. Any group of characters may be defined as terminators. The typing back of keyboard inputs may be suppressed when the keyboard is used as an input device for the display scope.

Hardware and software techniques for the production of displays on cathode ray tubes are areas in which much work has been done on TX-2 in the past. This work has been characterized by an unusually tight coupling between the display and the computer. In the past,[2,3,13] the picture typically has not been generated from a simple table of the points and lines to be displayed; it has been produced "on the fly" as the program works its way through a complicated ring structure that contains information not only about the points and lines to be displayed, but also about relations between the objects that the picture represents.[14] One of the most difficult requirements placed on the design of the APEX system was that it should allow display work of this type to continue in an environment of time-sharing.[15] The present design appears to be a reasonable first compromise between time-sharing and tight coupling of display and computer, but it may well have to be changed in the not-to-distant future as new techniques and new requirements develop.

The display equipment that is currently available consists of scopes at three consoles. All three are driven in parallel from a shared vector-and-curve generator[16] that gets its inputs directly from the computer memory. Analog integrators are used to generate lines, circles, and parabolas from the digital information obtained from the computer. The analog deflection signals are sent to all scopes, but only the scope for which the display information is intended receives an intensification signal. One console has a Charactron tube, but the others have simple cathode ray tubes which require characters to be generated by the vector-and-curve generator.

The information that the user's program wants the executive to display is contained in a ring-structured file that the user's program may set up in its own Map for inspection. However, the user's program cannot write in the file. When changes are wanted, it must request the executive to make them. This restriction was imposed because the executive's display routine may run wild if asked to follow an ill-formed ring. The structure can contain only one type of information about relations between parts of the picture: it can show what parts are subordinate to others. This capability is very important to the kind of display work that has been carried on in the past. In such work it is important that the user's program be able to request that the executive modify a part of the picture, including all subordinate parts, without rebuilding the whole file. If the user wants his program to keep track of other types of relations, the program must keep the extra information in private files of its own.

The rings in the display file specify the order in which the elements of the picture are to be displayed, and the display routine in the executive works its way through the complex, transmitting the relevant data to the display-generation hardware. A single pass around the ring defines a "frame," and the display-generator is time-shared among consoles on a frame-by-frame basis. The display routine maintains the display on the user's console even though his program is inactive because of time-sharing or because it is waiting for an input. A "push-to-see" button is used by some types of display programs to keep down the load that display maintenance would otherwise place on the system.

The task of processing signals from the light-pen is closely related to the task of generating displays since the pen is the principle way in which the user communicates to his program information about the ring structure from which the display is generated. The light-pen may be used in two modes—pointing and tracking. In both modes the executive maintains a

complete record of all light-pen returns in a buffer in the file of information being displayed. A light-pen return while in the pointing mode causes the executive to place in the buffer information from which the user's program can calculate both the pen position and the place in the ring structure from which the element seen by the pen was generated. In many light-pen applications it is necessary to associate the pushing of a button or the striking of a key with the pointing operation. This association is handled by the executive, and the associated character code is placed in the light-pen buffer.

In the tracking mode, the executive displays a tracking cross every 30 msec. If the pen sees any part of the cross, the executive moves the cross to center it in the field of the pen and the location of the center of the cross is placed in the buffer. Smoothing and extrapolation are done in the tracking routine to achieve good "writing" characteristics for the pen. The processing of light-pen signals is a high priority task for the executive since fast responses are essential, especially in the tracking mode.

The second category of input/output equipment that is available to the apparent computers consists of devices that must be shared by them. It includes magnetic tape and paper tape equipment, units for analog inputs and outputs, and a high-speed Xerographic printer. This class of equipment has an unusual status in the APEX system because most of the consoles will be in the computer room, and the users at those consoles will have easy access to the common, shared devices. While these shared devices posed a number of problems to the designers of the executive, the solutions are too specialized to the nature of TX-2 to warrant further discussion here.

CONTINUITY OF OPERATION IN THE APEX SYSTEM

The user for whom the on-line, operation-oriented system is designed is a man at work on the solution to a problem. His problem will rarely be solved in a single session on the computer, and so one of the major tasks of the executive system is to remember his programs, files of data, and other quantities that he may find useful in maintaining the continuity of his work from one session to another. A portion of the executive maintains a private directory for each user as well as a public directory which is shared by all users. A number of calls are available to the apparent console computer to allow a user's program to insert items into his private directory and to inquire about these items and about items in the public directory.

Items remembered through a directory are identified by names, which are strings of up to 50 characters. The characters are restricted to Roman capital letters, Aarbic numerals, and period. The directory itself is a ring structure arranged in the form of a tree to give a logarithmic search-time for names. When a name is entered into the directory, a unique block is created for it in the list structure, and the pointer to that name block is used as a compact and more efficient substitute for the original string of characters. Remembering an item in the directory involves two calls. The first asks the executive to accept a string of characters and return the resulting name pointer. The second call uses the pointer plus the necessary defining information to establish an association between the name and the item to be remembered.

The directory can keep track of the following kinds of items, either directly or by way of the file memory:

1. *Files.* A file is any contiguous group of memory registers. As was explained in connection with the storage structure of the apparent computer, the directory contains information showing whether the file is to be protected against writing operations, whether it is to be protected against being executed as a program, in which book it is normally to be set up (if it has a preferred book), and whether it contains program or data. If it contains data, the type of data is shown only by the internal format of the file which is left to the author of the program who created it. The executive system must serve such a wide variety of programmers that any attempt to reach an agreement on a standard classification of data types seemed hopeless.

2. *Scalars.* A scalar is a single-register quantity remembered in the directory itself. Scalars are useful for allowing the user to remember single numbers that are not part of some larger array of data. They are also useful in allowing public routines to use the user's directory to remember certain parameters from one usage to the next. Thus, for example, a compiler (which is normally a pure procedure) can remember the name of the last program it was compiling for each user.

3. *Entrances.* An entrance is a number associated with a file. A program file may contain a number of related routines which perform different functions. Entrances can then be used to call these different routines by entering the program file at different locations. If a Go Up call is given to the executive and the parameter on that call specifies an entrance, the file will be set up (if it is a program file), and control will be transferred to the location specified by the entrance. Entrances may also be used with files of data. For example, an entrance may identify the start of a particular ring in a list structure.

4. *References to public names.* A reference to a public name is a device for allowing a user to substitute a name of his own choosing for a public name that is unsatisfactory to him.

5. *File groups.* The file group, as the name implies, is merely a collection of related files. Its existence in the directory allows the group of files to be brought in from the drum and set up in memory by means of a single call. For example, consider the situation when a general translator is used to translate a particular problem-oriented language. In addition to the file containing the translator itself, a file of definitions for the particular language and a file of working storage must be set up before any translating can begin. Treating the files as a group allows the executive to get them all into core and set up before any attempt is made to run the program.

The directory not only maintains relations between names and things; it also maintains relations between names. A Synonym call to the executive allows the user's program to indicate that a particular item in the user's directory is to have a second, synonymous name. A name may have any number of synonyms. They are added one at a time by the Synonym call and may be removed one at a time by the Undefine call. If all of the names have been removed by the Undefine call, the item itself will be forgotten by the directory and destroyed. There is also a Drop call that allows all the names to be undefined and the entity destroyed with a single call. Synonyms are useful for abbreviation and for substitution of parameters. They are handled by the executive rather than left to particular translators because it is felt that the user regards them as language-independent relationships that should endure when he switches from one translator to another.

As a convenience to the user, and for his protection against system failures, his private directory and all his files that existed at the start of a session are retained until the end. If he ends the session in the normal way by logging out, his starting directory and any obsolete files are discarded. If the session has gone badly, he may elect to abort rather than log out. In that case his current directory and any new files he has created are discarded, and his starting directory and its files are saved for the next session. In the event of system failure his session is automatically aborted. In a long session he may update his protection by logging out and back in without incurring any special penalty.

THE EXECUTIVE AND THE SPAT HARDWARE

As was stated at the outset, the planning for the system involved continual interaction between the designers of the software and of the hardware. This section will consider those aspects of the hardware-software system which, while elaborated in the environment of TX-2, may have application elsewhere. These aspects have to do with the SPAT address-transformation hardware and its application in relocating, bounding, and protecting the user's programs and in facilitating the operations of the executive.

The SPAT transformation applies to all instruction, deferred address, and operand references to memory. It is effected in three stages. The first accomplishes the switching of memory between the user and the executive. The second handles relocation, bounding, and memory protection. The third gives memory paging.

In addition, because TX-2 is a multiple-sequence computer (i.e., it has in effect many program counters),[11] the transformation takes into consideration which sequence (i.e., which "program counter") is providing the location for the current instruction. A part of the computer called the Sequence Selector handles the switching of program counters on demand according to a built-in priority rule. Most of the program counters are tied to I/O devices or internal sources of interrupts and have priorities based on the speeds of the devices and the functions of the interrupts. A few have no such associations and are used for general computation. The flag bits that demand service for these latter sequences are entirely under program control.

In the SPAT system, 29 of the 33 sequences are grouped together and given the same transformation. This grouping is indicated schematically in the box labeled Sequence Selector in the upper left portion of Fig. 4. When an instruction is executed in any of these sequences all addresses are transformed according to the information read out of the Executive Shelf in the Location and Boundary Memory. Of the remaining four sequences, the one called Startover is peculiar and is not relevant to the executive system. It is not subjected to the SPAT transformation. The other three sequences are given individual Shelves in LBM and are treated by the computer as user's sequences; i.e., privileged instructions are prohibited. The present version of the system uses only one of these three sequences; the other two are treated as spares.

Since a sequence change in TX-2 is accomplished in a time comparable to a single memory cycle, this first stage of the SPAT transformation allows very fast switching between user and executive. As will become apparent presently, a Shelf in LBM corresponds to a complete Map of core memory. So the change of sequences that occurs in response to an I/O event or an interrupt automatically switches the address transforma-

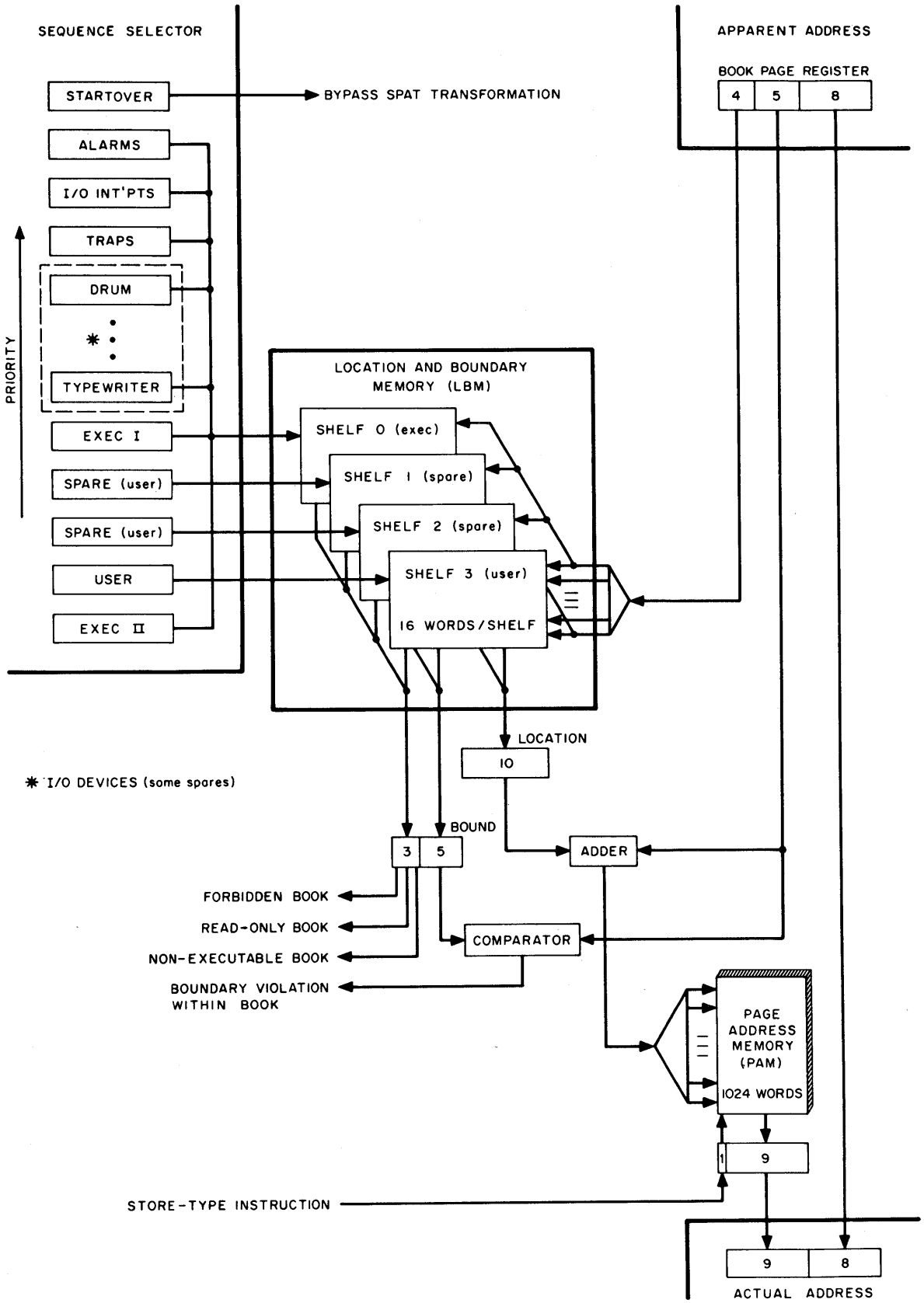


Figure 4. TX-2 address transformation hardware (SPAT).

tion from the user's Map to the executive Map.

There are executive sequences with both higher and lower priority than the user sequence. The sequences of higher priority handle I/O and interrupts, and the executive does its bookkeeping and scheduling in the sequence labeled Exec I in Fig. 4. The lowest priority sequence (Exec II) handles the user's calls to the executive. A special timer measures the time spent actually executing instructions in the User and Exec II sequences, and this measurement is used instead of real time to trigger a switch of users for time-sharing.

The second stage of the SPAT transformation is governed by information that the executive has previously stored in the Location and Boundary Memory (LBM). The information is automatically used to perform relocation, bounding, and file protection during every reference to core memory. The input to this stage consists of the nine most significant bits of the apparent address, which is illustrated in the upper right corner of Fig. 4. The four most significant bits (the book number) are used to select one of the 16 registers in the LBM shelf chosen by the first stage. The five-bit page number in the apparent address is tested against a five-bit boundary value stored in part of the selected register of LBM, and a Boundary Violation signal is generated if the address is outside the assigned range for the book. There are also three control bits in the LBM word. They can be set by the executive to prevent the user's program from writing in the book, executing an instruction in the book, or making any reference whatever to registers in the book. This last bit, which indicates a Forbidden Book, is used to indicate an unassigned book and to indicate a book that has been assigned core pages but is waiting for data to be transferred from the file memory. In the latter cases, the executive will allow the user's program to run in the absence of the file, but if his program refers to the file, the Forbidden Book bit is interpreted as a signal to switch to another user if any others are ready. The Read-Only control bit is ignored in the Executive Shelf. This feature allows files like display buffers to appear in both the user's Map and the executive Map, appearing to the user as read-only but appearing to the executive as writable.

Besides providing the control bits and the bound, the second stage of SPAT performs the relocation that provides segmentation into books. The selected register of LBM contains a location (i.e., a base address) to which SPAT adds the five-bit page number from the apparent address.

Observe that this sum could have been used as the final output of the whole transformation. With the last eight bits of the apparent address appended to it, the sum is a relocated address that has been checked for boundary violations and could in principle be used directly to address real memory.

But instead of being used directly, the sum of the location and the apparent page number is used as the input to the third stage of the transformation. It is used to select one of the 1,024 registers of Page Address Memory (PAM). In that register the executive has previously stored a 9-bit number, called the actual page number. The last 8 bits of the apparent address are appended to this number to form the 17-bit word that is finally used to address the real core memory.

Conceptually, the third stage of the transformation divides the real memory into pages, which are blocks of 256 consecutive registers. By storing the appropriate information into PAM, the executive can define an arbitrary transformation between the page numbers that enter the third stage and the page numbers that go out to address the real memory. In particular, pages that are in fact widely separated in real core can be made to appear contiguous, simply by changing the contents of the corresponding registers in PAM.

A word in Page Address Memory contains 12 bits, 3 more than are used to hold the actual page number. Only one of the three is used in the present system. It is set automatically whenever a store-type instruction refers to a register in the page selected by the transformation. By inspecting these bits for all the pages of a file, the executive program can determine whether the contents of the file could have been altered. This feature can be used to avoid copying back onto the drum a file that clearly has not been changed since it was copied into core.

The main purpose of SPAT is to speed up the response of the system by (1) making it possible for the executive to come closer to the ideal goal of keeping the active part of each user's program in core and ready to run, and (2) facilitating changes between users and, especially, between user and executive. The advantages of providing hardware for relocation and bounding when more than one program is to be in core are by now too familiar to require further comment. PAM was included to reduce the amount of time that must be spent on storage allocation when the goal is to keep as many programs as possible in core. Given hardware for relocating 16 files independently, each at its own base address, the problem of storage allocation is primarily that of creating a block of contiguous storage large enough to accommodate a file that must be brought into core. With SPAT, the task of allocating space in real core is replaced by the task of allocating

space in PAM. From the point of view of speed, there is an obvious advantage in reducing by a factor of 256 the number of registers that have to be changed when files must be moved to consolidate empty space into larger blocks. There is an added advantage in the fact that the number of registers in PAM is nearly three times as large as the number of real pages in the TX-2 memory. This factor of three greatly increases the probability that an adequate block of empty space will be found without making any moves. Segmentation increases the probability further since it puts a limit of 32 on the number of contiguous PAM registers that must be found to satisfy an allocation request. Assuming that PAM is already set up correctly, the change of storage references required in switching from one user to another is accomplished by storing into the User Shelf of LBM the 16 words appropriate to the new user's current Map. Switching between the user's Map and the executive's Map is even quicker, since a change of sequence automatically selects the proper Shelf. Such rapid switching is highly desirable in a system of this sort because of the large number of I/O interrupts and executive calls to be processed.

The SPAT hardware could provide most of the control information needed for automatic page-turning,[17] but the auxiliary memory facilities in TX-2 are too slow to justify this feature. While the hardware treats a file as a collection of independent pages in core, and the executive program handles it by pages in the file memory, the executive does not now keep track of the status of the individual pages which make up a file. But given a suitable auxiliary memory, the system could be augmented without great difficulty to include page-turning. In that case, the utilization of the two spare User Shelves in LBM would provide the user's program with nearly random access to almost 390,000 registers of apparent core memory.

CURRENT STATUS

At this writing, most of the executive program is either operational or in the debugging stage. Parts which are not yet ready for debugging are those having to do with file groups and the handling of some of the shared I/O equipment. Further work is also needed to replace the present primitive strategies for time-and memory-sharing with more sophisticated ones.

