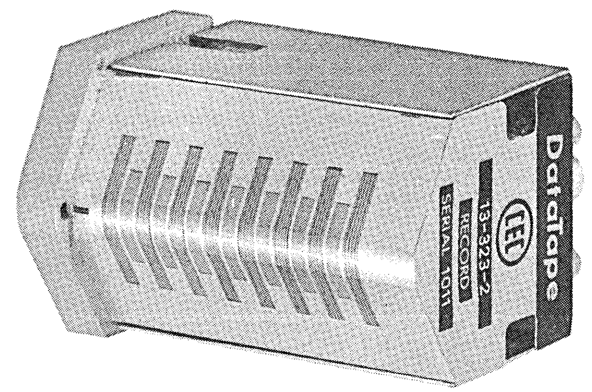
February, 1967

computers and automation

Testament to Computer Age: Automated Bible Distribution

More than 1000 hours from d-c to 1.5 MHz



Now try to get that through your head

You can . . . with a CEC Magnetic Recording Head.

In fact, considering that these recording heads last up to five times longer than any others, the 1000 hour guarantee seems rather conservative. In virtually every case, a CEC head will surpass a thousand hours with little or no indication of wear.

The reason: CEC recording heads are of a unique material and solid metal pole-tip design which completely eliminates the weakness of conventional lamination and rotary head designs.

This has resulted in superior performance at frequencies to 2 MHz and a minimum of wear and cleaning.

There is a CEC head for every analog and digital recorder.

Whatever make or model recorder you may now be using, CEC has the head to make the most of it, irrespective of the interface parameters. The "family" consists of more than 100 different recording heads, *all* with the same design advantages. Consequently, it is now possible to obtain state-of-the-art recording without replacing the basic instrument.

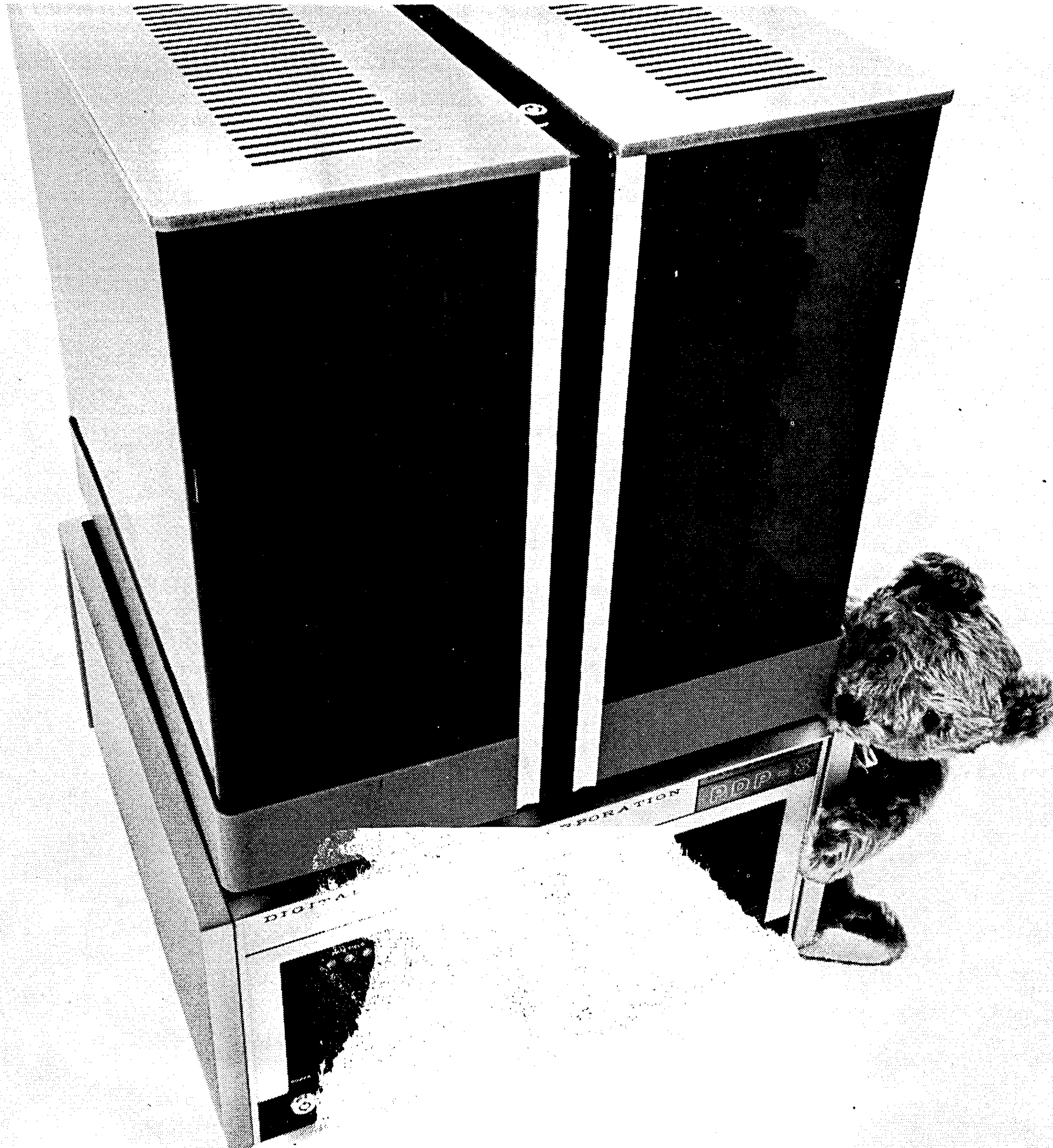
How were we able to achieve this breakthrough in head design? Through experience. CEC has not only been making magnetic recording heads longer than anyone else, but was the first to produce them on a quantity basis. *And* – because CEC uses *precision* machine construction where others rely on hand assembly.

For complete specifications and all the facts about this complete line of recording heads, call or write your nearest CEC field office.

CEC/DATATAPE PRODUCTS
Pasadena, California 91109

BELL & HOWELL

Designate No. 14 on Readers Service Card



“Can I ma er do the job?”

When buying a computer, the most i a scientist asks is: "Can I get it to do the

In physics, in chemistry, in life and earth sciences, there have been more than 500 yesses for the

More yesses than for any other real-time, on-lin eral purpose computer for science.

That should give you confidence. More than ther scientists have written programs that wor e potentialities (and limitations) of the PDP-8 have been exhaustively explored. There is an active and interested community of investigators ready to exchange on, techniques, even programs.

es PDP-8 the most immediately approachanputer you can buy.

Here are some other important questions — and answers: Q: How much? A: $18,000 complete. Q: When available? A: 30 to 90 days. Q: How much speed? A: 1.5 μsec. Q: How much memory? A: 4096 12 bit words. Q: Do I get a bear for added security? A: Only if requested.

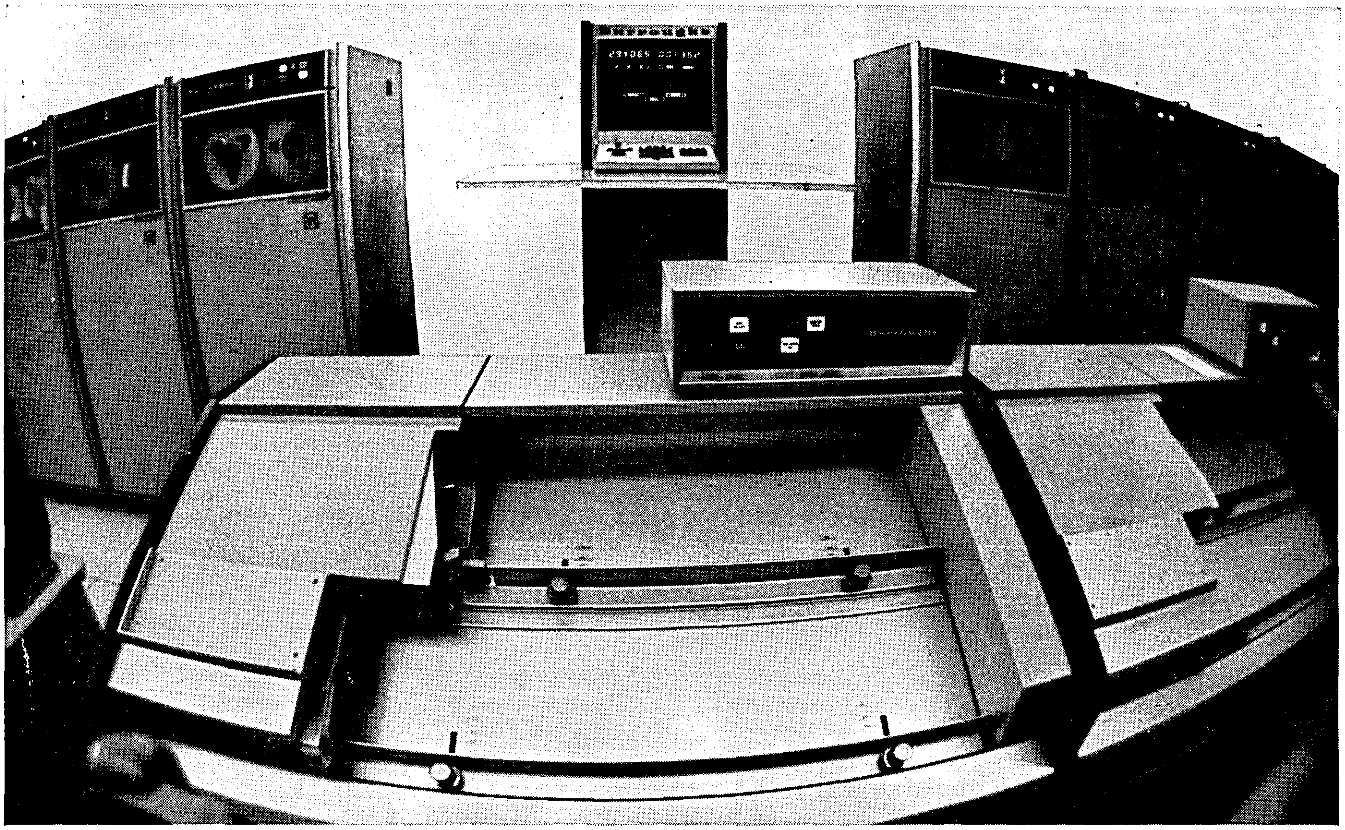
Send for your copy of our 540 page Small Computer Handbook and Primer. Free.