ACKNOWLEDGMENTS

The system described in this paper has resulted from the efforts of a large number of people both in programming and in hardware development. The original design for the system was developed by a group made up of D. B. Yntema, L. G. Roberts, J. E. K. Smith (now at the University of Michigan), and the author. The idea for the SPAT trnasformation was contributed by J. M. Frankovich, and the design of it and the other modifications to TX-2 were his responsibility. Much of the detailed design of the executive program was done by C. K. McElwain and A. N. Stowe. The program makes extensive use of the CORAL language developed under the direction of L. G. Roberts. The author is grateful to D. B. Yntema for his valuable editorial assistance.

REFERENCES

1. W. A. CLAR, "The Lincoln TX-2 Computer Development," *Proc. Western Joint Computer Conf.* (1957), pp. 143 - 45.
2. I. E. SUTHERLAND, "Sketchpad: A Man-Machine Graphical Communication System," *AFIPS Conf. Proc* (1963 *Spring Joint Computer Conf.*) vol. 23, pp. 329 - 46.
3. T. E. JONSON, "Sketchpad III: A Computer Program for Drawing in Three Dimensions," ibid, pp. 347 - 53.
4. B. GOLD, "Computer Program for Pitch Extraction," *J. Acoust. Soc. of Am.*, vol. 34, no. 7, pp. 916 - 21 (July 1962).
5. C. M. RADER, "A Speech Compression Simulation Compiler," presented at 69th Meeting of Acoust. Soc. of Am., Washington, D.C. (June 1965).
6. J. W. and C. D. FORGIE, "A Computer Program for Recognizing the English Fricative Consonants /f/ and /θ/." presented at 4th Int. Cong. on Acoustics, Copenhagen (Aug. 1962).
7. L. D. EARNEST, "Machine Recognition of Cursive Writing," *Information Processing* 1962, *Proc. of IFIP Congress* 62, North-Holland Publishing Co., Amsterdam, 1963, pp. 462 - 66.
8. D. B. YNTEMA, "The Operation-Oriented Approach to On-Line Interaction between Scientists and Computers," presented at meeting IEEE Prof. Gp. on Human Factors in Electronics, Boston (May 1965).
9. J. I. RAFFEL, et al, "The FX-1 Magnetic Film Memory," Lincoln Laboratory Technical Report #278 (Aug. 1962).
10. J. M. FRANOVIC and H. P. PETERSON, A Functional Description of the Lincoln TX-2 Computer," *Proc. Western Joint Computer Conf.* (1957), pp. 146 - 55.

11. J. W. FORGIE, "The Lincoln TX-2 Input-Output System," *Proc. Western Joint Computer Conf.* (1957), pp. 156 - 60.
12. M. R. DAVIS and T. O. ELLIS, "The RAND Tablet, A Man-Machine Communication Device," *AFIPS Conf. Proc.* (1964 *Fall Joint Computer Conf.*), vol 28, pt. 1, pp. 325 - 32.
13. L. G. ROBERTS, "Machine Perception of Three-Dimensional Solids," Lincoln Laboratory Technical Report #315 (May 1963).
14. L. G. ROBERTS, "Graphical Communication and Control Languages," *Information Systems Sciences: Proc. of the Second Congress,* Spartan Books, Washington, D.C. (to be published).
15. L. G. ROBERTS, "Graphical Communication in a Time-Sharing Environment," presented at 1965 IFIPS Congress, New York (to be published).
16. T. E. JONSON, "Analog Display Generators," Lincoln Laboratory Technical Report #398 (to be published).
17. T. KILBURN, et al, "One-Level Storage System," *IRE Trans. on Electronic Computers,* vol. EC-11, no. 2, pp. 223 - 35 (Apr. 1962).

Interactive machine-language programming*

BUTLER W. LAMPSON

University of California, Berkeley

INTRODUCTION

The problems of machine language programming, in the broad sense of coding in which it is possible to write each instruction out explicitly, have been curiously neglected in the literature. There are still many problems which must be coded in the hardware language of the computer on which they are to run, either because of stringent time and space requirements or because no suitable higher level language is available.

It is a sad fact, however, that a large number of these problems never run at all because of the inordinate amount of effort required to write and debug machine language programs. On those that are undertaken in spite of this obstacle, a great deal of time is wasted in struggles between programmer and computer which might be avoided if the proper systems were available. Some of the necessary components of these systems, both hardware and software, have been developed and intensively used at a few installations. To most programmers, however, they remain as unfamiliar as other tools which are presented for the first time below.

In the former category fall the most important features of a good assembler:[1,2] macro-instructions implemented by character substitution, conditional assembly instructions, and reasonably free linking of independently assembled programs. The basic components of a debugging system are also known but relatively unfamiliar.[3,4] For these the essential prerequisite is an *interactive* environment, in which the power of the computer is available at a console for long periods of time. The batch processing mode in which large systems are operated today of course precludes interaction, but programs for small machines are normally debugged in this way, and as time-sharing becomes more widespread the interactive environment will become common.

It is clear that interactive debugging systems must have abilities very different from those of off-line systems. Large volumes of output are intolerable, so that dumps and traces are to be avoided at all costs. To take the place of dumps, selective examination and alteration of memory locations are provided. Traces give way to breakpoints, which cause control to return to the system at selected instructions. It is also essential to escape from the switches-and-lights console debugging common on small machines without adequate software. To this end, type-in and type-out of information must be symbolic rather than octal where this is convenient. The goal, which can be very nearly achieved, is to make the symbolic representation of an instruction produced by the system identical to the original symbolic written by the user. The emphasis is on convenience to the user and rapidity of communication.

The combination of an assembler and a debugger of this kind is a powerful one which can reduce by a factor of perhaps five the time required to write and debug a machine language program. A full system for interactive machine language programming (IMP), however, can do much more and, if properly designed, need not be more difficult to implement. The basic ideas behind this system are these:

1. Complete integration of the assembler and the debugging system, so that all input goes through the same processor. Much redundant coding is thus eliminated, together with one of two different languages serving the same purpose: to specify instructions in symbolic form. This concept requires that code be assembled directly into core—or into a core image on secondary

*This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract SD-185.

storage. Relocatable output and relocatable loaders are thereby done away with. (A remark on terminology: It will be convenient in the sequel to speak of the "assembler" and the "debugger" in the IMP system. These terms should be understood in the light of the foregoing: different parts of the same language are being referred to, rather than distinct languages.)

2. Commands for editing the symbolic source program. The edit commands simultaneously modify the binary program in core and the symbolic on secondary storage. Corrections made during debugging are thus automatically incorporated into the symbolic, and the labor of keeping the latter current is almost eliminated.
3. A powerful string-handling capability in the assembler, which makes it quite easy to write macros for compiling algebraic expressions, to take a popular example, which can be handled in a few other systems but rather clumsily. The point is not that one wants to write such macros, but that in particular applications one may want macros of a similar degree of complexity.

These matters are discussed in more detail below. We consider the assembler first and then the debugger since the command language of the latter makes heavy use of the assembler's features.

Before beginning the discussion it may be well to describe briefly the machine on which this system is implemented. It is a Scientific Data Systems 930, a 2-microsecond, single-address computer with indirect addressing and one index register. Our system includes a drum which is large enough to hold for each user all the symbolic for a program being debugged, together with the system, a core image of the program and some tables. Backup storage of at least this size is essential for the editing features of the IMP system. The rest of the system could be implemented after a fashion with tapes.

THE BASIC ASSEMBLER

The input format of the assembler was originated on the TX-O at M.I.T. It has been adopted by DEC for most of its machines, but is unknown or unpopular elsewhere in the industry. Although it looks strange at first, it has substantial advantages in terms of simplicity, both for the user and for the system. The latter is a nonnegligible consideration, equally often ignored and overemphasized.

The basic idea is that the assembler processes each line of input as an *expression* (unless it is a directive or macro call).[5] The expression is evaluated and the value is put into core at the word addressed by the location counter, after which the location counter is advanced by 1. Expressions are made up of *operands*, which may be symbols, constants, numeric or alphanumeric and parenthesized subexpressions; and *operators*. Available operators are +, —, *, /, .AND, .OR, .NOT with their usual meaning and precedence; .E (equals), .G (greater), .GE, .L, .LE, .NE, which are binary operators with precedence less than +, and yield 1 or 0 depending on whether the indicated relation holds between the operands or not; and #, a unary operator with lowest precedence which causes its operand to be taken as a literal. This means that it is assigned a storage location, which is the same as the location assigned to other literals with the same value, and the address of this location is the value of the literal. Blanks have the following significance: Any string of blanks not at the beginning or end of an expression is taken as a single plus sign. An expression is terminated by carriage return or semicolon. Several instructions may therefore be written on one physical line. This trivial feature proves in practice to have significant advantages.

It is not immediately clear how instructions are conveniently written as expressions, and in fact the scheme used depends on the fact that the object machine is a single-address, word-oriented computer with a reasonable number of modifiers in a single instruction. It would work on the PDP-6, but not on the IBM 7030.

The idea is simple: all operation code mnemonics are predefined symbols with values equal to the octal encodings of the instructions. On the SDS 930, for instance, LDA (load A) is defined as 7600000 (all numbers are in octal). The expression LDA+200 then evaluates to 7600200. When the convention about spaces is invoked, the expression

```
LDA 200
```

evaluates to the same thing, which is just the instruction we expect from this symbolic line in a conventional assembler.

Modifiers are handled in the same spirit. In the 24 bit word of the 930 there is an index bit, which is the second from the left, and an indirect bit, which is the tenth. With the predefined symbols

```
I=40000
X=20000000
```

the expression

```
LDA I 200 X
```

evaluates to 27640200. In more conventional form it would look like this:

```
LDA* 200,2
```

There is little to choose between them for brevity or

clarity. Note that the order of the terms in the expression is arbitrary.

The greatest advantages of the uniform use of expressions accrue to the assembler, but the programmer gains a good deal of flexibility. Examples will readily occur to the reader.

Using this convention the implementation of the basic assembler is very simple. Essentially all that is required is an expression analyzer and evaluator, which will not run to more than three or four hundred instructions on any machine. Because all assembly is into core, there is no such thing as relocatability.

Two rather conventional methods are provided for defining symbols. A symbol appearing at the beginning of a line and followed by a comma is defined to be the current value of the location counter. Such a symbol may not be redefined. In addition, a line such as

```
SYM=4600
```

defines SYM. Any earlier definition is simply overridden. The right side may of course be any expression which can be evaluated.

The special symbol . refers to the location counter. It may appear on the left of a = sign. Thus, the line

```
A,      .=.  40
```

is equivalent to

```
A       BSS 40
```

in a conventional assembler.

Note that the first punctuation character in a line of input to the assembler must be comma or space. The character . is not a punctuation character, but behaves exactly like a letter. Symbols reserved by the system begin with dot ordinarily. For convenience in forming negative addresses, the symbol .. is provided with a permanent value such that ..—1 is —1 truncated to the address field. On the 930, a two's complement machine with a 14 bit address field, .. is 40000.

Strings of characters encoded in ASCII may be written surrounded by single or double quotes, ' ' or " ". If the string is less than 4 characters in length, it is equivalent to the number obtained by left-justifying it in a 24-bit word. Otherwise, it must appear alone on a line and generates enough words to accommodate all its characters. Strings in single quotes are scanned for : and & (see below); those in double quotes are taken literally.

The characters space * signal a comment, which is ignored up to the next carriage return. An initial * also has this effect.

There remains one point about the basic assembler which is crucially important to the implementation: the treatment of undefined symbols. When an expression is encountered during assembly, there is no guarantee that it can be evaluated, since all the symbols in it may not be defined. This is the reason why most assemblers are two pass: the first pass serves to define the symbols. The increase in speed obtained by looking at the symbolic only once is so great, however, that it is worth a good deal of trouble. Even if every expression contains an undefined symbol on the first pass, it still takes only one-fifth as long to evaluate the already analyzed expressions as to read the input again, and this for a program with no macros. The assembler therefore keeps track of undefined expressions explicitly.

There is a general way of doing this, in which the undefined expression, translated for convenience into reverse Polish, is added to a list of such expressions, together with the address of the word it is to occupy. At suitable intervals this list is scanned and all the newly defined expressions are evaluated and inserted in the proper locations. For complex expressions there is no avoiding some such mechanism, and it has the advantage of simplicity. It is, however, wasteful of storage and also of time, since an expression may be examined many times while it is on the list before it can be evaluated. One important case can be treated much more efficiently, and this is the case of an instruction with an undefined address, which includes perhaps 90% of the occurrences of undefined expressions.

For example, when the assembler sees this code:

```
X,  BRU A   *BRANCH UNCONDITIONAL
    LDA B
A,  STA C
```

the instruction at X has an undefined address which becomes defined when the label A is encountered. This situation can be kept track of by putting in the symbol table entry for A the location of the first word containing A as an address. In the address of this word we put the location of the second such word, and so build a list through all the words containing the undefined symbol A as an address. The list is terminated by making the address field point to itself. When the symbol is defined we simply run down the chain and fill in the proper value. This scheme will work as long as the address field contains only A, since there is then no other information which must be preserved. Note that no storage is wasted and that when A is defined the correct address can be filled in very quickly.

STRINGS AND MACROS

The description of the basic assembler is now complete, except for a few nonessential details, and we turn to the macro and string handling facility. There is a uniform method for delimiting strings of characters, which may be illustrated by the assignment of such a string as the value of a symbol:

```
A = <B,(C,D),E,F>
```

In order to describe the result of using A after this assignment, we introduce a distinction between the appearance of a symbol in a *literal* and in a *normal* context.

A symbol inside string brackets < > or single quotes or in a macro argument is in a literal context; all other contexts but one are normal. In a normal context, the value of the symbol, whether a string or a number, is substituted for the symbol. In a literal context, on the other hand, the characters of the symbol are passed on unaltered. The case of a symbol on the left side of an assignment is an exceptional one; such a symbol is of course not normally evaluated.

To permit the value of a symbol to be obtained in a literal context, the convention is introduced that a colon preceding the symbol causes it to be evaluated if the colon is at the top level of parentheses, brackets, and quotes. If its value is a string, the characters of the string replace the symbol; if it is a number the shortest string of digits which can represent the number in the prevailing radix replaces the symbol. Colon in a normal context is illegal.

For convenience in delimiting string names a second colon may follow a name preceded by a colon. This second colon serves only to delimit the name and is otherwise ignored. Thus if

AB = <XYZ>

then

<:AB> = <XYZ> and <:AB:CD> = <XYZCD>

There are times when it is desirable to force evaluation of a symbol in a normal context when it would normally pass unevaluated. The character & preceding the symbol has this effect; it is exactly like : except that it acts only in a normal context. Continuing the previous example:

VW&AB = VWXYZ and

&AB = 12 is equivalent to XYZ = 12.

A string may be thought of as having two kinds of structure:

1. It is composed of a sequence of characters.
2. It is composed of a sequence of substrings delimited by commas not enclosed in parentheses, brackets, or quotes.

With reference to the first structure, a single character may be selected by a subscript enclosed in brackets. Referring to the string assigned to A, we note that

A[2] is <,>, A[6] is <D>, and A[7] is <)>.

By an obvious extension of this notation,

A[3,7] is <(C,D)> and A[9,11] is <E,F>.

Subscripts which reference the substring structure are enclosed in parentheses. Thus

A(1) = <B> and A(2) = <C,D>.

Note that a single pair of parentheses surrounding a substring is removed. Subscripting may be iterated:

A(2)(2) = <D>.

Subscripting is applied only to a string-valued symbol which is in a normal context or is evaluated by a colon. Subscripting of a name on the left side of an assignment forces it to be evaluated even if it is not preceded by a colon.

Two operations, .L and .LC, determine respectively the number of substrings and the number of characters in their arguments. Thus

.L(A) = 4, .L(A(2)) = 2 and .LC(A) = 11.

Having dealt with the general machinery for handling strings, we now turn to the slight refinement which adds macros with arguments to the system. This takes the form of a modification to the ordinary line assigning a string to a symbol, which permits an argument string to be specified. Thus

```
STORE <ARG> =
<.RPT.FOR T = 1, .L(ARG(2)),1
<ST&ARG(1)  ARG(2)(T)>>
```

defines a macro with two arguments, the first a string which, when appended to <ST>, creates a store instruction, and the second a list of locations to be stored into. Whenever STORE is used, the string of characters beginning with the first following nonblank character and ending with a line delimiter or unmatched right parenthesis is made the value of ARG. The string which is the value of STORE is then substituted for it as usual.

STORE might be called with

```
STORE A,(S1,S2,S3)
```

which is, because of the definition, equivalent to

```
.RPT.FOR T = 1,3,1
<STA <S1,S2,S3> (T)>
```

To complete the expansion we must consider the .RPT directive which has been used above. This directive causes the string which follows to be scanned repeatedly. It takes one of two forms:

1. .RPT N <...>

which causes N repetitions, or

2. .RPT.FOR J = nl,n2,n3 <...>

which causes (n2—nl)/n3 + 1 repetitions with J initially set to nl, and then incremented by n3 until it exceeds n2. Zero repetitions are possible. The n3 may be elided if it is 1.

The STORE macro call above may now be seen to expand into

```
STA S1
STA S2
STA S3
```

We illustrate with two further examples. The first is a generalized MOVE macro which takes as its arguments a sequence of pairs of lists. The first list of each

pair specifies the locations to load from, while the second gives the corresponding locations to store into. A list may of course have only one element.

```
        MOVE <ARG> =
<.RPT.FOR S1 = 1, .L(ARG),2
*THIS LINE STEPS THROUGH THE PAIRS OF
  LISTS
<.RPT.FOR S2 = 1, .L(ARG(S1))
*THIS LINE STEPS THROUGH THE ELEMENTS
  OF ONE PAIR OF LISTS

< LDA ARG(S1)(S2)
< STA ARG(S1 + 1)(S2) >>>
```

thus

```
        MOVE A,B,C,D
```

becomes

```
        LDA   A
        STA   B
        LDA   C
        STA   D
```

So does

```
        MOVE (A,C),(B,D)
```

Suppose that we have some two-word data structures to manipulate. We can attach to the name of each structure a string of the form <A,B>. A is the address of the first word of the structure, B of the second. A macro can do this and assign the storage.

```
            TW <ARG> =
        <   TWS1 = TWS + 1
            ARG(1) = <TW:TWS,TW:TWS1>
        TW&TWS, 0
        TW&TWS1, 0
            TWS = TWS + 2 >
```

Now, if we call TW twice after setting TWS to 1:

```
        TW    A
        TW    B
```

we will have given A the value <TW1,TW2> and B the value <TW3,TW4> and defined the four TW symbols.