digital

COMPUTERS · MODULES

DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts 01754. Telephone: (617) 897-8821 • Cambridge, Mass. • New Haven • Washington, D.C. • Parsippany, N. J. • Rochester, N. Y. • Philadelphia • Huntsville • Pittsburgh • Chicago • Denver • Ann Arbor • Houston • Los Angeles • Palo Alto • Seattle • Carleton Place and Toronto, Ont. • Reading, England • Paris, France • Munich and Cologne, Germany • Sydney and West Perth, Australia • Modules distributed also through Allied Radio

Designate No. 7 on Readers Service Card



You get more work out of a Burroughs 500 system because more of the computer gets into the work.

That, in a nutshell, is how Burroughs 500 Systems solve two major problems of computer operation: **throughput** and **system utilization.** Their solution means a much higher ratio of performance to price—and a better return on your computer investment.

1. Throughput. In the past, the only way to increase throughput (the amount of work a computer system delivers in a given period of time) was to get a *bigger, faster computer*—at a sizeable increase in cost.

The Burroughs way is to provide a better organized computer that can do more than one job at a time. This pacesetting computer capability, called *multi*processing, has been enjoyed by Burroughs customers for nearly three years. Thousand-job-a-day installations are not unusual.

2. System utilization. You pay for the whole computer system; but chances are you use only a portion of it most of the time. Your computer has to be big enough to handle your biggest job. Whenever a smaller job is running, much of the system is idle because the typical computer can do only one job at a time.

The Burroughs method of *multi*processing combines these smaller jobs automatically, and runs as many of them together as the *full* size of your Burroughs computer will allow. No special programing or tricky operating procedures are required. It's all done by the computer itself, through its Master Control Program.

There are now five Burroughs 500 Systems, ranging from the small B 2500 to the super-scale B 8500. Perhaps one of them can improve your firm's computer operations and profitability. See your Burroughs representative, or write us at Detroit, Michigan 48232.

Burroughs Corporation

Designate No. 5 on Readers Service Card

The front cover shows the American Bible Society using a computer to handle the problems of distributing each week more than one million copies of its publications. For more information, see page 45.

computers and automation

FEBRUARY, 1967 Vol. 16, No. 2

Special Feature:
Software, Programming, and Future Developments

In Every Issue

IMPORTANT NOTICE TO SUBSCRIBERS IN THE UNITED STATES: Please look at your address label on this copy. IF YOUR ZIP CODE IS MISSING OR IS INCORRECT, please (1) attach your label to a reply card from this magazine, (2) write on the card your correct zip code, and (3) mail the postage paid reply card to us at once. Please note that your zip code must be for your street address or your post office box, whichever appears on your label.

COMPUTERS AND AUTOMATION IS PUBLISHED MONTHLY AT 815 WASHINGTON ST., NEWTONVILLE, MASS. 02160, BY BERKELEY ENTERPRISES, INC. PRINTED IN U.S.A. SUBSCRIPTION RATES: UNITED STATES, $15.00 FOR 1 YEAR, $29.00 FOR 2 YEARS, INCLUDING THE JUNE DIRECTORY ISSUE; CANADA, ADD 50c A YEAR FOR POSTAGE; FOREIGN, ADD $3.50 A YEAR FOR POSTAGE. ADDRESS ALL EDITORIAL AND SUBSCRIPTION MAIL TO BERKELEY ENTERPRISES, INC., 815 WASHINGTON ST., NEWTONVILLE, MASS., 02160. SECOND CLASS POSTAGE PAID AT BOSTON, MASS.

POSTMASTER: PLEASE SEND ALL FORMS 3579 TO BERKELEY ENTERPRISES, INC., 815 WASHINGTON ST., NEWTONVILLE, MASS. 02160. CHANGE OF ADDRESS: IF YOUR ADDRESS CHANGES, PLEASE SEND US BOTH YOUR NEW ADDRESS AND YOUR OLD ADDRESS (AS IT APPEARS ON THE MAGAZINE ADDRESS IMPRINT), AND ALLOW THREE WEEKS FOR THE CHANGE TO BE MADE.

c&a CAPITAL REPORT

The first section of the Post Office Department's source-data collection system, originally announced as a $22 million project and now estimated at $33.5 million, will start operating this month, according to Lawrence F. O'Brien, the Postmaster General. The Eastern seaboard section of the system is scheduled to go on the air as the first operational phase, and includes New York and the New England states.