We can now use A and B in the MOVE macro. In fact

```
        MOVE  A,B
```

expands to

```
        LDA   TW1
        STA   TW3
        LDA   TW2
        STA   TW4
```

With the addition of one more device we can proceed to the definition of a very grandiose macro. The directives .IF and .ELSF, used thus:

```
        .IF     E1  <...>
        .ELSF   E2  <...>
            .
            .
            .
        .ELSF   En  <...>
```

cause each E_i in turn to be evaluated until one is greater than zero. The string following this one is then scanned and the rest of the structure ignored.

```
*THIS MACRO COMPILES AN ARITHMETIC EXPRESSION CONSISTING OF SINGLE-
*LETTER VARIABLES, BINARY + AND - AND PARENTHESES.  IT CALLS THE
*MACRO ERROR IF THE EXPRESSION IS NOT WELL FORMED.

    ARITH <ARG> =
  < EXPR=<:ARG(1).>      *APPEND . TO THE EXPRESSION
    STK=<*>              *INITIALIZE THE STACK WHICH HANDLES
                         *PARENTHESES
    J=1                  *INITIALIZE THE CHARACTER POINTER
    TI=0                 *INITIALIZE THE TEMPORARY STORAGE COUNTER

*IF TEMPORARY STORAGE IS REQUIRED IT IS ASSIGNED AS TEMP1,
*TEMP2, ETC., AND TI KEEPS TRACK OF THE NEXT AVAILABLE LOCATION.

    X1                   *THIS IS THE MACRO WHICH DOES THE WORK
    .IF T .NE '.'        <ERROR> >

*CHECK THAT EXPRESSION WAS NOT TERMINATED BY A RIGHT PARENTHESIS.
*THIS MACRO COLLECTS A SUB-EXPRESSION CONSISTING OF OPERANDS
*STRUNG TOGETHER WITH + AND -.  IF THE SUBEXPRESSION IS A SINGLE
*VARIABLE, COP (CURRENT OPERAND) WILL BE THAT VARIABLE ON EXIT.
*OTHERWISE IT WILL BE EMPTY.

    X1 =
  < COP = <***>          *ENSURE THAT COP IS NOT EMPTY INITIALLY

*AN EMPTY COP MEANS THAT CODE HAS BEEN ASSEMBLED LEAVING A VALUE
*IN THE A REGISTER.  IF COP IS A LETTER, IT IS THE VARIABLE
*WHICH IS THE CURRENT OPERAND.

    OPERAND              *GET THE FIRST OPERAND
    .RPT .FOR E=1,1,0    *E IS SET TO 2 WHEN THERE ARE NO MORE + OR -
                         *SIGNS
  < T=':EXPR[J]'         *EXPECTING AN OPERATOR OR TERMINATION
    J=J+1
    .IF T .E '.' .OR T .E ')'  <E=2>

*SET E TO TERMINATE THE LOOP IN THIS CASE.

    .ELSF  T .E '+'  <COMPILE ADD,ADD>
    .ELSF  T .E '-'  <COMPILE SUB,(CNA;ADD)>

*IF A + OR - IS PRESENT, GET THE SECOND OPERAND AND COMPILE CODE.

    .ELSF 1  <ERROR>     *OTHERWISE, ERROR
    > >                  *CLOSE LOOP AND MACRO

*THIS MACRO COLLECTS THE SECOND OPERAND OF A BINARY OPERATOR AND
*CONSTRUCTS CODE TO PERFORM THE SPECIFIED OPERATION.  IT USES ITS
*FIRST ARGUMENT IF THE FIRST OPERAND IS IN THE A REGISTER, ITS
*SECOND ARGUMENT IF THE SECOND OPERAND MUST BE IN A AND THE FIRST
*TAKEN FROM MEMORY.

    COMPILE <CARG> =
  < OPERAND              *GET THE SECOND OPERAND
    .IF  .LC(COP)  .G 0

*IN THIS CASE THE SECOND OPERAND IS A SINGLE VARIABLE.

  < .IF  .LC (PREVOP)  .G 0  <LDA PREVOP>

*IF THE FIRST OPERAND IS ALSO A VARIABLE (OR A TEMP LOCATION)
*BRING IT INTO A

    CARG(1)  COP >       *AND COMPILE CODE
    .ELSF 1      <CARG(2) PREVOP>

*OTHERWISE THE SECOND OPERAND MUST BE IN A, AND THE FIRST IN MEMORY

    COP=< >      >

*SET COP TO INDICATE A VALUE IN A AND CLOSE THE MACRO.

*THIS MACRO COLLECTS AN OPERAND, WHICH MAY BE A PARENTHESIZED
*SUBEXPRESSION

    OPERAND=
  < T = ':EXPR[J]'       *GET THE NEXT CHARACTER
    J=J+1                *IT SHOULD BE A LETTER OR (
    .IF T .E '('
  < .IF .LC(COP) .E 0

*IF WE ALREADY HAVE A VALUE IN A IT MUST BE SAVED IN TEMPORARY
*STORAGE WHILE THE SUBEXPRESSION IS EVALUATED.

  < TI = TI +1:
    STA TEMP&TI          *CONSTRUCT A TEMP LOCATION TO SAVE IT IN
    COP=<TEMP:TI> >      *AND REMEMBER IT IN COP
    STK=<:COP,:STK>      *STICK COP ON THE FRONT OF STK
    X1
    .IF T .NE '('  <ERROR>
    E=1                  *RESET THE TERMINATION SWITCH FOR X1
    PREVOP=<:STK(1)>     *SET PREVOP TO THE OLD COP WHICH WAS SAVED
    STK=<:STK(2,.L(STK))> >

*REMOVE OLD COP FROM STK AND TERMINATE THIS CASE.  X1 HAS SET COP

    .ELSF  T .GE 'A' .AND T .LE 'Z'

*IF T IS A LETTER (RECALL THAT THE CHARACTER CODE IS ASCII)

  < PREVOP=<:COP>
    COP=<:EXPR[J-1]>  >
    .ELSF  1  <ERROR>  >
```

This macro, called by

```
ARITH ((A + B) — (C— D))
```

would generate

```
LDA A
ADD B
STA TEMP1
LDA C
SUB D
CNA
ADD TEMP1
```

Note that there are only three lines in the definition which actually generate code. The temporary storage location TEMP1 must be defined elsewhere.

The implementation of all this is quite straightforward. When a string is encountered, it is collected character by character, due attention being paid to colons, ampersands, brackets, and quotes, and stored away. When it is referenced, the routine which delivers characters to the assembler, which we will call CHAR, is switched from the input medium to the saved string. This process is of course recursive. When the string which is the current source of characters ends, CHAR is switched back to the string it was working on before. All the various occurrences of strings are treated perfectly uniformly, except that in the case of macro definitions the substrings of the argument string are delimited when the latter is collected to improve the efficiency. Perfectly arbitrary nesting of the various constructs is possible because of the recursiveness of the string collection and reference routines.

In the interests of efficiency the .IF directive is not handled in this way, since its subject string is scanned either once or not at all. All that is necessary is a flag which indicates whether an .ELSF directive is to be considered or ignored.

THE DEBUGGING SYSTEM

An interactive debugging system should not be designed for the occasional user. Its emphasis must be on completeness, convenience, and conciseness, not on highly mnemonic commands and self-explanatory output. The basic capabilities required are quite simple in the main, but the form is all important because each command will be given so many times.

One essential, completely symbolic input and output is half taken care of by the assembler. The other half is easier than it might seem: given a word to be printed in symbolic form, the symbol table is scanned for an exact match on the opcode bits. If no match is found, the word is printed as a number. Otherwise the opcode mnemonic is printed, indirect and index bits are checked, the proper symbols printed, and the table is scanned for the largest symbol not greater than the remainder of the word. This symbol is printed out, followed if necessary by a + and a constant.

The most fundamental commands are single characters, possibly preceded by modifiers. Thus to examine a register the user types

```
/x1-3; LDA I NUTS + 2
```

where the system's response is printed in capitals. This command may be preceded by any combination of modifiers:

- C for printout in constant form
- S for printout in symbolic form
- O for octal radix
- D for decimal radix
- R for relative (symbolic) address
- A for absolute address
- H for printout as ASCII characters
- I for printout as signed integer
- N for no printing of addresses
- L (load) for no printing of register contents

The modifiers hold until the user types a carriage return or gives another / command.

For examining a sequence of registers, the commands + and — are available. The former examines the preceding register, the latter the following register. In the absence of a carriage return the modifiers of the last examination hold. The → command examines the register addressed by the one last examined.

The contents of a register may be modified after examination simply by typing the desired new contents. Note that the assembler is always part of the command processor, and that debugging commands are differentiated by their format from words to be assembled (as noted above, an assembler line has comma or space at its first punctuation character, and all debugger lines have some other initial punctuation character). Furthermore, debugging commands may occur in macros, so that very elaborate operations can be constructed and then called on with the two or three characters of a macro name.

To increase the flexibility of debugging macros, the unary operator @ is defined. The value of @ SYM3 is the *contents* of location SYM3. With this operator, macros may be defined to type out words depending on very complicated conditions. A simple example is

```
TG<A> =
< .RPT.FOR TEMP = A(1),37777,1
  *SCAN THROUGH ALL OF STORAGE STARTING
      AT THE LOCATION GIVEN BY
  *THE FIRST ARGUMENT
< .IF @ TEMP E. (A)2
  *IF THE CURRENT LOCATION MATCHES THE
```

```
        SECOND ARGUMENT, THE SCAN IS OVER
</TEMP;                 *PRINT OUT THE
        TEMP1 = TEMP      CONTENTS
        TEMP = 37777    *SAVE THE ADDRESS
        >>>>            *AND TERMINATE
                          THE SCAN
```

Called with

TG 100,20

it will type out the first location after 100 with contents greater than 20.

Another important command causes an expression to be typed in a specified format. Thus if SYM has the value 1253 then

= sym; 1253

would be the result of giving the = command. All the modifiers are available but the normal mode of typeout is constant rather than symbolic. If no expression is given, the one most recently typed is taken. Thus, after the above command, the user might try

s= ; SYM (the system's response, the symbolic equivalent of 1253, follows the ;)

It is often necessary to search storage for occurrences of a particular word. This may be done with a macro, as indicated above, but long searches would be quite slow. A faster search can be made with

↑expression;

which causes all the locations matching the specified expression to be typed out. The match may be masked, and the bounds of the search are adjustable. This command takes all the typeout modifiers as well as

E

which searches for a specified effective address (including indexing and indirect addressing) and

X

which searches for all exceptional words (which do *not* match). For additional flexibility the user may specify a macro which will be executed each time a matching word is found.

In addition to being able to examine and modify his program, the user also needs to be able to run it. To this end he may start it at a specified location with

,G location

If he wishes to monitor its progress he may insert breakpoints at certain locations with the command

,B location

This causes execution of the program to be interrupted at the specified location. Control returns to the system, which types some useful information and awaits further commands. An alternate form of this command is

,B location,marco name

which causes the specified macro to be executed at each break, instead of returning control directly to the typewriter. Very powerful conditional tracing may be done in this way.

After a break has occurred, execution of the program may be resumed with the ,P command. The breakpoint is not affected. To prevent another break until the breakpoint has been passed n time the form

\n;

may be used. Modifiers may precede the command.

To step through the program, instruction by instruction, the command ,S may be used instead of ,P. It allows one instruction to be executed and then breaks again. $n; allows n instructions to be executed before breaking. A fully automatic trace has been deliberately omitted, but presents no difficulties in principle.

THE EDITOR

There remains one feature of great importance in the IMP system, the symbolic editor. The debugger provides facilities, which have already been described, for modifying the contents of core. These modifications, however, are not recorded in the symbolic version of the program. To permit this to be done, so that reloading will result in a correctly updated binary program, several commands are available which act both on the assembler binary and on the symbolic.

This operation is not as straightforward as it might appear, since there is no one to one correspondence between lines of symbolic and words of binary. Addresses given to the debugger of course refer to core locations, but for editing it is more convenient to address lines of symbolic. To permit proper correlation of these line references with the binary program, a copy of the symbolic file is made during loading with the address of the first and last assembled words explicitly appended to each line. Since the program is not moved around during editing, these numbers do not change except locally. When a debugging session is complete, the edited symbolic is rewritten without this information.

We illustrate this with an example. Consider the symbolic and resulting binary

```
S1  MOVE A,B   (200,201)   S1  LDA   A   200
                               STA   B   201
    ADD C      (202,202)       ADD   C   202
    STORE D,E  (203,204)       STA   D   203
                               STA   E   204
S2  BRU S1     (205,205)   S2  BRU   S1  205
```

and the editing command

```
,I   S2-1        insert before line S2-1
     SUB F
```

which gives rise to the following:

```
S1  MOVE A,B   (200,201)   S1      LDA   A     200
                                   STA   B     201
    ADD C      (202,1512)          BRU   .END  202
```

```
    SUB F       (1513,1513)          BRU  .END 1  203
    STORE D,E   (1514,204)           STA  E       204
S2  BRU S1      (205,205)       S2   BRU  S1      205
                                     ...
                                .END ADD  C       1512
                                     SUB  F       1513
                                     STA  D       1514
                                     BRU  S1 4    1515
                                     BRU  S1 5    1516
```

All the BRU (branch unconditional) instructions are inserted to guarantee that the right thing happens if any of the instructions causes a skip. The alternative to this rather simple-minded scheme appears to be complete reassembly, which has been rejected as too slow. The arrangement outlined will deal correctly with patches made over other patches; although the binary may come to look rather peculiar, the symbolic will always be readable.

To give the user access to the readable symbolic the command

,L symbolic line address[,symbolic line address];

(where the contents of the brackets is optionally included) causes the specified block of lines to be printed. Two other edit commands are available:

,D symbolic line address[,symbolic line address];

which deletes the specified block of lines, and

,C same arguments;

which deletes and then inserts the text which follows. Deleting S1 1 from the original program would result in binary as follows

```
S1    LDA   A
      BRU   .END
      BRU   .END 1
      STA   D
      STA   E
S2    BRU   S1
      ...
.END  STA   B
      BRU   S1 3
```

The implementation of these commands is quite straightforward. One entire edit command is collected and the new text, if any, is assembled. Then the changed core addresses are computed and the appropriate record of the symbolic file rewritten.

The scheme has two drawbacks: It does not work properly for skips of more than one instruction or for subroutine calls which pick up arguments from following locations, and it leaves core in a rather confusing state, especially after several patches have been made at the same location. The first difficulty can be avoided by changing large enough segments of the symbolic. The second can be alleviated by reassembly whenever things get too unreadable.

The only other published approach to the problem of patching binary programs automatically is that of Evans,[6] who keeps relocation information and relocates the entire program after each change. This procedure is not very fast, and in any event is not practical for a system with no relocation.

EFFICIENCY

The IMP system depends for its viability on fast assembly. The implementation techniques discussed in this paper have permitted the first version of the assembler to attain the unremarkable but satisfactory speed of 200 lines per second. Simple character handling hardware would probably double assembly speed on simple assemblies and produce even greater improvement on programs with many macros and repeats.

Using the latter figures, we deduce that a program of 10,000 instructions, a large one by most standards, will load in 25 seconds. This number indicates that the cost of the IMP approach is not at all unreasonable—far more computer time, including overhead, is likely to be spent in the debugging operations which follow this load. When only minor changes are made it is, of course, possible to save the binary core image and thus avoid reloading.

In spite of the speed of the assembler, it is possible that a relocatable loader might be a desirable adjunct to the system. There are no basic reasons why it should not be included.

As to the size of the system, the assembler is about 2500 instructions, the debugger and editor about 2000.

ACKNOWLEDGMENTS

The ideas in this paper owe a great deal to many stimulating conversations between the author and Peter Deutsch. I am especially indebted to him for the idea that all strings in the input can be handled uniformly with string brackets. A system very similar to this one has been implemented by him for the CDC 3100.

REFERENCES

1. M. Halpern, "XPOP—A Metalanguage without Metaphysics," *AFIPS Conf. Proc.*, vol. 25, 1964.
2. G. Mealy, "Anatomy of an Assembly System," RAND Corporation (Dec. 1962).
3. S. Boilen et al, "A Time-Sharing Debugging System for a Small Computer," *AFIPS Conf. Proc.*, vol. 23, Spartan Books, Washington D.C., 1963, pp. 51-58.
4. L. P. Deutsch and B. W. Lampson, "DDT—Time-Sharing Debugging System Reference Manual," Project GENIE Doc. 30.40.10 (May 1965).
5. "The MIDAS Assembly Program," internal memorandum, M.I.T.
6. Thomas G. Evans and D. L. Darby, "DEBUG—

An Extension to Current Online Debugging Techniques," *Comm. ACM,* vol. 8, no. 5, pp. 321-25 (May 1965).

7. C. N. Mooers, "TRAC, A Procedure-Describing Language for the Reactive Typewriter," *Comm. ACM* vol. 9, no. 3, pp. 215-219 (Mar. 1966).

An integrated computer system for engineering problem solving

DANIEL ROOS

Department of Civil Engineering,
Massachusetts Institute of Technology
Cambridge, Massachusetts

SYSTEM OBJECTIVES

Computers should provide the mechanism that enables engineers to do better engineering. By permitting faster, more accurate and complete problem analysis to be performed, computers assist the engineer in his computational and decision making roles. The engineer today is faced with problems of increasing magnitude and complexity where the effects and interrelationships of all relevant information must be considered. The computer can provide the coordinating and integrating mechanism for this problem information.

This role of the computer in engineering is only achieved when the computer is adequately integrated in the problem-solving environment. The computer must be properly integrated with both the programmer developing it and the engineer using it.

How is this integration of engineer and computer accomplished? The engineer must do more than use the computer. He must actively participate in the computer solution. To do this he needs a language to communicate with the computer, physical accessibility to the computer, and the ability to obtain engineering oriented results from the computer. The communication language must be oriented to the problem rather than the machine and must allow the engineer to easily specify his problem solving requirements. Accessibility, provided through some type of remote console, permits the engineer to interact with the computer during the problem solution. Meaningful engineering-oriented results are obtained using scopes, plotters and other output devices.

Integration of the programmer and the computer is provided through a powerful programming language and the necessary system programs to support the language. The programming language should be dynamic with respect to both problem solution and computer memory requirements. It should allow complete problem solutions where the type and the amount of data can vary. To satisfy these requirements, the system should include dynamic memory allocation, alternate forms of data structure, and a data management and transfer mechanism so that the same data can be used in all aspects of the problem solution.

The Civil Engineering Systems Laboratory at M.I.T. is currently developing a computer system, ICES (**I**ntegrated **C**ivil **E**ngineering **S**ystem) based on the above requirements. It includes a computer programming language, ICETRAN (**ICES FORTRAN**), oriented toward engineering problem solving, and an engineering-oriented operating monitor system. These programming aids enable programmers to easily create and modify ICES, which can then be used by the engineer in the solution of engineering problems.

ENGINEERING PROBLEM CHARACTERISTICS

What are the characteristics of engineering problems and how do they affect the design of ICES? The solution of civil engineering problems generally involves the consideration of many disciplines. For example, even in a relatively small problem such as the design of a highway interchange, engineers must consider the highway location and design (highway engineering), settlement, stability, and foundation conditions (soil engineering), highway bridges (structural engineering), drainage (hydraulic engineering), and traffic flow (transportation engineering). Engineers recognize each of these

separate disciplines and the necessary interactions that must exist for effective problem solving. In the past, the complexity of engineering problems and the large amount of data often forced an engineer to unnaturally decompose a problem into noninteractive tasks. Many of the feedback aspects of the problem had to be overlooked.

A computer system such as ICES offers the mechanism to permit complete problem solutions where all factors and interactions are included. *In ICES each of the civil engineering disciplines is included as one or more subsystems which may interact with one another.* An engineer proceeds in his problem solution by using the necessary subsystems at the proper time. At any point in his problem solution he can leave one subsystem, enter another to perform calculations and then reenter the original subsystem using the results just obtained.

Two engineers given the same problem will quite often arrive at correct yet totally different solutions. Engineering is a creative process, so it is natural to expect a problem solution to reflect an engineer's own experience. The ICES subsystems allow a civil engineer to solve the many different problems that arise, while not constraining his role as a decision maker.

The subsystem design reflects the way an engineer does engineering. He performs a series of fundamental engineering and mathematical operations to obtain a problem solution. These fundamental operations can be considered basic computational building blocks. *Each subsystem of ICES contains a set of building block subroutines.* These subroutines, developed by ICES subsystem programmers, are written in ICETRAN, a compiler language somewhat similar to FORTRAN.

The set of basic building blocks alone, however, is not sufficient. The engineer must decide what operations (building blocks) are to be performed (used), the sequence of these operations and the data associated with each operation. It is impractical and undesirable to try to include all these decision making functions in the subsystem computer programs. Instead, they must be specified by the engineer at execution time.

PROBLEM-ORIENTED LANGUAGES

A language is therefore needed to allow the engineer to easily communicate this information to the computer. This language must not be oriented to the programmer as is ICETRAN but instead oriented toward the engineer and the problem he is solving. An engineer should be able to communicate with the computer in much the same way he communicates with another engineer. He must instruct the machine rather than follow instructions that have been imposed by machine.

The engineering communication language should be command structured where each command represents an operation or group of operations the computer is to perform. The commands are technical terms which have meaning to the engineer. Each command may also contain data relevant to the requested operations. An engineer must choose which commands to use, the order of their use, and the associated data. Two engineers might solve the same problem using totally different commands. With a command structured language, the quality and efficiency of the computer solution is dependent on the engineering ability of the user.

A command structured language enables the engineer to program a computer without being concerned with programming details. Although the commands result in a program to the computer in that the engineer is "instructing" the computer, they appear to the engineer as a logical problem statement in engineering terminology. Such languages may therefore be referred to as *problem-oriented languages.*

Two different types of programming languages, each used by a different type of person, are therefore involved in ICES. The ICETRAN language is used by programmers to write ICES subsystem subroutines. Problem-oriented languages are used by engineers at execution time to specify the characteristics of the particular problem being solved.

FORTRAN and other similar languages (ICETRAN) have sometimes been referred to as problem-oriented languages. It appears more realistic to refer to these languages as procedure-oriented, and reserve the term problem oriented for a language that satisfies the following requirements:

1. It is oriented to the user rather than the programmer. No computer programming knowledge is required to effectively use the language.
2. The language is command structured where the commands are composed of technical terms.

To avoid further confusion in this paper, it is necessary for the reader to distinguish between a problem-oriented language used by an engineer and a procedure-oriented language used by a programmer.

The benefits of the problem-oriented language approach have already been demonstrated. COGO (**CO**ordinate **GeO**metry) is a problem oriented language used for the solution of geometric problems developed by Professor C. L. Miller of the Department of Civil Engineering at M.I.T. It is interesting to note the impact of COGO on the civil engineering profession since its introduction in 1960. It is used by many state highway

departments and private consulting firms. One state highway department uses COGO for over 50% of its computer runs. COGO supersedes several hundred special-purpose geometric programs. The use of COGO has also resulted in increased engineerng productivity and improved engineering design. It has introduced many somewhat reluctant engineers to the computer, and it has produced considerable self-satisfaction among the engineers who have used it. Many engineers have become very frustrated as a result of computer use where they were relegated to filling out cumbersome input forms for use with library programs. COGO allows these engineers to perform engineering while using the computer rather than simply filling out forms. Similar favorable reactions from the profession have also been observed with regard to another problem oriented language STRESS (**STR**uctural **E**ngineering **S**ystem **S**olver), also developed by the Department of Civil Engineering at M.I.T. Past COGO and STRESS work serves as the basis for two subsystems of ICES.

In ICES the engineer formulates his problem solution using the appropriate subsystem commands. The engineer's commands are processed by the *ICES executive program* which performs the following operations:

Read and encode command.
Analyze, convert, and store data.
Perform consistency checks.
Transfer control to subroutines for command execution.

The computer solution consists of two phases: the analysis of the command by the ICES executive and the execution of the command using the appropriate computational subroutines. A command is completely processed before the next command is read. The system operates somewhat as an interpreter, where programs that have previously been written in a procedure-oriented language and translated to machine language instructions are CALLed.

COMMAND STRUCTURE

With ICES the structure of problem-oriented languages has been generalized so that the same ICES executive program can be used by all ICES subsystems. The command vocubulary of each subsystem differs, but the command structure is identical. A command consists of a command name and data relating to the requested operation. The following features are included in the command structure:

1. The data are free format, whereby the engineer need not be concerned with cumbersome card column restrictions.

2. The data associated with a command may be identified by labels and specified in any order. If the engineer prefers, he may omit the labels and enter the data in a standard order. An example of a command with labeled data items to store the X and Y coordinates of point 10 is

```
STORE  POINT 10  X 500  Y 750.63
```

Without labels, the command would be

```
STORE  10  500  750.63
```

3. The engineer may omit a command data value and a standard value will automatically be inserted by the executive. This value may be either:

(a) permanently preset by the programmer when the command is initially set up;
(b) temporarily preset by the engineer at the beginning of the problem;
(c) the current value of the data item.

The programmer who initially sets up the command indicates which of these options will be followed. If a standard value is not associated with the data item, the executive will indicate an error condition whenever the data value are omitted by the engineer. The use of standard values reduces the amount of unnecessary data an engineer must include with a command, since input entries are made only when a nonstandard value is encountered.

4. Data values may be alphanumeric as well as numeric. For example, in a geometric problem the quadrant of an angle may be identified as NE, SE, SW, or NW. The executive will automatically set a switch based on the alphanumeric value given by the user. This switch can then be interrogated by the subsystem command processing routines.

5. Incremental as well as complete processing of the input command is permitted, i.e., if so instructed, the executive will process part of the data, transfer control to execute that portion of the process, and then return to repeat the cycle, continuing until the data are exhausted. Incremental processing minimizes storage requirements when a command contains a variable number of data items.

6. The executive can initialize variables and increment counters that are specified by the programmer at command definition time. These variables and counters can then be used by the command processing routines.

7. The command data field can contain command name modifiers which permit complicated tree structured commands to be set up by the programmer.

To illustrate command name modifiers, consider the three versions of the INTERSECTION command shown below.

```
INTERSECTION  POINT 5  LINE 10
  LINE 20
INTERSECTION  POINT 5  LINE 10
  ARC 20  NEAR 3
INTERSECTION  POINT 5  ARC 10
  ARC 20  NEAR 3
```

The command data consist of the POINT number assigned to the intersection point and the two geometric objects (ARC or LINE) being intersected. In the last two commands, a known point NEAR the desired intersection point must be specified, since a line and an arc can intersect at 2 different points.

The engineer thinks of these as the same command (INTERSECTION); but the programmer must think in terms of three different commands, since each has a different type and amount of data associated with it, and each requires different subroutines for execution. The programmer views the command as the tree structure shown in Fig. 1, where the labels ARC and LINE serve as command name modifiers. These modifiers

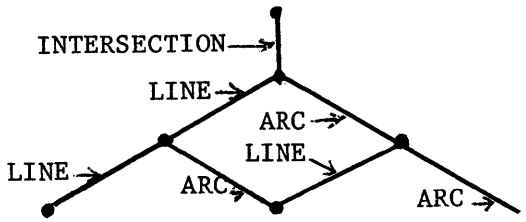


Figure 1. The use of command name modifiers.

determine the appropriate branch of the tree and the data and subroutine requirements associated with that branch.

Command name modifiers minimize the number of necessary commands and increase the capabilities of the commands.

COMMAND DEFINITION LANGUAGE

Before an engineer can use a subsystem and its related commands, the subsystem must be added to the ICES system. A set of utility programs are available to perform these modifications. The modifications are performed as normal jobs under ICES, making the system self-modifying. The utility programs include a command definition language which enables a programmer to specify the characteristics of the subsystem commands. This command definition language is a problem-oriented language designed for the programmer which automatically generates the command tables used by the ICES executive.

ICES therefore includes a problem-oriented language to generate problem-oriented languages. In addition to generating new problem-oriented languages, the command definition language can be used to easily add, modify, or delete commands from a subsystem that already exists. This ability to easily modify a subsystem is a necessary requirement in engineering organizations where both the problems and the organization itself can change.

The simple example below illustrates some of the features of the ICES executive and the command definition language. The STORE command used to define the X and Y coordinates of a known point will be added to the COGO subsystem of ICES. The STORE command as entered by the engineer could appear as:

```
STORE POINT 10  X 1000.53  Y 960
```

The command definition program adds the new command to the COGO vocabulary. The underlined words are the vocabulary of the command definition language and the information in quotes is the input data the programmer supplies.

```
SYSTEM 'COGO'
ADD 'STORE'
ID 'P' INTEGER 'NPOINT' REQUIRED
ID 'X' REAL 'XCOORD' STANDARD 0
ID 'Y' REAL 'YCOORD' STANDARD 0
EXECUTE 'STORE'
FILE
```

SYSTEM specifies the ICES subsystem (COGO) being modified, and ADD specifies the type of modification (addition of a command) and the name of the command (STORE). ID gives the characteristics of the data items associated with the command. One ID entry is required for each of the three data items of the STORE command. The information included as part of the ID entry includes:

1. *The identifier for the data item.* The identifier defines permissible labels that the engineer can use to label the data items. A permissible label is one that begins with the specified identifier. For example, the identifier of P permits the engineer to use P, PT, POINT, etc., as labels for the point number. The identifiers for the X and Y coordinates given in the second and third ID commands are X and Y.
2. *The internal machine representation of the data item* (REAL or INTEGER).
3. *The symbolic computer location where the data*

item will be stored (NPOINT, XCOORD, YCOORD).

4. *The action to be followed if the data item is omitted by the engineer.* A REQUIRED data item must be entered by the engineer or an error will be indicated by the executive. The STANDARD entry is used to specify a preset data value that will be used by the executive if the data item is omitted by the engineer. Since the coordinates of a point are quite often (O,O) they have been preset to these values in the above example. If the action field of the ID command is left blank (this does not occur in the above example), then the data item will retain its current value if omitted from the command by the engineer.

EXECUTE specifies the subroutine to be entered (STORE) to execute the command, and FILE concludes the command definition. If no errors have been detected, the necessary command tables for the newly defined command will be generated and the command will be added to the COGO subsystem of ICES. The command definition language enables a programmer to easily incorporate problem-oriented language command input in ICES. These commands can then be used by engineers in the solution of their problems.

ON-LINE OPERATIONS

In certain cases the engineer can use a problem-oriented language to write all the commands for a problem solution. It is not always possible or desirable, however, for the engineer to function in this manner. Quite often he will formulate part of the solution and then base the remaining portion on the results of the first part. This mode of operation is incompatible with typical batch processing operations. Instead, the engineer must be able to communicate with the computer in an *interactive* environment where he can:

Request an operation.
Examine his results.
Determine the next operation to be performed based on the previous results.

This suggests that a problem oriented language functions best in an on-line environment. In this environment the engineer can try many alternate designs and compress into one session the same work that would require many individual sessions in a batch processing mode.

Engineering is typically performed under severe time restraints. The turn-around time problem inherent in batch processing can totally negate the otherwise beneficial results of computers. The engineer needs immediate access to the computer. He can obtain this accessibility through a remote console. The combination of accessibility of a computer and the communication capability of problem oriented languages provides the engineer with the necessary tools for effectively using the computer in engineering problem solving.

ICES DATA STRUCTURE

An engineer solves a problem by performing operations on data. He collects, reduces, analyzes and evaluates data to arrive at meaningful decisions. The type, amount, and reliability of these data vary considerably from problem to problem. A flexible and efficient computer system such as ICES, therefore, requires a flexible and efficient internal data structure.

It is difficult to classify engineering data since they are of many types and varieties. In some cases only the numeric value of data is important, where, in other cases, hierarchical and structural relationships between the data are also important. Data with different external characteristics require different internal computer representation. Some engineering data are best stored as arrays, others as lists, and others as combinations of arrays and lists. In the past, computer systems have generally been limited to one predominant form of storage (either list or array). *ICES has alternate forms of data structure, namely, arrays, lists and array-lists.* The programmer chooses the appropriate structure for his data based on considerations such as space requirements, access time, information content, manipulation ability, and system overhead. The alternate forms of data structure allow new representations of engineering problems on a computer.

DYNAMIC MEMORY ALLOCATION

Most engineering data are best represented in a computer in array form. To achieve maximum capability and remove the restrictions presently associated with normal FORTRAN DIMENSIONed array storage, arrays can be dynamically allocated. Dynamic allocation of data achieves the following:

1. Arrays are allocated space at *execution* time rather than at *compilation* time. They are only allocated the amount of space needed for the problem being solved. The size of the array (i.e., the amount of space used) may be changed at any time during program execution. If an array is not used during the execution of a particular problem, then no space will be allocated.
2. Arrays are shifted between primary and secondary storage to attempt to optimize the use of primary memory. The programmer need not be concerned

with this shifting which is performed by ICES system programs.

Dynamic memory allocation is a necessary requirement for an engineering computer system capable of solving different problems with different data size requirements. The result of dynamic memory allocation is that the size of a problem that can be solved is virtually unlimited since secondary storage becomes a logical extension of primary storage. An explanation of the ICES dynamic memory allocation scheme and an example of its use appear later in this paper.

With ICES, dynamic memory allocation is extended to programs as well as data. Programs are brought into primary memory only when they are needed. The allocation of programs and data is properly balanced so that the use of primary memory is optimized.

The memory allocation problem has been considerably simplified as a result of newly announced third generation computer hardware, in particular, new random access storage devices. The new machines are faster, more powerful, larger and cheaper than their second generation counterparts. The large primary and secondary storage available at reduced prices should increase the size of problems that can be solved on the computer and decrease the cost of the problem solution.

DATA MANAGEMENT

Secondary storage can contain one or more data files associated with the complete problem solution. ICES data management system programs control the organization of these data files. They allow the same data file to be used by all subsystems of ICES, and each subsystem to use different names to refer to the same data file. Variable and array names in one subsystem are mapped into different variable and array names in the other subsystems. Whenever an engineer switches from one subsystem to a different subsystem, data transfer programs use the mapping function to automatically rearrange the data for the new subsystem.

Data management also allows an engineer to easily operate on data files. He can print, modify, or delete any of his files. The data files can serve as permanent documents of completed problems. Space considerations will of course influence how much information an engineer should keep on secondary storage and for how long this information will be retained.

The data management facility also permits public files to be created so that several engineers can have access to the data to work on the same problem. Suitable protection features are provided to ensure that unauthorized users may not examine another person's files.

ICETRAN PROGRAMMING LANGUAGE

If system programming capabilities such as data management, data transfer, dynamic memory allocation, etc., are to be effectively used by programmers, they must be implemented in a suitable programming system. In ICES, these system programming capabilities are included in the ICES operating system and the ICES programming language. The FORTRAN programming language has been extended into a language called ICETRAN (**ICES-FORTRAN**) which will be used to program the ICES subsystems. ICETRAN contains all of the FORTRAN statements plus additional capabilities to facilitate the problem solving capabilities of ICES. These additional capabilities are imbedded in the normal FORTRAN structure. No new programming restrictions are imposed on the programmer.

To illustrate ICETRAN, an example will be presented showing how dynamic memory allocation is incorporated in the language. To understand the example, however, it is first necessary to briefly summarize the ICES dynamic memory allocation scheme.

Associated with each dynamic array is one COMMON location known as a codeword-pointer. This location contains the following information about the dynamic array:

1. The size of the array.
2. The residence of the array (primary storage, secondary storage, or array not yet defined).
3. A pointer to the first location of the array in the data pool. All dynamic arrays are allocated space in a data pool, which consists of the unused primary memory space during program execution time.
4. The status and priority of the array (used during memory reorganization). Dynamic memory allocation implies that the array space requirements are constantly changing. If the data pool becomes full and more space is needed, then a memory reorganization must be performed. This reorganization is based on the status (active, released, or destroyed) and the assigned priority (high or low) of the array. A sufficient number of arrays are transferred to secondary storage to make room for new active arrays. If an array on secondary storage is later referenced, it will be brought back into primary memory by ICES system programs.

An ICETRAN subroutine illustrating the use of dynamic memory allocation appears below. This sub-

routine adds two one-dimensional dynamic arrays (A and B) to form a new dynamic array (C).

```
      SUBROUTINE ADD (N)
      COMMON A, B, C, . . . . .
      DYNAMIC ARRAYS A, B, C
      DEFINE C, N, HIGH
      DO 1 L = 1, N
1     C(L) = A (L) + B (L)
      DESTROY A
      RELEASE B
      RETURN
      END
```

The DYNAMIC ARRAYS statement is used to specify all dynamic arrays. The statement does not cause any instructions to be generated by the compiler. The DEFINE statement is used by a programmer to specify information about a dynamic array that will be stored in the codeword-pointer. The DEFINE statement in the above example causes the size (N) and priority (HIGH) of dynamic array C to be inserted in the codeword-pointer of array C. Dynamic arrays A and B have already been DEFINEd in the subprogram that called subroutine ADD.

The DEFINE statement does not cause allocation of space for the array. The allocation of space is delayed until the first reference to an array element is made (statement 1). The first time statement 1 is referenced "N+1" contiguous unused locations in the data pool will be obtained. If they are unavailable, a memory reorganization will occur. The pointer of the codeword will then be set to point to the first of the N+1 locations. The first N locations are used to store the array, and the N+1 location contains a back-pointer to the codeword. If the array is shifted in the data pool during memory reorganization, the pointer of the codeword is appropriately adjusted.

After the DO loop is completed, array A is destroyed (DESTROY A) and array B is temporarily released (RELEASE B). An array should be destroyed if it is no longer needed and released if it is not presently needed. Intelligent use by the programmer of Destroy and Release statements decreases the likelihood of memory reorganization and increases the efficiency of one when it does occur.

The dynamic memory allocation scheme can handle arrays of any number of dimensions. *An* n *dimensional array is treated internally as a partitioned set of sub-arrays*. This offers extreme flexibility since:

1. Only the subarray an engineer is using need be in core.
2. The *size* of each subarray can differ. Figure 2 shows a two dimensional array where the size of each subarray (column) varies (2, 4 and 3).
3. The *structure* of the subarray can vary so that tree structures can be represented using array notation. A tree structure with the associated subscript notation is shown in Fig. 3.