The system will be used to collect data on post office operations and management functions, and should measurably speed mail service by freezing manpower which can be used elsewhere. Four cities are in the first trial-run network, and four more are three-quarters along the way to tie-in. O'Brien estimates that the system will pay for itself in approximately three years by saving man-hours now required for employee time reporting, part-time hiring, and reports on mail flow through the postal system.

Manual preparation of thousands of reporting forms will also be eliminated when the new system is fully operational, said O'Brien; he considers the main valuable aspect to be in improved management by one of the government's largest employers. The pre-Christmas pile-up and delay of tons of third and fourth class mail in the Northeast has speeded up P.O.'s ADP effort. St. Louis and Paramus, N.J., each with a CDC 3300, will eventually be linked with batteries of 1205's. Paramus, Rochester, Syracuse, Albany (N.Y.), and Springfield (Mass.), are scheduled to go on the air first.

The U.S. Army Combat Developments Command has let a contract to Technical Operations Research, Inc., for a $2.3 million project involving war gaming, weapons evaluation, material requirements, cost/effectiveness, field testing, and logistics. The work will be carried out at Fort Belvoir, Va., by their Systems Sciences Division headquartered in Arlington, Va. The initial contract funds 1967 effort on the work, scheduled to take 5 years to complete, by their Combat Operations Research Group.

The Federal Bureau of Investigation, long a user of punch card equipment, is putting two IBM 360/40's on the air this month in a year's pilot operation of a National Crime Information Center. It is scheduled to begin operations with 15 cooperating law enforcement agencies feeding information into, and getting replies out of, the central tandem 360's in their Washington headquarters. One 360/40 processes administrative work and serves as a back-up.

Currently, teletypes linked directly into the central processor and random-access files, will carry coded requests for interrogation, update, and purge of data for stolen cars and guns, wanted criminals, etc. Uniform codes have been developed with common formats to aid inexperienced remote users. The agency, in step with industry public relations practices, has announced that it is considering the use of satellites when that becomes practical.

First trials will use standard teletypes operating at 100 words per minute. Future plans call for transmitting and receiving video images of faces, fingerprints, and other visual data, with perhaps video receivers mounted in patrol cars. The trial network and operating system have been designed by the Institute for Telecommunication Sciences and Aeronomy in the Department of Commerce under a grant from the Office of Law Enforcement Assistance.

J. Edgar Hoover, still head of the FBI though now over 70, states that, ". . . the FBI has made certain that its computers are able to interface, or be compatible with, equipment produced by any major manufacturers . . . and that local police agencies can select any equipment they desire . . ." The FBI is also funding research on automatic classification and translation into machine language, of fingerprint data, using scanning devices.

To allay fears of invasion of privacy, which have been the subject of considerable recent controversy between Hoover and Robert Kennedy, he states that the system, ". . . means many things, all of them good . . . with no intrusion whatsoever upon the right to privacy . . . with a guarantee of the security of information in its files against access or removal by unauthorized persons."

How these guarantees can be met and accuracy maintained with input and output from, ". . . local officers . . . within . . . even the smallest police departments . . ." is open to some question. The ultimate effect in deterring crime is however likely to be beneficial.

The government's ban on shipment of large scientific computers to France has been lifted, and two CDC 6600's have been ordered, with one already shipped to the French Power Bureau, and another on order for S.I.A., a French service bureau.

The Department of Commerce, which controls exports of certain material to foreign countries, had previously banned shipment of the large-scale scientific hardware by U.S. firms. Although reasons for denial are rarely given, it was widely believed that the ban centered around France's possible use of the scientific machines to develop atomic weapons, which the U.S. hoped to slow.

In a related move, Bull-General Electric, G. E.'s French affiliate, discontinued the Gamma 140 and 145, eliminated about 200 jobs connected with their production, and cut back work weeks from 45 to 42½ hours.

IBM, American Telephone and Telegraph, ITT World Communications, Xerox, Honeywell, Business Equipment Manufacturers' Association, Informatics, Bunker-Ramo, and Western Union were among more than 20 organizations who responded to the call by the Federal Communications Commission for comments and opinions on rate structures and quality of service, connected with data communications associated with computing facilities. The FCC restated its promise, made last month, that it will proceed swiftly with the investigation. It hopes to identify items for the inquiry agenda within a short time so that it can determine if regulation of computing/communication links is necessary.

Senter Stuart
Washington, D.C.

Computer-Assisted Explanation in Programming

The special feature of this issue of "Computers and Automation" is "Programming, Software, and Future Developments." In this area, we can notice a great change happening in the computer field:

> The barrier to progress nowadays is not so much the limited capacity of the equipment as it is the limited capacity of the programming.

In other words, to make powerful computers is now much easier than to make full use of the powers of computers. We have acquired a sorcerer's machine: how are we as apprentices to use it well?