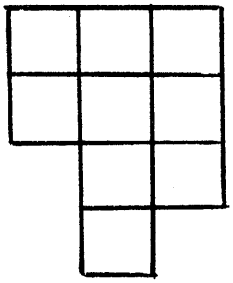


Figure 2. Array size variation.

The programmer can easily set up a desired structure using the DEFINE command and then operate on the structure using normal FORTRAN array notation.

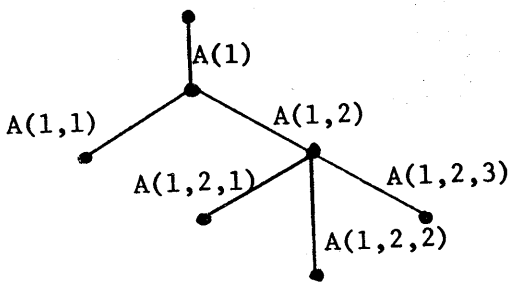


Figure 3. Array structure variations.

Dynamic memory allocation capability is merely one of many features available with ICETRAN. Others include list processing, data management, subsystem data transfer, and matrix manipulation.

A completed ICETRAN program must first be processed by the *ICES precompiler,* which will translate the ICETRAN statements into legitimate FORTRAN statements. Although the generated FORTRAN statements might appear somewhat confusing to a FORTRAN programmer, they will accomplish the requested operations. The resulting FORTRAN program is then processed by the conventional FORTRAN compiler. Use of the precompiler eliminates the necessity of modifying the FORTRAN compiler or developing a totally new programming language.

ICES SYSTEM PROGRAMS

ICETRAN is being developed by people who are experienced in both the information sciences and civil

engineering. In the past, a schism has existed between computer language developers and application users. Language developers have not completely appreciated the needs of users, hence their languages were not well suited for the intended users. ICES, however, is developed for civil engineers by civil engineers.

A major difficulty associated with the COGO and STRESS development was the amount of manpower, time, and money that had to be invested to produce the necessary system programs. For example, each language had its own executive routine. There was concern that as more problem-oriented languages developed, the system programming effort would become immense.

One of the goals of ICES was to create the necessary system programs so that programmer-engineers not experienced in system programming could create powerful subsystems. The one set of ICES systems programs are applicable to all the ICES subsystems.

Some people will no doubt express concern about the overhead time of the system programs included in ICES. Quite often, however, an increase in running time is completely justified by the benefits obtained from the operations performed. For example, dynamic memory allocation does involve additional running time, but it also eliminates many bookkeeping requirements and the necessity of packing information. It also increases the problem-solving capabilities.

With ICES, no system programs are forced upon a programmer. He is, for example, given the choice between normal dimensioned arrays and dynamic arrays, and between the problem-oriented language input capability and normal FORTRAN READ statements. The programmer must consider the problem being solved and the tradeoffs associated with each of the available alternatives before making his choice.

SUMMARY

Briefly summarizing, ICES contains the following capabilities:

1. Flexible and powerful problem-oriented languages for the engineer to communicate with the computer.
2. On-line capabilities to make the computer accessible to the engineer.
3. An orientation toward complete problem solving, not just computation.
4. An interaction between discipline areas—or subsystems—for solving a problem which encompasses more than one engineering discipline.
5. A data management scheme to organize the data and a data transfer mechanism to pass the data between subsystems.
6. A modular internal building block structure.
7. An efficient internal data structure where alternate forms of data can be represented.
8. Dynamic allocation of data and programs based on the problem being solved.
9. The ICETRAN programming language to be used by programmers to develop ICES subsystems.

The first operational version of ICES is currently being implemented on the model 40 IBM System/360 computer in the Civil Engineering Systems Laboratory at M.I.T. The system programs have been written and several subsystems are now being developed. ICES subsystem work should not, however, be restricted to M.I.T. Instead, ICES should be considered as a framework for computer work in civil engineering, where all members of the profession make contributions. It is thus the integrating mechanism for the programs that have been developed in the past and the work that will be performed in the future.

ACKNOWLEDGMENTS

A project such as ICES is the result of careful planning and represents a natural evolution of previous work. For the past 10 years, Professor C. L. Miller, as director of the Civil Engineering Systems Laboratory at M.I.T., has demonstrated how computers can effectively be used in engineering practice and education. This past work serves as the foundation for ICES, which was conceived by Professor Miller and is now being developed by individuals who were trained and inspired by him. These individuals include Robert Logcher, Jay Walton, Alan Hershdorfer, Joe Sussman, Alden Foster, Ron Walter, and Richard Goodman. The work reported in this paper was formulated by these people. The author is also indebted to Miss Betsy Schumacker of IBM and Joel Winett of M.I.T. Lincoln Laboratory for their advice and contributions.

BIBLIOGRAPHY

CHAMPY, J. A., "Man Machine Computer Systems in a Public Works Agency: A Management View," *Industrial Management Review,* Spring 1965.

FENVES, S. J., et al, *STRESS: A User's Manual,* M.I.T. Press, 1964.

FENVES, S. J., R. D. LOGCHER, and S. P. MAUCH, *STRESS: A Reference Manual,* M.I.T. Press, 1965.

MILLER, C. L., "Man-Machine Communications in Civil Engineering," M.I.T. Department of Civil Engineering, T63-3 (June 1963).

MILLER, C. L., and R. WALTER, "Communicating with Computers in Civil Engineering Design," M.I.T. Department of Civil Engineering, T65-4. (Mar. 1965).

ROOS, D., and B. SCHUMACKER, "ICES: Integrated Civil Engineering System," M.I.T. Department of Civil Engineering, P65-8 (May 1965).

ROOS, D., and C. L. MILLER, "COGO-90: Engineering User's Manual," M.I.T. Department of Civil Engineering, R64-12 (Apr. 1964).

ROOS, D., and C. L. MILLER, "The Internal Structure of COGO-90," M.I.T. Department of Civil Engineering, LR64-5 (Feb. 1964).

ROOS, D., and C. L. MILLER, "COGO-90 Time-Sharing Version, M.I.T. Department of Civil Engineering, LR 64-18 (May 1964).

ERRATA

to

Volume 27, Part I

AFIPS
Conference Proceedings*
1965 Fall Joint Computer Conference

*Published by Spartan Division of Books, Inc., 1250 Connecticut Ave., Washington, D.C.

ERRATA to

AFIPS Conference Proceedings

1965 Fall Joint Computer Conference—Part I

(*Abbreviations*: *p.* = *page; L* = *left column;* R = *right column*)

W. H. BURKHARDT
Universal Programming Languages . . .

p. 2, line 26, L: *Change to read* in the SEAC machine[1]) this idea . . .
last line, R: *Change to read* when relating a machine with 0.5 — microsecond cy —

p. 3, line 1, L: *Change to read* cle time in 1965 to one with 25 microsecond cycle

p. 4, line 5, R: *Change* Lanig *to* Laning
line 15, L: *Change to read* translation of previously . . .
line 22ff. R: *Change to read*
3. Time savings in debugging and correcting the programs,
4. Easier modification possibilities for slightly different problems,
5. Higher . . .

p. 6, line 22, L: *Change to read* areas (not vertical combinations or notations[23]).
line 33, L: *Change to read* machines by combination . . .
section on "Mathematical Definition and Development": *Replace all subscript numerals 1 by lower case letter "el."*
last line of this section: by an example in reference 28).
second last line, R: *Delete* 30.

p. 7, line 9, R: *Change to read* IBSYS[31]). The . . .
line 11, L: *Change to read* As mentioned in the section on Possible Solutions, there
in section on "Transformation of Programs by Processors": *Replace all subscript numerals 1 by lower case letter "el."*

p. 8, line 5 fr. below, L: *Change to read* Language A into target

p. 9, figures: *Delete all underlining*

p. 10, figures: *Delete all underlining; change first arrow to a simple one in each figure*
in first figure: *Change* L *to* L_3

p. 11, figures: *Delete all underlining; change first arrow to a simple one in each figure*
line 14, R: *Change* (case 1) to (case 31)
line 15, L: *Change to read* —spectively, for the . . .
line 15, R: *Change* (case 2) to (case 32)
line 19, R: *Change* (case 2) to (case 32)

p. 12, Table 1: *Move the words* Processor in M *from* TGS row, col. 4, *down to* Meta A row, col. 4.

p. 13, figures: *Change first and third arrow in figure to a simple arrow.*
figure on left: *Move to end of paragraph*

p. 14, second row of figures, last figure: *Change* M_1 *to* N_1

p. 15, figures: *Delete all overlines*
last figure: *Add the angles in the brace*

p. 16, first figure: *Change* $|N_2|$ *to* (N_2)
last figure: *Change* P(L) *to* p(L)

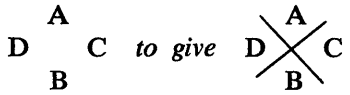
p. 17, first figure, R: *Delete all vertical bars*
second figure, R: *Change all horizontal arrows to double arrows*
last figure, R: *Exchange this figure for the first figure on page* 18

p. 18, first figure: *Exchange this figure with the last figure on p.* 17, *R*
second figure: *Insert a simple arrow between* M_2 *and the big* X
last figure: *Insert a simple arrow between* E *and the big* X

p. 19, line 39ff., L: *Change to read* the plan is not

possible and to state the conditions as to which way it would then be possible. But new ideas for solutions and new insights into the problems . . .

line 27ff: *Insert a big* X *between the letters*:



TOBEY, BOBROW, and ZILLES
Automatic Simplification in FORMAC

p. 38, footnote, R: *Change* called *to* call
p. 39, line 31, L: *Change* (1 + Z**N *to* (1 + Z)**N
p. 40, line 38, L: *Change to read* 30 years
line 30, R: *Invert the factor* $\frac{D}{C}$ *to read* $\frac{C}{D}$
line 31, R: *Insert* = *after* ϕ
p. 42, line 22, R: *Change* A = 0 *to* A $\neq$ 0
line 31, R: *Second* n *belongs above* Σ
line 33, R: *Both* j = 1 *belong under* Σ
line 34, R: j $\neq$ k *belongs under second* j = 1
line 36, R: B_j *belongs close to first* Π
p. 43, line 43, L: *Change to read* unexpanded
line 32, R: *Change* script "e1" *to* lower-case italic "el"
p. 44, line 8, L: *Place arrow after* A_j
line 8, R: *Change* optional *to* optimal
line 36, R: *Change to read* transformation
line 40, R: *Insert parenthesis between* * *and* C
p. 45, lines 20 and 21, R: *Should begin with the symbol* →
p. 46, last line, L: *Change first* of *to* or
line 14, R: *Change* of *to* or
p. 47, line 7, L: *Change* K *to lower-case* k
line 10, L: *Insert* ↑ *at end of line*
line 24, L: *Change* ↕ *to* ↑
line 5, R: *Insert a dot over the* * *and delete the dot over the* 5
line 6, R: *Insert a dot over the* * *and delete the dot over the* 5
line 11, R: *Insert dots over the* *BC
line 42, R: *Change* ↕ *to* ↑
p. 48, line 6, L: *Change both* ↕ *to* ↓; P *should point to* +
line 7, L: *Change* ↕ *to* ↑
line 13, L: *Change both* ↕ *to* ↑
line 18, L: *Change both* ↕ *to* ↓
line 19, L: *Change* ↕ *to* ↑
line 37, L: *Change* ↕ *to* ↓
line 38, L: *Insert a dot over the second* * *and the* W; *change* ↕ *to* ↑
p. 48, line 38, L: *Delete the dot over the* 2
line 12, R: *Change* ↕ *to* ↑
p. 50, line 12, L: *Change to read* identical
line 14, L: *Insert right parenthesis after* numeric
line 15, L: *Change to read* * A B C K_2] where K_1 and K_2 are numbers;
line 17, L: *Change* SI X *to read* SINX
line 23, L: *Change both* ↕ *to* ↑
line 24, L: *Change both* ↕ *to* ↑
line 27, L: *Change* ↕ *to* ↑
line 4, R: *Delete left parenthesis before* COS(A)
line 45, R: *Change* ↕ *to* ↑
p. 51, line 25, L: *Change* ↕ *to* ↑
line 31, L: *Change* ↕ *to* ↑
line 22, R: *Insert* ↑ *after* K_1
line 23, R: *Change* ↕ *to* ↑
line 36, R: *Change* UTSIM *to* AUTSIM
p. 52, line 22, R: *Change first* and *to* or
line 44, R: *Change* is *to* in

B. J. KARAFIN
New Block Diagram Compiler . . .

pp. 60-61: *Correct entries in* Table 1 *as follows*:

Block Type	Function	Output
ACC	Accumulation	$y_k = x_k + y_{k-1}$ (p_1-2)
FLT	Transversal filter	$y_k = \sum_{1=0} p_{p_1-1}\, x_{k-1p_2}$
AMP	Amplifier	$y = p_1 x$
QNT	Quantizer	p_1 in the number of representative levels. The representative levels are $p_2, p_4, \ldots, p_{2p1}$. The decision levels are $p_3, p_5, \ldots, p_{2p1-1}$.
LQT	Linear quantizer	$y = \left(\frac{x}{p_1}\right)$ rounded up. p_1.
SQT	Square root	$y = \sqrt{x}$
FOP	Floating point to integer conversion	

S. P. FRANKEL and J. HERNANDEZ
Precession Patterns in Delay Line Memories

p. 99, line 21, L: *Change* 10 *to* 10^6

D. N. SENZIG and R. V. SMITH
Computer Organization for Array Processing

p. 117, title, remove asterisk and footnote at bottom left

p. 118, line 45, L: *Change to read* $\underline{w}$, $\underline{u}$, $\underline{s}$
line 50, L: *Change* X^i *to* $\underline{X}^i$
line 51, L: *Change* Z^i *to* $\underline{Z}^i$
line 1, R: *Change* X^i *to* $\underline{X}^i$
line 2, R: *Change to read* $\underline{X}$. $\underline{X}^i_j$ will represent the jth bit . . .
line 3, R: *Change* X^i *to* $\underline{X}^i$
line 3, R: *Change* X^i *to* $\underline{X}^i$
line 5, R: *Change* Z^i *to* $\underline{Z}^i$
line 7, R: *Change to read* $\underline{X}^i$. The $\underline{Z}^i$ register . . .
line 8, R: *Change to read* the $\underline{u}$, $\underline{s}$, and $\underline{w}$ . . .
line 9, R: *Change* X *to* $\underline{X}$
line 10, R: *Change* Z *to* $\underline{Z}$
line 11, R: *Change* X *to* $\underline{X}$
line 14, R: *Change* Z *to* $\underline{Z}$
line 15, R: *Change to read* in $\underline{Z}$ and $\underline{X}$
line 16, R: *Change* X *to* $\underline{X}$
line 17, R: *Change* Z *to* $\underline{Z}$
line 19, R: *Change* X *to* $\underline{X}$
line 20, R: *Change* X *to* $\underline{X}$
line 30, R: *Change to read* $\underline{s}$ (screen) and $\underline{u}$
line 32, R: *Change* X^i *to* $\underline{X}^i$; Z^i *to* $\underline{Z}^i$; *and* s *to* $\underline{s}$
line 33, R: *Change* u *to* $\underline{u}$; s_i *to* $\underline{s}_i$ *and* u_i *to* $\underline{u}_i$
line 34, R: *Change* X^i *to* $\underline{X}^i$; *and* Z^i *to* $\underline{Z}^i$
line 35, R: *Change* s_i *to* $\underline{s}_i$
line 44, R: *Change* u *to* $\underline{u}$
line 46, R: *Change* X *to* $\underline{X}$
line 48, R: *Change* X *to* $\underline{X}$; *and* u *to* $\underline{u}$
line 50, R: *Change* s *to* $\underline{s}$, *Change* u *to* $\underline{u}$

p. 119, line 7, L: *Change* u *to* $\underline{u}$
line 9, L: *Change* Iversons's *to* Iverson's
line 11, L: *Change* compress *to compress*
line 12, L: *Change to* vector $\underline{a}$. . .
line 13, L: *Change* u *to* $\underline{u}$, *Change* c←u/a *to* $\underline{c}\leftarrow\underline{u}/\underline{a}$
line 14, L: *Change* c *to* $\underline{c}$
line 15, L: *Change* a *to* $\underline{a}$; a_i *to* $\underline{a}_i$; *and* u_i *to* $\underline{u}_i$
line 16, L: *Change* c *to* $\underline{c}$
line 17, L: *Change* u *to* $\underline{u}$; *and* u *to* $\underline{u}$
line 18, L: *Change* a *to* $\underline{a}$; *and* u/a *to* $\underline{u}/\underline{a}$
line 19, L: *Change* a *to* $\underline{a}$
line 20, L: *Change* X *to* $\underline{X}$
line 22, L: *Change* c *to* $\underline{c}$; *and* X *to* $\underline{X}$
line 23, L: *Change* u *to* $\underline{u}$
line 24, L: *Change* X *to* $\underline{X}$; *and* c *to* $\underline{c}$
line 26, L: *Change* expand *to expand; and* c←u/a *to* $\underline{c}\leftarrow\underline{u}\backslash\underline{a}$
line 27, L: Change c to $\underline{c}$, *and* u to $\underline{u}$
line 29, Change c_i to $\underline{c}_i$; and a_i to $\underline{a}_i$
line 30, L: Change u_i to $\underline{u}_i$
line 31, L: *Change to read* $\underline{u} = (1,0,0,0,1,1)$; *change* X to $\underline{X}$
line 32, L: *Change* X *to* $\underline{X}$
line 34, L: *Change* u *to* $\underline{u}$; *and* X to $\underline{X}$
line 51, L: *Change* w *to* $\underline{w}$
line 1, R: *Change* X *to* $\underline{X}$; *and* s *to* $\underline{s}$
line 2, R: *Change* w *to* $\underline{w}$
line 3, R: *Change to read* $(\underline{w}\leftarrow\underline{w} + \sum_i \underline{X}^i$; $\underline{w}\leftarrow\underline{w} \times \prod_i \underline{X}^i$
line 5, R: *Change* w *to* $\underline{w}$; *and* X *to* $\underline{X}$
line 6, R: *Change* w *to* $\underline{w}$
line 11, R: *Change to read* for each AU.
line 12, R: *Delete* (Fig. 3.
line 51, R: *Change to read*: *vector direct* mode,

p. 120, line 17, L: *Change to read*: vector indirect
line 19, L: *Change* Z *to* $\underline{Z}$