This problem is reflected in several facts:

- Often a third and sometimes half or more of the money spent for a computer installation is spent on programming and programmers.
- Persons to understand problems and systems and to program them for computers are far fewer than needed.

What is the way out? One important avenue for progress in understanding of programming is better explanation than exists nowadays of programming and related information. This is explanation that would enable a person to learn and understand in, say, half the time that he previously would have needed. Even though problem-oriented languages like FORTRAN or COBOL are available, still much of the time we have to use assembly language to achieve essential speed and versatility. We can't for long tolerate an inefficiency factor of 10, or 5, or even 3, for many classes of problems. We greatly need good explanation and much more understanding of the principles of programming in assembly language and many related subjects.

What is good explanation of programming?

According to the dictionary, to explain is to make plain or clear, to render intelligible. In other words, to explain an idea means to express that idea in terms of other ideas; it means expressing a strange idea in familiar relations to familiar ideas.

For example, what is LISP?

Explanation 1:

> LISP is a problem-oriented machine-independent computer-programming language particularly adapted to handling symbolic expressions.

But for you, the reader, to be satisfied with this, you need to understand already: problem-oriented; machine-independent; computer-programming language; symbolic expression. If you do not know the meaning of even one of these terms, you are thwarted. Also, you have to be satisfied with *not* understanding why the word "LISP" was chosen. (In fact, the mystique of "LISP" stimulates people at one time-shared computer system to put into their time-shared supervisor when you call LISP: "LITHP IS LITHTENING.")

Let's try again.

Explanation 2:

> LISP is a language in which you can quite easily write programs for computers; it is well adapted to expressing the conditions that occur in many kinds of problems; it is independent of any particular computing machine; it is able to handle letters, digits, numbers, words, statements, commands, tables, lists, and many other items, i.e. *expressions* composed of symbols of almost any kind. The name "LISP" comes from the first three letters of "LIST" and the first letter of "PROCESSING." As a programming language, it was worked out in 1958-60 at Massachusetts Institute of Technology by John McCarthy and several other computer scientists.

Which of these two explanations is more satisfactory? The second one. Why? Essentially, because it fulfils more of the needs of a broader audience. It conforms better with the maxim:

> Never underestimate a man's intelligence; never overestimate his information.

In fact, I often refer to a list of 35 common properties and relations which an explanation should be checked against. If you would like a copy of this list, designate 1 on the Readers Service Card.

When an explanation is successful, the explainee fully understands. What is understanding?

Most of the definitions given in the dictionary for "understanding" do not convey an operational meaning: they are synonyms like "grasping," "comprehending," etc. But among the definitions, we do find one which has an operational meaning: understanding is "the power to distinguish truth from falsehood and to adapt means to ends."

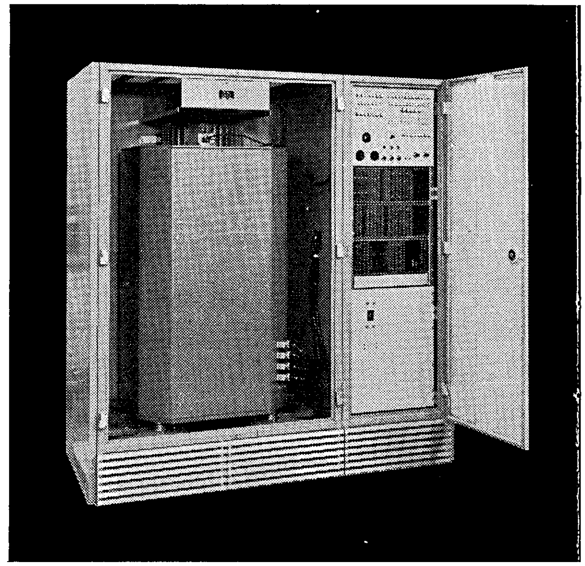
Now the understanding of programming can be demonstrated on a computer. To demonstrate that a programming explanation is good, has produced understanding, we can look to see if the person trying to understand it can do something that he was not able to do before. If you "understand," then your program runs; if you don't "understand," your program does not run.

In producing the understanding of programming and related subjects, (1) really good explanation of programming step by step, plus (2) direct access to a computer for experimenting and learning, can become seven league boots. The computer as a teacher has many powers and virtues. The computer can act as a device which gives explanations, offers examples, presents problems, suggests hints, and verifies understanding. The computer is infinitely patient, yet completely intolerant of errors; and it can be programmed to be courteous and friendly!

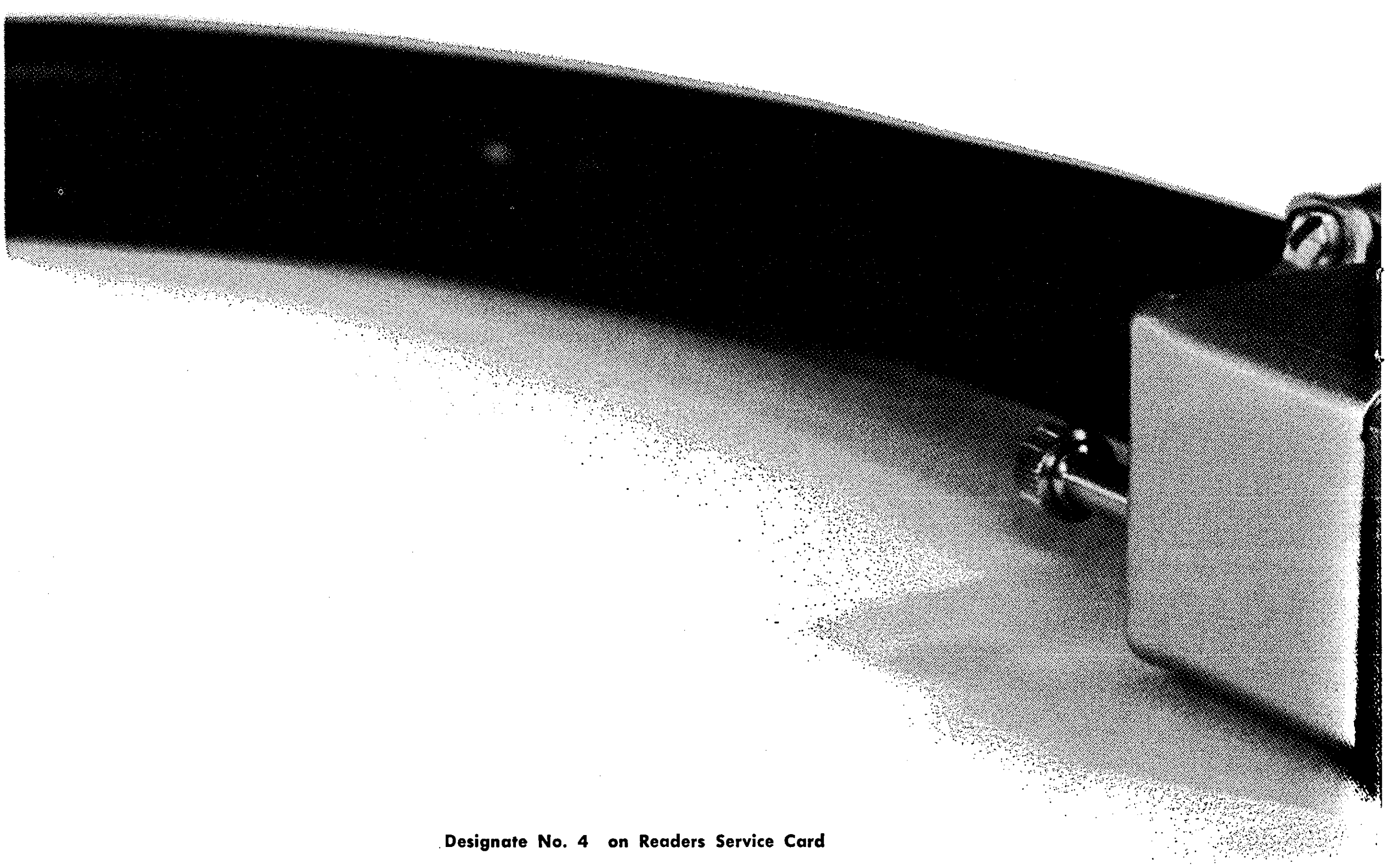
Once in a while a human being will be needed of course in order to answer the questions that the computer is deaf to or stupid about. But most of the time the computer, properly programmed to explain well, can be your explainer, your instructor, your solution-checker, and your expert in making clear how to program and produce software — and perhaps even, eventually, your guide, philosopher, and friend.