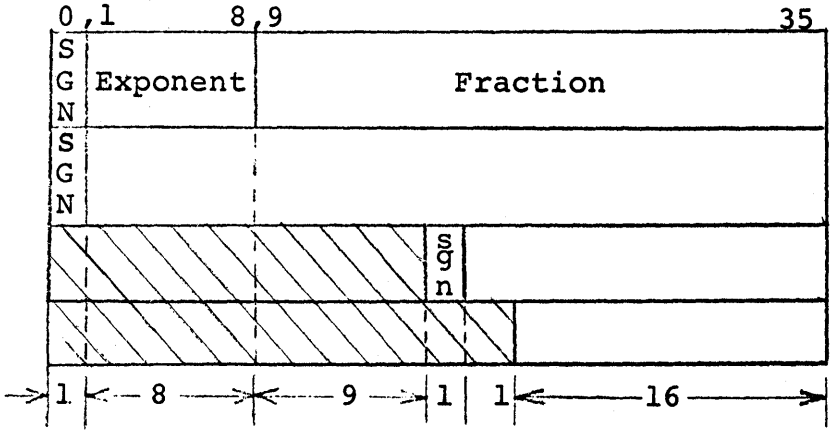
p. 124, line 33, L: *Change* poit *to* point
left side, below line 40, and entire right column should appear as figure shown on page 166.

p. 125, line 8, L: *Change to* (and, or, equivalence)
line 24, L: *Change to* (the right-most
line 30, L: *Change to* computed from I1
line 31, L: *Change to* Bit 13 is
line 42, L: *Change to* the *address vector,* $\underline{a}$, is
line 5, R: *Change* X *to* $\underline{X}$
line 22, R: *Change* stapping *to* stepping
line 14, R: *Change* X *to* $\underline{X}$

p. 126, line 23, L: *Change* w, s, *to* $\underline{w}$, $\underline{s}$,
line 24, L: *Change* u, X and Z arrays *to* $\underline{u}$, $\underline{X}$ and $\underline{Z}$ arrays
line 27, L: *Change* Z, A is *to* $\underline{Z}$, $\underline{A}$, is

p. 127, line 47, L: *Change* X *to* $\underline{X}$; *and* s *to* $\underline{s}$
line 2, R: *Change* X *to* $\underline{X}$
line 4, R: *Change* s *to* $\underline{s}$
line 5, R: *Change* X *to* $\underline{X}$
line 6, R: *Change* X *to* $\underline{X}$
line 8, R: *Change* X *to* $\underline{X}$
line 10, R: *Change* X *to* $\underline{X}$
line 12, R: *Change* X *to* $\underline{X}$
line 14, R: *Change* s *to* $\underline{s}$
line 19, R: *Change* X *to* $\underline{X}$; *and* s *to* $\underline{s}$
line 30, R: *Change* X *to* $\underline{X}$; *and* s *to* $\underline{s}$
line 31, R: *Change* X *to* $\underline{X}$
line 35, R: *Change to* BSS A Block Started by Symbol

p. 128, line 5 of Fig. 6: *Change to* SCREEN TO 1
line 6 of Fig. 6: *Change to* A2(I + 1),..., A2(I + 15)
line 7 of Fig. 6: *Change to* BINARY PT. AT RIGHT
line 8 of Fig. 6: *Change to* EXPONENT



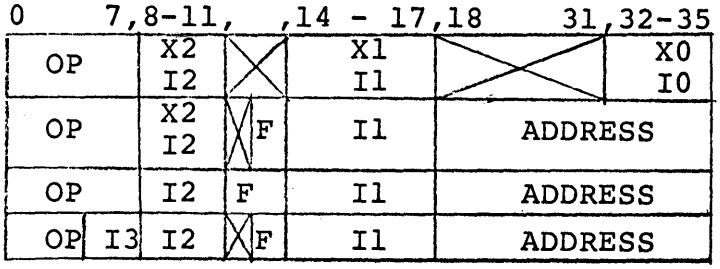
Floating Point Word

Double Length Index

Single Length Index/Address

Logical Acc./Screen

Instruction Formats:



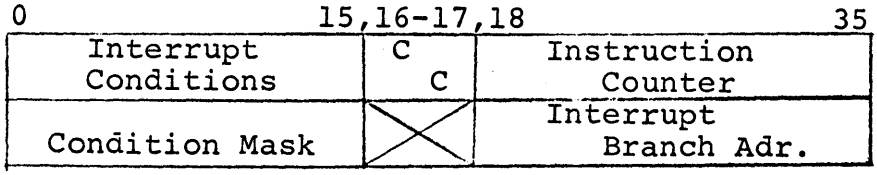
Scalar Register-Register

Scalar Register-Memory

Vector

Transfer

Control Words:



Program Status Word

Program Status Word Mask

(CC - condition code)

line 10 of Fig. 6: *Change to* B(1) to ELEMENTS
line 11 of Fig. 6: *Change to* B(A2(I)), B(A2(I + 1)),
line 12 of Fig. 6: *Change to* A2 + 1,2,,1 ADD A2(I + 1),A2(I + 2),..., A2(I + 16)
line 13 of Fig. 6: *Change to* A2—I,2,,1 ADD A2(I — 1),A2(I),...,A2(I + 14)
line 15 of Fig. 6: *Change* to A2(I + 1),..., A2(I + 15)

F. J. CORBATO and V. A. VYSSOTSKY
Introduction and Overview of Multics System

p. 186, line 6, R: *Change* Refs. 23 and 24 *to* Refs. 42 and 24
p. 190, line 45, L: *Change* more compilations. *to* more compilation time.
p. 196: *Insert at bottom,* Ref. 42 *as*: A. L. Samuel, "Time-Sharing on a Computer," *New Scientist* 26, 445 (May 27, 1965) 583-87.

E. A. BOWLES
The Role of the Computer . . .

p. 270, line 51, L, last word: *Change* oppositions *to* opposition
line 10, R: *Change* kowing *to* knowing
line 15, R: *Change* Blender *to* Blendor
line 20, R: *Delete* Old
p. 272, lines 38 and 39, L: *Reverse their order*
line 6, R: *Insert comma after* tables
p. 273, line 18, R: *Change* separate *to* different
p. 276, line 3, R: *Change* Pres *to* Press
line 11, R: Italicize and capitalize Ibid

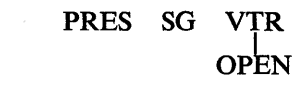
D. C. CLARKE and R. E. WALL
An Economical Program . . .

p. 313, footnote to Table 1: *Should be associated with* Table 4, p. 314.

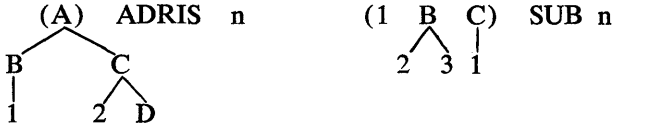
ZWICKY et al
MITRE Syntactic Analysis Procedure

p. 320, lines 43-47, R and p. 321, lines 1-4, L: Correct format to read as follows:
Phrase Structure Component:
75 rules
approximately 275 subrules
Transformational Component:
13 initial singularies
26 embeddings and related singularies, including 9 embeddings
15 final singularies
54 rules

p. 321, line 34, L: *Change placement of* OPEN to read



p. 322, line 17-19, L: Add lines to diagrams to read



p. 322, line 26, R: *Correct* "Checking by Synthesis" *as secondary heading (as in line* 18*), italized, flush left, and with following text entered with paragraph indentation.*
p. 323, line after line 12, R: *Insert* "OTHER APPROACHES TO THE ANALYSIS PROBLEM" as major heading (as in line 26, L above—"Areas for Further Investigation") above "Analysis by Synthesis"
p. 325, lines 12-29, L: *Correct format to read:*
(Forward) Grammar
Phrase Structure Component:
61 rules
105 subrules
Transformational Component:
11 initial singularies
6 embeddings and related singularies, including 2 embeddings
3 final singularies
20 rules
Surface Grammar 32 rules
306 subrules
Reversal Rules 6 final singularies
15 embeddings and related singularies
11 initial singularies
32 rules

R. C. MINNICK
Cobweb Cellular Arrays

p. 328, line 8, L: *Change* index *to* Index
p. 329, caption for Fig. 2: *Change* one *to* One
p. 329, caption for Fig. 3: *Should read* Cutpoint Realization for a Three-Bit Parallel Adder
p. 330, caption for Fig. 4: *Should read* Cutpoint Reali-

zation for a Five-Bit Shift Register
caption for Fig. 5: *Should read* Cutpoint Realization for Three Functions of Three Variables
p. 331, caption for Fig. 6: *Should read* Structure of the Cobweb Array
p. 332, caption for Fig. 7: *Should read* Diode-Transistor Realization of Cobweb Cell
line 17, L: *Poorly printed word is* cutpoint
p. 333, Eq. (3): *Replace* θ *with* $\oplus$
line 1, R: *Replace* θ *with* $\oplus$
p. 334, caption for Fig. 8: *Should read* Shannon and Reed Decompositions Using Cobweb Arrays
p. 335, caption for Fig. 9: *Should read* Cobweb Realization for a Three-bit Parallel Adder
caption for Fig. 10: *Should read* Cobweb Realization for a Five-Bit Shift Register
p. 336, line 21, R: *Change* focal *to* local
caption for Fig. 11: *Should read* Cobweb Realization for Three Functions of Three Variables
caption for Fig. 12: *Should read* Cobweb Supercells
p. 337, caption for Fig. 13: *Should read* Exhaustive Listing of the Cobweb Array Fault-Avoidance Algorithm
p. 338, caption for Fig. 14: *Should Read* Block Diagram for a Twelve-Bit Serial Multiplier
p. 339, caption for Fig. 15: *Should read* Realization of the Multiplier in Terms of Five Cutpoint Arrays
p. 340, caption for Fig. 16: *Should read* Realization of the Multiplier in Terms of One Cobweb Array

R. H. CANADAY
Two-Dimensional Iterative Logic

p. 343, line 23, R: *Change to* $\bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C}$
p. 344, line 17, R: *Change to* $f^d(x_1,...,x_n) = \bar{f}(\bar{x}_1,...,\bar{x}_n)$
p. 345, line 17, L: *Change* Xf_o^d *to* Yf_o^d
p. 346, line 25, R: *Change* C *to* $\bar{C}$
line 26, R: *Change* C *to* $\bar{C}$ *and* $\bar{U}$ *to* U
line 27, R: *Change* $\bar{U}$ *to* U
line 33, R: *Change to* $f^{sd} = B\bar{C}D\bar{U} + A\bar{B}C\bar{D}\bar{U} + \bar{A}BC\bar{D} + \bar{A}\bar{B}C\bar{D} + AB\bar{C}$
line 34, R: *Change to* $+ \bar{A}\bar{B}\bar{C}U + \bar{A}BCU + A\bar{B}CDU + B\bar{D}U + \bar{C}\bar{D}U$
p. 349, line 11, L: *Change* {B,B,U,U} *to* $\{B,\bar{B},U,\bar{U}\}$
line 31, L: *Change* {C,C,O,1} *to* $\{C,\bar{C},O,1\}$
line 7, R: *Change* (n-1 *to* (n-1)
line 22, R: *Change to* $g_1 = \bar{B}\bar{C}_{g100} + \bar{B}C_{g101} + B\bar{C}_{g110} + BC_{g111}$
Array 5, R: *Change* element 1,6 *from* B *to* $\bar{C}$
p. 350, Array 7, R: *Change* element 2,6 *from* g_{1101} *to* g_{1100}
line 15, R: *Change* A + B *to* $A \oplus B$
line 16, R: *Change* + C +D *to* $\oplus C \oplus D$
p. 351, line 4, L: *Change to* $\{x_1,\bar{x}_1x_2,\bar{x}_2,...,\bar{x}(n-1),U,\bar{U}\}$ (U,$\bar{U}$) being
Array left boundary, L: *Change all* $\bar{V}$ *to* $\bar{U}$
p. 352, line 17, L: *Change* UU *to* $U\bar{U}$
line 18, L: *Change* U_{g0}^d *to* $\bar{U}_{g0}^d$
line 19, L: *Change* UU *to* $U\bar{U}$
line 20, L: *Change* U_{g1}^d *to* $\bar{U}_{g1}^d$
line 5, R: *Change* U_{g0}^d *to* $\bar{U}_{g0}^d$
line 6, R: *Change* U_{g1}^d *to* $\bar{U}_{g1}^d$
line 10, R: *Change* UU *to* $\bar{U}U$
line 22, R: *Change to* $g_0^{sd} = UR_0g_{00} + US_0g_{01} + \bar{U}R_0g^d_{01} + \bar{U}S_0g^d_{00} + UR_0S_0$
line 24, R: *Change to* $g_1^{sd} = UR_1g_{10} + US_1g_{11} + \bar{U}R_1g_{11}^d + \bar{U}S_1g_{10}^d + UR_1S_1$
line 29, R: *Add the term* VXR_0S_0
line 30, R: *Change to* $+ VYS_1g_{11} + VYR_1S_1 + WXR_1g^d_{11} + WXS_1G^d_{10}$
line 31, R: *Change* g_{01} to g^d_{01} and g_{00} to g^d_{00}
line 33, R: *Change* UU *to* $\bar{U}U$
line 35, R: *Change* UU *to* $\bar{U}U$; *following that line should be the heading* SUMMARY
line 41, R: *Change* AB+AC+BC *to* $\bar{A}\bar{B}+\bar{A}\bar{C}+\bar{B}\bar{C}$
p. 353, line 1: *Eliminate* SUMMARY

R. A. SHORT
Two-Rail Cellular Cascades

p. 360, line 12, R: *Change* f *to* $\bar{f}$
line 15, R: *Change* f *to* $\bar{f}$
p. 361, lines 4, 5, 6 L, eq. for f $(x_1,..., x_n)$: *Change all* + *to* $\oplus$
line 10, L: *Change* f_2: $(y_2, y_2 + xy_1)$ *to* f_2: $(y_2, y_2 \oplus xy_1)$
line 20, L, eq. for N: *Change* $\binom{n}{i}$ *to* $\binom{n}{i}$
line 7 from bottom, R: *Change* + *to* $\oplus$
p. 362, line 3, L: *Change* + *to* $\oplus$
line 5, 6, 7 from bottom, L: *Change all* + *to* $\oplus$
line 2, R: *Change* $\binom{n-3}{i}$ *to* $\binom{n-3}{i}$
p. 363, line 2, 3 from bottom, L: *Change all* + *to* $\oplus$
p. 365, line 5, 6 from bottom, R: *Change all* + *to* $\oplus$
p. 367, Fig. 11 (f): *Change* g_2 *to* y_2

G. E. ROUDABUSH et al
Left Hand of Scholarship . . .

p. 399, line 2, title: *Change* MEDIA *to* MEDIUM
p. 401, line 39, L: *Omit* the
line 33, R: *Space between* we *and* received
p. 402, line 2, L: *Change* PC *to* PC6
p. 403, line 6, L: *Change* printd *to* printed
p. 404, line 16, R: *Change* gold *to* bold
p. 405, line 2, L: *Change* slit *to* split
p. 406, line 20, L: *Change* in our office *to* done
line 11, R: *Omit* that
line 12, R: *Change* dattum *to* datum
p. 408, line 13, L: *Change* see *to* set

D. ROOS
An Integrated Computer System . . .

p. 424, Fig. 1, R: *Distance of* 500.26 *left out in figure*
p. 426, line 50, L: *Change* sybsystems *to* subsystems
p. 427, line 38, R: *Change* DO I L = 1, I *to* DO 1 L = 1, I
p. 428, line 6, R: *Change* DOP *to* DO
p. 430, line 38, R: *Change* 'N POINT' *to* 'NPOINT'
p. 432, line 8, L: *Change* System 360 *to* System/360
line 11, L: *Change* System 360 *to* System/360
p. 433, line 5, R: *Replace line in* 4 *with name* R. D. Logcher

C. W. ADAMS
Responsive Time-Shared Computing . . .

p. 486, line 6, R: *Should follow* line 8

L. W. COTTEN
Circuit Implementation . . .

p. 489, line 19, L: *Change* 23 *to* 20
p. 491, line 32, L:

$$p_{12} = \sum_{i=1}^{n-k} A_i B_{i+k}$$

p. 492, line 36, L: *Change* NOR-OA *to* NOR-OR
line 27, R: *Change* $f = AB$ *to* $f = A\overline{B}$
line 28, R: *Change* $+ AB$ *to* $+ \bar{A}B$
p. 494, line 2, L: *Change* (X) *to* $(\overline{X})$
line 36, L: *Change* $(X_1 -)$ *to* $(\overline{X}_1 -)$
line 34, R: *Change* G *to* $\underline{G}$
line 35, R: *Change to* $\underline{G} = \overline{C} - \underline{C} + \overline{D} + \bar{A}$
line 41, R: *Change to* $\underline{G} = \overline{H} - \underline{H} + \overline{I} + \overline{E}$
p. 495, line 5, L: *Change to* $\underline{G} = \overline{K} - \underline{K} + \overline{M} + \overline{J}$
line 18, L: *Change* T *to* $\overline{T}$
line 19, L: *Change* T *to* $\underline{T}$
line 23, L: *Change to* $\underline{G} = \overline{3T} - \underline{T}$
line 24, L: *Should read* As $\overline{T}$ is made to approach $\underline{T}$, $\overline{G}$ will approach $\underline{G}$,
line 25, L: *Change* G *to* $\underline{G}$
line 26, L: *Change to* $\underline{G} = 2\overline{T}$
line 42, L: *Change* 4T *to* $4\overline{T}$
line 7, R: *Change* 4T *to* $4\overline{T}$
line 25, R: *Change* our *to* out
line 27, R: *Change* (X_1) *to* $(\overline{X}_1)$
p. 497, line 6, L: *Change* 4T *to* $4\underline{T}$; *and* where T *to* where $\underline{T}$
line 12, L: *Should read* alter $\overline{T}$ and $\underline{T}$
line 10, R: *In two places change* 4T *to* $\underline{4T}$
line 12, R: *Should read* $3\overline{T} - \underline{T}$
line 18, R: *In both places change to* $3\overline{T} - \underline{T}$
line 22, R: *Should be* $3\overline{T} - \underline{T}$
Change 4T *to* $4\underline{T}$
line 23, R: *Change* T/T *to* $\overline{T}/\underline{T}$
line 24, R: *Change* T *to* $\overline{T}$
line 27, R: *Change* 4T *to* $4\overline{T}$
line 28, R: *Change* T *to* $\overline{T}$
line 30, R: *Change* T *to* $\underline{T}$; *and* T/T *to* $\overline{T}/\underline{T}$
line 32, R: *Change* $T =$ *to* $\underline{T} =$; *and* 3T *to* $3\overline{T}$
line 33, R: *Change* T/T *to* $\overline{T}/\underline{T}$
line 39, R: *Change* T *to* $\underline{T}$
line 43, R: *Change* T *to* $\underline{T}$
line 45, R: *Change* T *to* $\underline{T}$
p. 498, line 42, L: *Should read* clock system of $\underline{W} =$
line 4, R: *Change to* $(\underline{T_1 + T_2 + T_3 + T_4})$
line 7, R: *Change* (4T) *to* $(4\underline{T})$
line 11, R: *Should read* $\underline{T} = 2.0$
line 17, R: *Change to* $(\overline{T_1 + T_2 + T_3}) - \underline{T}$
line 23, R: *Should read* has negligible $\overline{T} - \underline{T}$
line 26, R: *Should be* $(\overline{T_2 + T_3})$
line 27, R: *Should read* with negligible $\overline{T} - \underline{T}$
p. 499, line 4, L: *Should read* $W_2 = 4\underline{T} - W_1 = 4\underline{T} - W_c - S_c$
line 9, L: *Should read of* $4\underline{T} + Z$; *and* delay is $S_c + 4\overline{T} + Z$
line 10, L: *Should be* $W_3 = (S_c + 4\overline{T} + Z) - (4\underline{T} + Z)$
line 11, L: *Should be* $= 4(\overline{T} - \underline{T}) + S_c$
line 13, L: *Change* W_4 *to* W_1
line 15, L: *Should be*

$$= \frac{1}{(W_c + 2S_c) + (4\underline{T} - W_c - S_c) + 4(\overline{T} - \underline{T})}$$

line 18, L: *Should read* When $4\underline{T}$
line 22, L: *Change to* $f = 1/(S_c + 4\overline{T})$
line 23, L; *Should read* When $4\underline{T}$
line 25, L: *Change term* $(T - T)$ *to* $(\overline{T} - \underline{T})$
line 27, L: *Change* If 4T *to* If $4\underline{T}$

Change then $T =$ *to* then $\underline{T} =$
line 28, L: *Should read* If this $\underline{T}$ is
line 30, L: *Change both* 4T terms *to* $4\overline{T}$
line 31, L: *Should be* It is seen that $\underline{T}$
line 33, L: *Should be* When $4\underline{T}$
line 36, L: *Change* 4T *to* $4\underline{T}$
line 2, R: *Change both* 4T terms *to* $4\underline{T}$
line 4, R: *Change* 4T term *to* $4\overline{T}$; *Should read* speed $\overline{T}$ might
line 6, R: *Should read* If $\overline{T} = 4.5$, $\underline{T} = 2.5$,
line 10, R: *Change term* 4T *to* $4\overline{T}$
line 11, R: *Change term in parens to* $(\overline{T} - \underline{T})$
line 15, R: *Should read* such that the $\underline{T}$

p. 503, line 26, R: *Change* Relay *to* Delay
line 29, R: *Change* T *to* $\overline{T}$
line 31, R: *Change* T *to* $\underline{T}$
line 35, R: *Should read* $\overline{X}$: Logic sense
line 38, R: *Should read* $\underline{X}$: Minimum value

G. H. BALL
Data Analysis in the Social Sciences . . .

p. 533: *Delete entire left column, replacing the two existing paragraphs with the three following paragraphs*:

The advent of the relatively inexpensive digital computer makes iterative cluster-seeking methods of analyzing complex multivariate data practical when they could not be seriously considered before. These "cluster-seeking" techniques provide a way of viewing multivariate data that differ from factor analysis and discriminant analysis.