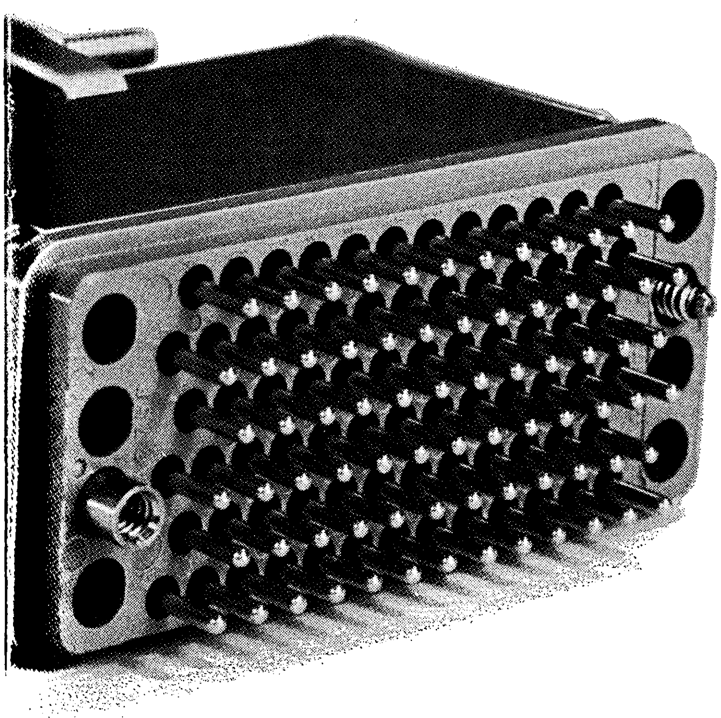
Edmund C. Berkeley
EDITOR

The Bryant Series XLO-1000 Controller



This is a plug for our new





MULTI-ACCESS FORUM

ALL-OUT PURSUIT OF ECONOMIC GROWTH TO CREATE JOBS

Excerpts from a speech by Thomas J. Watson, Jr., which appeared in: "Information Forum on Technology & Nation" Newark, Del. 19711

For a decade or more, many people have feared, with some justification, that technological change in our time is going so fast that it threatens a great many jobs while it eases the work burden of the vast majority. Every year fewer and fewer people produce more and more goods.

In fact, all through the peacetime years of the 1950s, despite the economy's continuing growth, unemployment climbed higher with every peak of the business cycle. Then starting with the recession of 1957-58, it climbed to a plateau and stayed there. For seven years, despite increases in everything else — production sales, profits, wages — unemployment never averaged less than 5 percent.

These were some of the facts and fears that led to the establishment of the Automation Commission by the Congress in August of 1964. Underlying all its subjects of inquiry was this basic question: do automation and technological advance represent a major threat to employment in the decade ahead and therefore require inhibiting controls or other action by the Federal Government or by someone else?

Despite a frank recognition of the problems of technological change, the answer to the question of controlling technology in one way or another was "No."

Having reached this conclusion, the Commission made a number of recommendations. Four major ones follow:

- Economic Growth: The Commission recommended an all-out pursuit of economic growth to heighten living standards for everyone and to create jobs — for the total of 93 million workers who will be our work force in 1975.
- Education: The Commission recommended, among other things, universal education for everybody in a trade school or community college; and "realistic access" to university education for "all qualified students."
- Employer of Last Resort: The Commission agreed that for the physically and mentally handicapped, the unskilled and uneducated, the victims of prejudice — the last people whom employers would hire — the Government should gradually and experimentally offer them jobs in hospitals, schools, urban renewal projects, etc.
- Negative Income Tax: The Commission suggested that the time has come to take a tough-minded look at the hodge-podge of payments to the poor and non-poor alike, and possibly revise it in order to make sure that the money will go where it will do the most good. The Commission did not recommend enactment of a negative income tax, but did urge Congress to study it as a possible solution.

Studies by the Commission have consistently reached two main conclusions:

- We should and must drive forward the thrust of technology as a great good for the United States and a great hope for the whole world;
- We should realistically face the problems which such advance creates and try innovative solutions where good sense tells us we must.

SOFTWARE GAP — A GROWING CRISIS FOR COMPUTERS

**Based on a report in Business Week for November 5, 1966, p. 127, published by:
McGraw Hill, Inc.
330 W. 42nd St.
New York, N. Y. 10036**

A shortage of programmers — and the fruits of their solitary art — is stunting the growth of computer use and costing industry hard cash. The supply of skilled computer personnel is far short of the demand. There are only about 120,000 programmers in the United States — and right now there's an estimated need for 175,000 or more. And the gap is widening. The number, power, and widespread application of computers has far outstripped the supply of programmers.

The computer has gotten so complex it is demanding ever more intricate instructions. Programs written for old machines must be rewritten for third-generation computers with integrated circuits. Businessmen fail to grasp the significance of programming or anticipate its cost. And even the computer makers have failed to anticipate the cost and time needed to develop complicated systems programs that control the computer's memory and peripheral equipment.

Programming is expensive and is likely to become more so. In 1964 the government spent 42% of its computer budget on software; in 1966 it will spend 51%. On a national basis, in 1960 software and hardware were equal in cost at about $1-billion. By 1970, it is predicted that software and programming services will reach an annual cost of $7-billion compared to $5-billion for hardware.

Over the years, the biggest share of the software tab has been paid by the computer customer, not the manufacturer. But the programming bottleneck has pinched the computer makers too. Users blame manufacturers for their unrealistic promises and delivery dates for software. Some users simply don't pay their bills until the new machine is running; others invoke a penalty clause in contracts.

The software shortage has given birth to a new group of companies. Their aim is to help business with programming, but there's a twist: software consultants are rapidly acquiring so much expertise in specific program skills that computer makers are their largest customer group. The field is burgeoning; new software companies are formed every day. But recruiting and training an adequate staff needed to get sizable contracts takes capital, and so does the service needed to keep customers happy. And the service provided is dependent upon the key element in the software gap — the new breed of intellectual artist — the programmer.

HOW SMALL CAN WE MAKE A BOOK?