Cluster-seeking techniques are best suited to examining problems where the data are multi-model. They provide a way of detecting isolated data points that are not "close" (relative to the data set) to any other points. These techniques can be used to show the relationship of a single data point to the entire set of data—thus allowing an examination of the details in the data. Large numbers of data points can be structured and related to each other in the original high-dimensional space.

We feel that the techniques described below provide a useful adjunct to other methods of analyzing multivariate data. We compare and describe below, a number of these methods reported in the literature.

p. 534, L: *A portion of* Fig. 1 *has been omitted. See the accompanying figures for complete* Fig. 1.
R: *Delete last paragraph and replace with the following text*:

A useful description of all the data can be plotted using the axes found by factor analysis or principal components analysis. Even this may not be as meaningful as desired as can be seen by examining Figs. 2a and 2b where we see that two quite different two-dimensional probability distributions both give rise to marginal distributions that are uniform along vertical and horizontal axes. We can see that this inability to distinguish between these different distributions can be resolved by using more axes to describe the data (as shown by the marginal distributions along the diagonal axes). Relating events on the different axes, particularly when the data are high-dimensional is difficult, however. The major problem, then, is that two sets of data that are quite dissimilar can appear to be quite similar when viewed by data analysis techniques implicitly oriented toward Gaussian distributions.

p. 535, L: *Delete lines* 1, 2, and 3.
R: *A portion of* Fig. 2b *is missing in print. See the accompanying figures for complete* Fig. 2.

p. 538, L: *Delete the second paragraph and replace it with the following*:

The sample size requirement relates to any quantities that must be estimated (in a statistical sense) from the set of sample patterns. Allais[3] showed that this requirement cannot be taken lightly. He shows that given N samples the estimation of a covariance matrix of dimension greater than N/10 usually increases the probability of error for predictions based on that covariance matrix as compared with predictions using a covariance matrix of fewer dimensions. Small sample sizes seem to require that simpler quantities be estimated—such as the means of clusters, or only the largest eigenvector of the covariance matrix.

p. 539, Table 2 (section Weighted Boolena "and": *Add a dot to show multiplication of terms.*
$l, X_{kj} \bullet X_{rj} = O$
(Section "Normalized correlation"): *Equation should read* $l = 1$

p. 540, Table 3 (section "Entropy"): *Change equation* $i = 1$ *to* $j = 1$
(Section "Square (used for deviation from single signal)"): *Equation on last line should read* $j = 1$
(Section "Value for a cluster"): *Equation starting with* I_{xi} *change* to I_{xy}
(Section "Coefficient of belongingness"): *Term shown on last line should be* i,j∉Ck

p. 541, table (section "Total entropy"): *term* $-\frac{1}{2}\sum_{ij}$. *should be* $-\frac{1}{2}\sum'_{ij}$.
In the other two instances where the symbol Σ *is used,* prime symbol *should be at right, thus*: Σ'

p. 543, line 20, L: *Change* Mattson (1965) *to* Mattson & Dammann (1965)

p. 546, line 8, L: *Add* stored in memory *so that the line will read as follows*: fraction of the waveform stored in memory as it exists at the time the

p. 547, lines 44 and 45, L: *Change* troubles *to* troubled

p. 548, L: *Change the following lines to read*:
line 20: cluster is broken into two clusters by creating two

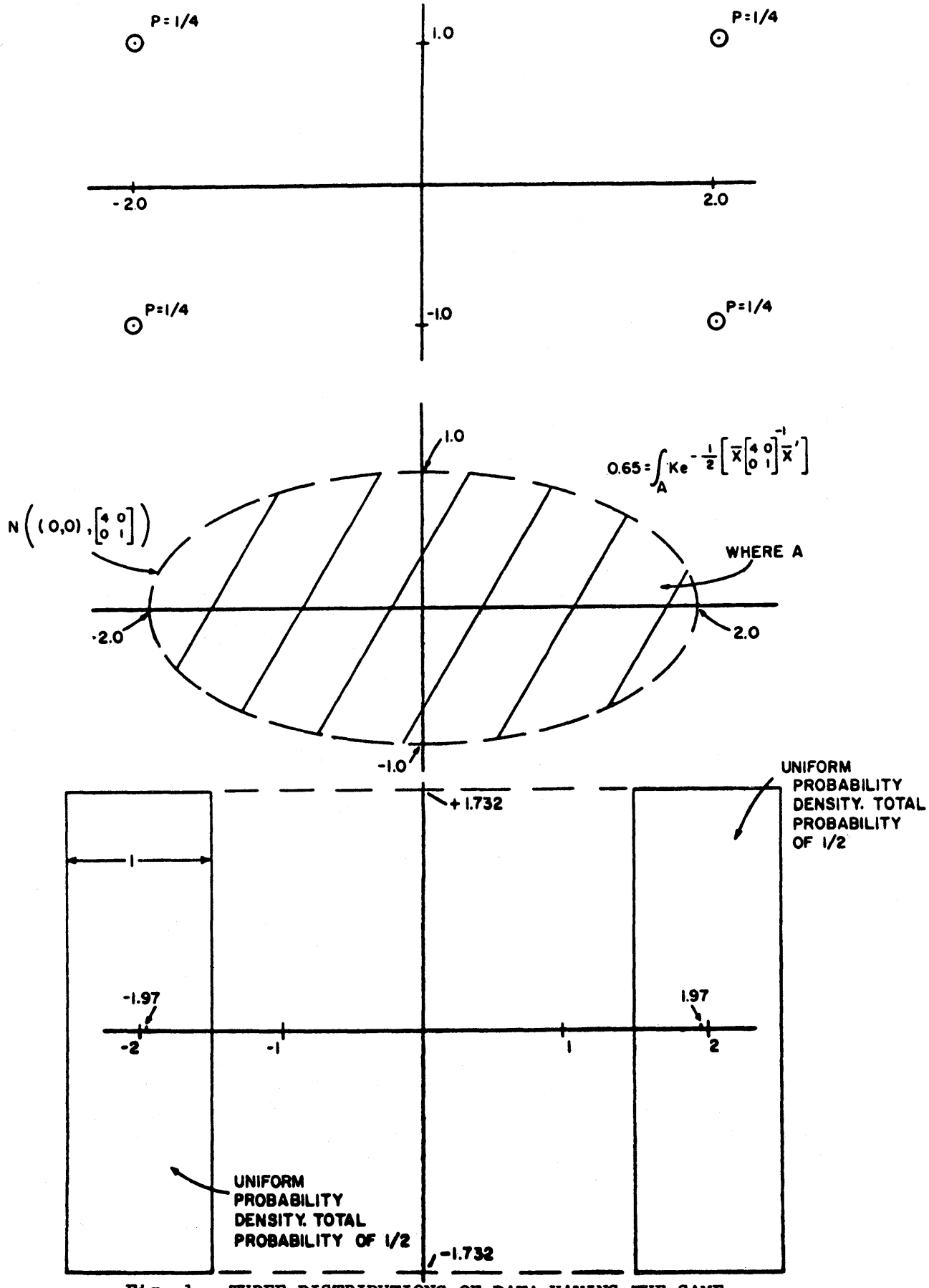


Fig. 1 THREE DISTRIBUTIONS OF DATA HAVING THE SAME COVARIANCE MATRIX

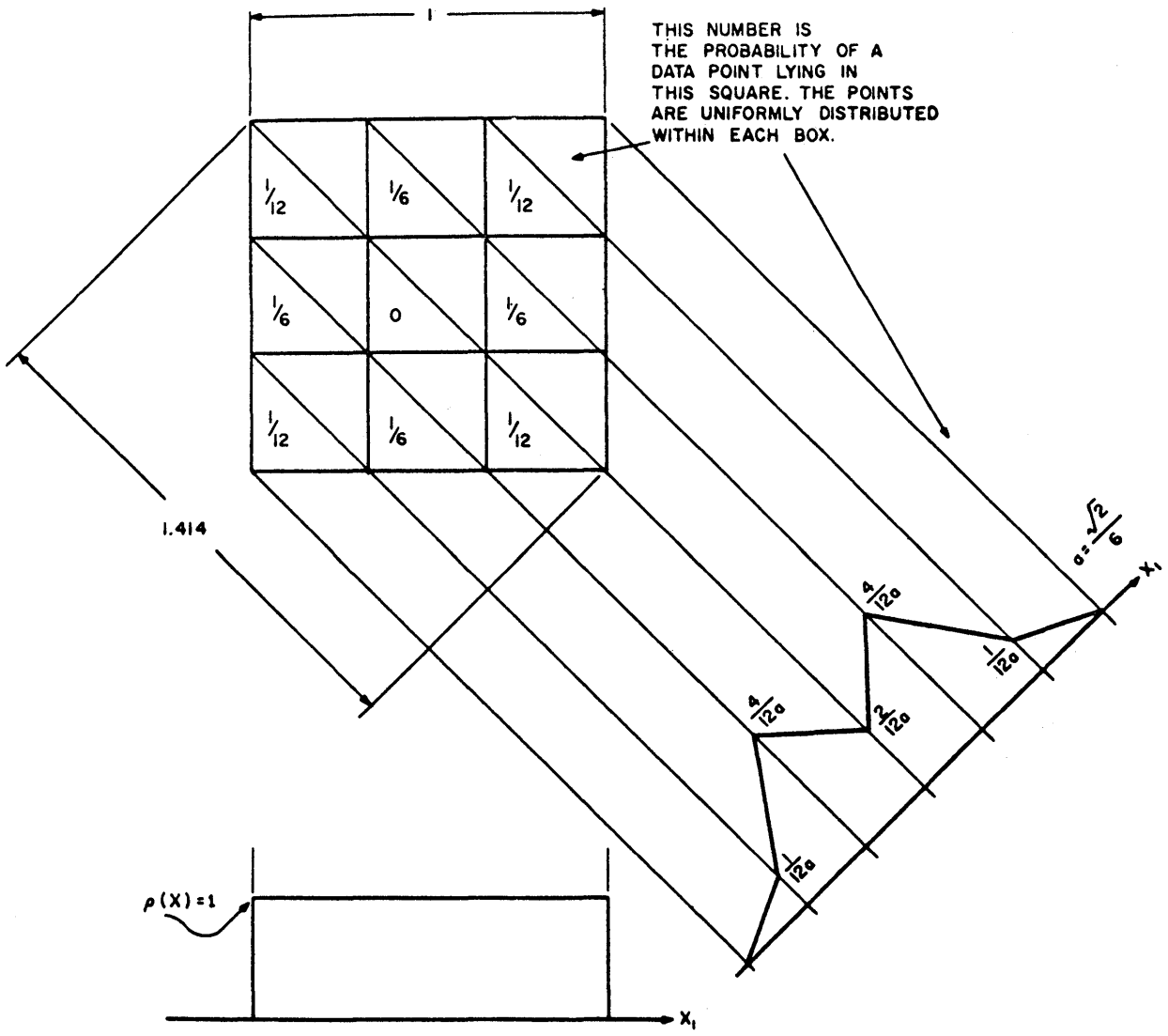


Fig. 2b A UNIFORM, TWO DIMENSIONAL DISTRIBUTION OF POINTS HAVING UNIFORM MARGINAL DISTRIBUTION

line 21: cluster points out of the original cluster. A second

line 22: allowable number of patterns in a cluster, the allowable maximum

p. 550, line 17, L: *Change to read*: around this by utilizing an arbitrary threshold. All

p. 552, lines 1-8, L: *Change to read*: normally distributed so that correlations have the desired significance. It seems quite probable that in cases in which a relatively large number of patterns is being used and in which a priori knowledge is not really adequate to divide these patterns into homogeneous subsets that this will not be true.

line 17, L: *Correct spelling of* phenomena

line 49, R: *Change to read* tween the technique of Mattson & Dammann[27] and that of Cooper

line 51, R: *Change to read* son and Dammann have taken the underlying philosophy of the

p. 553, line 2, L: *Should read* the Mattson and Dammann technique particularly useful. The most

p. 554, line 38, R: Alphnumeric should be alphanumeric

p. 555, L: *Change the following lines to*
line 12: distance of all patterns from this average. Set a
line 14: from this overall average) where $0 \leq k$. All
line 18: allowable cluster size is 1. The overall average of
line 19 L: *Add* all patterns would be used as the initial cluster point.

p. 558, line 11, L: *Add* n *to name* Dammann.

A. R. HOLMES and R. K. AUSMAN
Information Processing of Cancer Chemotherapy Data

p. 583, line 4, L: *Change* chemotheraphy *to* chemotherapy

p. 585, line 51, R: *Change* fransit *to* transit

p. 586, line 12, L: *Change* deliquent *to* delinquent
line 1, R: *Change* Institutes or *to* Institutes of
line 15, R: *Change* IBM1620 *to* IBM 1620

LEHMAN, SENZIG, AND LEE
Serial Arithmetic Techniques

p. 716, line 42, R: *Change* or *to* of

p. 718, line 4, L: *Should read* $(1 + \log_2 n) \cdot t$
line 5, L: *Delete* t

p. 721, line 35, L: *Should read* $b\prod_i m_i$
line 37, L: *Should read* $a\prod_i m_i$
line 42, L: *Should read* $\epsilon 2^{-2t_0}$
line 43, L: *Should read* $1 > |\delta| > 2^{-t_0}$ or $\delta = 0$
new line 44, L: *Change to* $1 > \epsilon \geq 0$
line 24, R: *Change* 4 *to* 5
line 25, R: *Change* 5 *to* 6

p. 725, line 3, L: *Change* 2. _____ *to* 2. M. Lehman

N. V. FINDLER
Human Decision Making . . .

p. 738, line 25 and 26, R: *Change to* Logic Theorist or the General Problem Solver

p. 739, line 19, R: *Change to* $\{y\} = F(\{x\})$
line 40, R: *Change to* to this, the concept

p. 741, line 40, R: *Change* behaps *to* perhaps

p. 743, line 6, L: *Change* ding on sich *to* Ding on sich
line 5, R: *Change* etc.; Hooke *to* etc., Hooke
last line, R: *Footnote omitted*:

*One subject, disregarding the fact that there were three independent variables instead of two, considered the environment as some terrain and the $\underline{C}$ values representing the height of mountains, hills, and valleys. His "hill-climbing" was a tourist's excursion.

p. 744, line 32, L: *Change* the one *to* the ones
last lines, L: *Footnote not needed*
line 29, R: Indent to start under "Well

p. 746, line 17, L: *Change* lowing *to* lowering

p. 748, last line, L, and first line, R: *Write last line of formula as*:
$+ 8.\sin k \frac{\pi}{4} + 10. \delta_{k,0(\mathrm{mod}\ 7)},$
line 7, R: *Change to*
$y_2 = 250 - 200.\ \sigma_{x_2,9} + 20.\cos \frac{1}{5}\pi +$

p. 749, Appendix III: *Text sections on left and right-hand sides incorrect relative to each other. Corrected* Appendix III *is reproduced on next pages.*

MAEDA, TAKESHIMA, and KOLK
High-Speed, Woven Read-Only Memory

p. 800: *Figures and text that should appear on this page appear on p.* 1034.

MAY, POWELL, and ARMSTRONG
A Thin Magnetic Film Computer Memory . . .

p. 804, line 1, R: *Should follow* line 12

D. E. RIPPY and D. E. HUMPHRIES
MAGIC-A Machine for Automatic Graphics . . .

p. 820, lines 35-37, R: *Change to* The Z Field specifies

(Continued page 180)

APPENDIX III

Comparison between Excerpts from a Representative Protocol and Computer Performance

In both instances, in the experiment and in its computer simulation, Task Environment 1 was used with the simplest cost function, in which $c_i = y_1$. Editorial remarks are put in square brackets.

The subject's initial search here could be classified as systematic with slight random components. He adopted the Mathematical Language of Representation without any hesitation. In Stage (A) he followed a scheme fairly thoroughly but was not satisfied with the rate of improvement and entered Stage (B). In this, he employed the so-called univariate method which consists of changing only one control variable at a time. The optimization takes place in terms of single control variables and at the end an attempt is made to specify a global optimum.

* * *

E: Can we start now?

S: O.K., let's try around zero, say, 2, 3 and 4. (I.e. $x_1=2$, $x_2=3$ and $x_3=4$; the first output followed: $c_1=42$, $c_2=77$, $c_3=90$, $\sum c=209$.) Well, let us try another type...(Interrupted by a new output: $c_1=42$, $c_2=83$, $c_3=90$, $\sum c=215$)...hmm, it's increased... O.K., let them be 45, 5, 4. (i.e. $x_1=45$, $x_2=5$, $x_3=4$; output $c_1=60$, $c_2=31$, $c_3=698$, $\sum c=789$.)

*

* In describing the computer's actions, instead of articulated English sentences, we can only refer to brief statements, such as EXTRAPOLATION, INTERPOLATION, ASPIRATION LEVEL NOW:, COUNTER NOW:, NEW POINT:, etc. These had been planted in the program to indicate at appropriate times what action the machine is taking. The following excerpts from the trace of the program (P) are approximately equivalent to the segments of human behavior at the left, almost paragraph by paragraph.

*

* * *

* INITIAL SEARCH TYPE 1.

* NEW POINT: X1= 3, X2= 47, X3= 49.
OUTPUT: C1=273, C2= 84, C3=196, SUM=553

* OUTPUT: C1=273, C2= 90, C3=196, SUM=559.

* (The noise appears here.)

* NEW POINT: X1= 4, X2= 2, X3= 0.
OUTPUT: C1= 36, C2= 98, C3=541, SUM=675.

* NEW POINT: X1= 5, X2= 48, X3= 3.
OUTPUT: C1=260, C2=102, C3=474, SUM=836.

*

*

Whoops, it's up...(interrupted by a new output: c_1=60, c_2=29, c_3=698, $\sum c$=787). Now, let them be 4, 48, 48. (Output: c_1=273, c_2=90, c_3=754, $\sum c$=1117)*

Oh, that's bad. 0, 50, 0. (Output: c_1=271, c_2=52, c_3=460, $\sum c$=783.)

Still no good, let's mix them further...(interrupted by output: c_1=271, c_2=50, c_3=460, $\sum c$=781)... Well, x_1=45, x_2=48, x_3=4. (Output: c_1=278, c_2=27, c_3=879, c=1184.) This is the worst so far.

Now how about the center? Say, 20, 25 and 30. (Output: c_1=173, c_2=83, c_3=617, $\sum c$=873.)

. . .

. . .

. . .

E: How did you pick these values? Can you explain it now?

S: Well, I thought I would first try all these corners...a sort of local search at the two ends of every x range, and also in its middle. This might give me some idea of how that sum behaves...It could, of course, go up and down in between...but that may not be important. I hope these functions (the hypothesis is formed!) do not oscillate too much...

. . .

. . .

OUTPUT: C1=260, C2=104, C3=474, SUM=838.

NEW POINT: X1= 24, X2= 27, X3= 22.

OUTPUT: C1=191, C2= 64, C3=589, SUM=844.

. . .

. . .

. . .

(P names nine points with coordinates either at the two ends of the ranges, 0 x 5 and 45 x 50, or in the middle, 20 x 30.)

. . .

. . .

. . .

. . .

. . .

THE MATHEMATICAL LANGUAGE OF REPRESENTATION IS ACCEPTED.

INITIAL ASPIRATION LEVEL: 8.

. . .

. . .

. . .

. . .

. . .

S: Now I have...how many...four, five reasonable points. The sums here are no worse than, say, 800. Let us be careful now...I want to give you a good x_1... It's better if I don't care about these continuous outputs now...Well, when x_1 was around zero, the sum was about 210; when it was 28, we had almost 600. So, how about...how about, say, if x_1 equals 10...that might hit the minimum...This kind of thing can give us, shall we say, 15 for x_2, and...well...I'm doing the same for x_3...O.K., let it be 20. (Output: c_1=134, c_2=108, c_3=736, $\sum c$=778.) It didn't do much good...O.K., let us include this point as well....

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

. . .

SELECTED SUBSET OF POINTS:

X1= 3, X2= 47, X3= 49, SUM=553;
X1= 4, X2= 2, X3= 0, SUM=675;
X1= 4, X2= 3, X3= 50, SUM=680;
X1= 46, X2= 4, X3= 1, SUM=779.

(P selects four points out of the nine "non-noisy ones, with the lowest total cost values.)

INTERPOLATION: X1=21; (=3+(46-3).553/(553+779))
INTERPOLATION: X2=22; (=2+(47-2).675/(675+553))
INTERPOLATION: X3=25. (=0+(50-0).675/(675+680))

NEW POINT: X1= 21, X2= 22, X3= 25.
OUTPUT: C1=130, C2= 97, C3=563, SUM=780.

OUTPUT: C1=130, C2=103, C3=563, SUM=786.

COUNTER NOW: 1.

(P fails to hit upon a sum better than at least 779, by interpolation. It is counting the number of failures.)

SELECTED SUBSET OF POINTS:

X1= 3, X2= 47, X3= 49, SUM=553;
X1= 4, X2= 2, X3= 0, SUM=675;
X1= 4, X2= 3, X3= 50, SUM=680;
X1= 46, X2= 4, X3= 1, SUM=779;
X1= 21, X2= 22, X3= 25, SUM=780;
X1= 5, X2= 48, X3= 5, SUM=804.

(P selects six points out of the ten "non-noisy" ones so far, with the lowest total cost values.)

. . .

. . .

. . .

S: I just can't get a better point...This is silly. Why don't I move off? Suppose, we just try 35, 40, 45 for a change.

. . .
. . .
. . .
. . .
. . .

S: (After hitting a point with $\sum c=581$) Here we are, that's reasonable...I see it now...This thing seems to move downwards, toward the smaller x_2's. Let us choose...well, x_1 was all right, let it stay at 8,...x_2 has to be down...down...say, 15 and x_3 stays in the middle, 25.

. . .
. . .
. . .

E: What is wrong?

S: I just don't get that sum any better. I somehow feel, I have to do something about the c's...I mean, about c_1, c_2, and c_3 separately. Maybe I should minimize c_1 only, at first. Let c_2 and c_3 be whatever they are, and then come back to them... Yes, that's what I am going to do. Minimize them, one by one, and then get them together again. Where was c_1 the best?

COUNTER NOW: 5.

RANDOM BREAK-OUT

ASPIRATION LEVEL NOW: 7.

NEW POINT: X1= 17, X2= 38, X3= 9.

(After five failures the aspiration level drops and the next point is specified at random.)

. . .
. . .
. . .

EXTRAPOLATION: X1=0;
EXTRAPOLATION: X2=48;
INTERPOLATION: X3=23.

NEW POINT: X1= 0, X2= 48, X3= 23.

. . .
. . .
. . .
. . .
. . .

ASPIRATION LEVEL NOW: 4.

ENTER STAGE (B).

POINTS WITH BEST C1:

X1= 4, X2= 2, X3= 0, C1= 36;
X1= 45, X2= 4, X3= 46, C1= 59;
X1= 32, X2= 11, X3= 17, C1=106;
X1= 21, X2= 22, X3= 25, C1=130.

(The difference between the inital and current aspiration levels is four, which fact makes P quit Stage (A) and enter Stage (B). P selects four points with the lowest c_1 values so far.)

. . .

. . .

. . .

S: Well, c_1 depends on x_2 and x_3, and a little bit, maybe a little bit, on x_1. And it also varies with time... Yes, low values of x_2 are all right...

. . .

. . .

. . .

. . .

. . .

S: You see, this was easy, to minimize c_2...I now know what it is like....

. . .

. . .

. . .

. . .

S: That c_3 is tough, I can't see much reason behind it. Somehow, when I don't move with the x's, it's small er, and when I change them a little, it jumps up...It is a funny variable....

. . .

. . .

. . .

. . .

° ° °

° . .

° . .

POINTS WITH BEST C1:

X1= 4, X2= 2, X3= 0, C1= 36;
X1= 18, X2= 5, X3= 2, C1= 52;
X1= 28, X2= 4, X3= 6, C1= 57.

(A further search produces these three best points with regard to c_1.)

. . .

. . .

. . .

POINTS WITH BEST C2:

X1= 35, X2= 11, X3=47, C2= 39;
X1= 45, X2= 38, X3= 3, C2= 47;
X1= 41, X2= 2, X3=13, C2= 51;
X1= 36, X2= 43, X3=30, C2=53.

(The four best points with regard to c_2.)

. . .

. . .

. . .

POINTS WITH BEST C3:

X1= 3, X2= 47, X3= 49, C3=196;
X1= 27, X2= 13, X3= 5, C3=343;
X1= 6, X2= 7, X3= 21, C3=352;
X1= 11, X2= 15, X3= 34, C3=355;
X1= 3, X2= 4, X3= 18, C3=374.

. . .

. . .

. . .

S: If I take, say, 36 for x_1, 5 for x_2 and...and a small value, say, 2 for x_3, I should get just about the minimum...I don't think I can do any better...not when this noise is on all the time. Output: c_1[1/] 48, c_2=21, c_3=408, Σc=477. End.

* * *

THE BEST POINT SO FAR:
X1= 28, X2= 2, X3= 47, SUM=463.

(P was cut off here.)

* * *

The following points may be worth mentioning with regard to the comparison between the above two records:

The simulation of both the results of and the reasoning behind the subject's decision making is fairly faithful, although the trial points are, of course, not identical. The only serious shortcoming of the model can be seen at Stage (B), when it does not notice the effect of large step sizes. Consequently, the program's best points with regard to c_3 just, so to speak, happen to be the best. The first point on the list (x_1=3, x_2=47, x_3=49, c_3=196) is "too good" and its weight causes the final selection mechanism to choose a 'global optimum' with a much too large x_3. (The x_3=47 value of the first point on the best-c_2-list was also guilty in this decision.)

The quality of the computer search for minimum was also very similar to the human one. The machine obtained a minimum of 463 after 68 trials, as contrasted with the subject's minimum of 477 after 59 trials.

the display characteristics for the associated X and Y coordinate fields.

p. 821, Fig. 3, last line: *Change* (short, med., long) *to* (solid, dashed)

p. 822, lines 40 & 44, L: *Change* 127_8 *to* 177_8
line 51, L: *Change* one bit *to* three bits

p. 824, Table 2: *Change left column pair heading to* Data List Before Delete

p. 826, line 19, L: *Change* 127_8 to 177_8

p. 827, lines 33-36, R: *Change first sentence of paragraph to* A typical delete subroutine also consists of two machine instructions. *Delete remainder of sentence* (lines 34-36)

p. 828, line 24, L: *Remove* s *from* blocks
line 34, L: *Add* s *to* instruction
line 42, L: *Change* if *to* of

p. 830, line 32, L: *Change to* the same central computer

A. BURNS
Hybrid Computer for Lunar Excursion . . .

pp. 899-900, Figs. 3 and 4: *Figures reversed (captions are correct)*

A. P. SAGE and R. W. BURT
Optimum Design and Error Analysis . . .

p. 903, line 10, L: *Change* t *to* T
eq. 3: *Change* x(T) *to* x(t)

p. 904, eq. 10: *Change* y(n—1)T to $y(\overline{n-1}T)$; *and* y'(n — 1)T *to* $y'(\overline{n-1}\ T)$

p. 909, eq. 20: *Change* cosdT *to* cosbT

p. 912, eq. 21: *Change* $\sum_{K=0}^{N}$ *to* $\sum_{k=0}^{N-1}$

eqs. 24 and 25: *Should be*

$$\underline{\lambda}_y\ (nT) = \nabla_y \underline{f}[\underline{y}_d\ (nT), \underline{P}]\ \underline{\lambda}_y\ (\overline{n+1}\ T) + R[\underline{y}_a\ (nT) - \underline{y}_d\ (nT)] \quad (24)$$

$$\underline{\lambda}_p\ (nT) = \nabla_p \underline{f}[\underline{y}_d\ (nT), \underline{P}]\ \underline{\lambda}_y\ (\overline{n+1}\ T) + \underline{\lambda}_p\ (\overline{n+1}\ T) \quad (25)$$

line 1, R: *Change* $\triangle$ *to* ∇

eq. 29: *Change* $\underline{C}_i(jT)$, $\underline{x}(jT)$ + *to* $<\underline{C}_i(jT), \underline{x}(jT)>$

p. 913, eq. 32: *Change* $\Omega^{q\div1}$ *to* Ω^{q+1}
eq. 34: *Change* $\Omega^{q+1}(nT)$ *to* $\Omega^{q+1}(jT)$

M. W. GOLDMAN
Design of a High Speed DDA

p. 930, line 26, L: ples equation *should be* pler equation
line 13, R: $\frac{1}{u}$ *should be* $\frac{1}{\bar{u}}$

p. 931, line 17, L: very *should be* vary
line 21, R: 1.000 *should be* 1000

p. 932, line 7, L: $d\frac{1}{u}$ *should be* $d\left(\frac{1}{u}\right)$
line 14, L: convential *should be* conventional
line 16, L: sole *should be* scale

p. 933, line 15, R: ½ × IT *should be* ½ × 1T

p. 934, line 10, L: *Replace all symbols* u *in* eq. (1) *with* $\bar{u}$; *Quantity* βr *should be under square root sign*
line 12, L: Period *after* range
line 1, R: *Replace all symbols* u *in* eq. (2) *with* $\bar{u}$; *delete* comma *and* ft *following integral*
line 2, R: *Replace all symbols* u *in* eq. (3) *with* $\bar{u}$
line 3, R: *Replace all symbols* u *in* eq. (4) *with* $\bar{u}$; *minus sign precedes quantity on right side*
line 6, R: *Replace* u_i *with* $\bar{u}_i$

p. 935, *Replace all symbols* u *with* $\bar{u}$
line 8, L: mixi- *should be* maxi-
line 18, R: *Quantity* βr *should be under square root sign*
Fig. 2: Integrator *marked* cos ψ *in upper right is* integrator 10

p. 936, *Replace all symbols* u *with* $\bar{u}$
line 3, L: *Quantity* βr *should be under square root sign*
line 18, L: 1Z *should be* 1/Z
line 24, R: *Quantity* x *should be under square root sign*

p. 937, line 2, L: *Ditto*
line 5, L: *Ditto and* $x = \frac{9}{4}$ *should be* $x = (\frac{9}{4})^2$
line 3, L: $d\frac{1}{x}$ *should be* $d\frac{1}{\bar{x}}$
Fig. 3: chapman's *in caption should be* Chapman's

p. 938, line 11, L: th *should be* the

p. 941, line 1, R: *Quantity* ½ *should be* ½y'
"Registers and adders" should be italicized and spaced as a heading
line 27, L: C ≡ no overflow *should be* $\bar{C}$ ≡ no overflow
line 30, L: ≡ yields *should be* ⇒ ≡ yields
line 43, L: (C -OUT-) *should be* (C ⇒ -OUT-)
bottom, R: following tabulation *should be*

tabulation following Table 1

p. 943, Table 7: *Should be replaced by*

Condition of Integrator	Input to Integrator			
	ΔX +	ΔX −	ΔY +	ΔY −
$Y_M = 0$; $Y_S = +$	DN	DN	$1 \rightarrow Y$ $+ \rightarrow Y_S$	$1 \rightarrow Y$ $- \rightarrow Y_S$
$Y_M = 0$; $Y_S = -$	DN	DN	$1 \rightarrow Y$ $+ \rightarrow Y_S$	$1 \rightarrow Y$ $- \rightarrow Y_S$
$Y_M \neq 0$; $Y_S = +$ $Y_M \neq$ MAX	$Y_T + R \rightarrow Q$ $Q_T \rightarrow R$ $C \Rightarrow +$ OUTPUT	$Y_c + R \rightarrow Q$ $Q_T \rightarrow R$ $\overline{C} \Rightarrow -$ OUTPUT	$Y_T + 1 \rightarrow Q$ $Q_T \rightarrow Y$	$Y_c + 1 \rightarrow Q$ $Q_c \rightarrow Y$
$Y_M \neq 0$; $Y_S = -$ $Y_M \neq$ MAX	$Y_c + R \rightarrow Q$ $Q_T \rightarrow R$ $\overline{C} \Rightarrow -$ OUTPUT	$Y_T + R \rightarrow Q$ $Q_T \rightarrow R$ $C \Rightarrow +$ OUTPUT	$Y_c + 1 \rightarrow Q$ $Q_c \rightarrow Y$	$Y_T + 1 \rightarrow Q$ $Q_T \rightarrow Y$
$Y_S = +$ $Y_M =$ MAX	$Y_T + R \rightarrow Q$ $Q_T \rightarrow R$ $C \Rightarrow +$ OUTPUT	$Y_c + R \rightarrow Q$ $Q_T \rightarrow R$ $\overline{C} \Rightarrow -$ OUTPUT	$1 \Rightarrow$ OVERFLOW	$Y_c + 1 \rightarrow Q$ $Q_c \rightarrow Y$
$Y_S = -$ $Y_M =$ MAX	$Y_c + R \rightarrow Q$ $Q_T \rightarrow R$ $\overline{C} \Rightarrow -$ OUTPUT	$Y_T + R \rightarrow Q$ $Q_T \rightarrow R$ $C \Rightarrow +$ OUTPUT	$Y_c + 1 \rightarrow Q$ $Q_c \rightarrow Y$	$1 \Rightarrow$ OVERFLOW

p. 946, Fig. 11: *Add caption*: Front and back view of typical logic cards
line 8, R: comparisn *should be* comparison

p. 947, Ref. 20: IEE *should be* IEEE

p. 949, Ref. 51: 161 *should be* 1961
Ref. 52: *Add* Ref. 53 *as follows*: 53. H. J. Weber, "Inertial Guidance System Uses Digital Integrator", *Space/ Aeronautics,* vol. 30, pp. 134-38 (Nov. 1958)

M. RASPANTI
Training for the No. 1 ESS

p. 967, line 1, R: *Change* last months *to* last six months

p. 968, line 8, R: *Change* repeating backtracking *to* repeating and backtracking

p. 969, line 4, R: Analogous *misspelled*

p. 970, line 40, L: *Change* approximate *to* appropriate

p. 973, line 2, R: *Change* respective *to* respectively

p. 975, line 38, R: *Change* X × Y *to* X × Y*
line 43, R: *Change* X □ N *to* X ↲ N*
line 44, R: *Change* of *to* by
line 45, R: *Change* X □ N *to* X ↳ N*

p. 976, line 4, L: *Change* W *to* ⌐↱ W Transfer to
line 13, R: *Change* x, y, and Z *to* X, Y and Z register
line 17, R: *Close* ORing

T. J. GILLIAN and P. B. PERSONS
High Speed Ferrite 2½ D Memory

p. 1017: Graph *in* Fig. 11 *should be interchanged with photo shown in* Fig. 15

J. T. QUATSE
STROBES: Shared-Time Repair . . .

p. 1066, line 31, R: *Change* reduce *to* require
line 33, R: *Change* reduce *to* require

p. 1068, diagram (Test Oscillator), top center portion: + 12 *should read* — 12

p. 1069, line 1, L: Par *should be* PAR

p. 1070, bottom, R: *should read*
STROBES IS NOW PREPARED FOR MM11 E
T

p. 1071, line 9, L: *Should read* T TSK, MEM, 2, 1;
line 15, R: *Should be underlined* STROBES IS NOW PREPARED FOR MM11, E
last paragraph, line 2: modules *should be* module
last paragraph line 4: modules *should be* module

p. 1072, line 3, L: *Deleted; should read* procedures which are essential to high system per-