**Based on an editorial in the December, 1966 issue of:
Current
905 Madison Ave.
New York, N.Y. 10021**

John R. Platt, a biophysicist at the University of Michigan, asks a deceptively simple question: How small can we make a book, and still read it?

One thing seems clear: at the rate new books are being published (there are 2,000-odd pages of print being produced each minute!), they cannot remain their conventional size. Library shelves are already bursting. One hundred years ago the Harvard library contained 212,050 volumes; the Yale library, 95,200. Today these two libraries have almost 8 and 5 million volumes respectively. As recently as 1940, only 9 university libraries in the United States had more than a million volumes; today that number has risen to 36.

One solution to the problem of the amount of space required for books lies in improved storage and retrieval systems: computerized catalogues, microfilms, electronic indices. It is already practical to reduce a page's area by 500 to 1,000 times so that an entire book can be printed on an ordinary catalogue card.

In the near future, using an electron microscope and tiny dots of metal-film, we will be able to store 1,000 books of 500 pages each on an area no larger than the proverbial head of a pin. At that point, microlibraries will become feasible. All the world's books could then be put on a desk top, and the living room archive will become commonplace.

But the problem doesn't end there. On the contrary, that's exactly where it begins. For once we have instant access to all knowledge, we will, more than ever, have to decide which knowledge is important.

Philip H. Abelson, the editor of "Science," stated it precisely in an editorial last November: "There is increasingly wide distribution of unevaluated material. The bottleneck in utilization of knowledge is not . . . inadequate information retrieval. The lag occurs in the step between the pile of books on a man's desk and the transfer of that information to his mind. We need to devote much more energy to determining what is significant and then conveying it in concentrated form."

To which we can only add, from beneath our piles of books and magazines, Amen.

PROJECT HINDSIGHT*

I. Based on a report by Philip H. Abelson, Editor, in Science, December 2, 1966, Vol. 154, No. 3753, Published by American Assoc. for the Advancement of Science
1515 Mass. Ave. N.W.
Washington, D.C.

In an effort to understand factors contributing to successful management of its research and development programs, the United States Department of Defense is conducting a retrospective study of the science and technology used in weapons systems. This effort, known as Project Hindsight, has been underway for 2½ years. About 20 proven weapons systems have been analyzed. Typically a team of five to ten expert scientists and engineers dissects the system into its subsystems and components, and identifies contributions from recent science and technology important to improved cost or effectiveness. Such a contribution is called an Event.

Once the Event has been recognized, its history is traced. It has been possible to identify the individuals who were principal contributors, their organizations, and the dates and circumstances under which the work was done. Eight percent of the Events are categorized as science, 92 percent as technology.

Of the science Events, the majority were applied research clearly oriented toward a DOD need. Most of the remainder were applied research with a commercial objective. Only two science Events were identified as arising from basic academic research. Nine per cent of all the Events, mostly in applied research and technology, were performed by universities.

About 90 percent of the federal funds for university research is furnished by mission-oriented agencies including the Department of Defense. Some of the conclusions from Project Hindsight reached to date are of relevance to such agencies and university scientists:

1) Contributions from recent (post-1945) undirected science to the systems studied appear to have been small.

2) The length of time to utilization of scientific findings is decreased when the scientist is working in areas related to the problems of his sponsor.

3) The efficient production of timely knowledge useful to a mission-oriented agency is most readily achieved when that agency funds and manages its own research programs.

Because of its unprecedented nature and impressive scope Project Hindsight is likely to be influential. The report implicitly raises questions concerning government support of academic research which university scientists will do well to consider.

---

*A report "Project Hindsight" is available from the Clearinghouse for Federal Scientific and Technical Information, 5285 Port Royal Road, Springfield, Virginia 22151.

II. From the Editor

Computer persons pursuing certain research projects for the Department of Defense, take note: The green light is changing to yellow and red.

C&A PROBLEM CORNER

Readers are invited to submit problems (and their solutions) for this column to: Problem Editor, Computers and Automation, 815 Washington St., Newtonville, Mass. 02160.

The following problem is submitted by Walter Penney of Greenbelt, Maryland. The solution will appear next month.

Problem 672: Coffee-Time Analysis

As soon as Hank Rivers and John Lawthorne had ordered their coffee, Hank began, "Things were hectic today! The big brass were in a huddle all day trying to decide what computer to buy. The choice had narrowed down to a BIVAC, RENN, DIGS 170 and M.C.P's new Y 219 machine."

John took a sip, and said, "I suppose there's always some trade-off in connection with these choices."

"Trade-off? That just isn't the word! The BIVAC was fastest, the RENN the most compact, and the DIGS 170 the cheapest. And to make things worse, no machine occupied the same rank in any two categories."

John took a pencil out of his pocket and began to write on the back of the check. "I assume the Y 219 was rejected since it wasn't best in anything."

"Far from it! Next to the RENN it was fastest, and, what seemed to carry a lot of weight with the non-scientific members of the panel, it was considerably cheaper than the BIVAC." Hank's expression left little doubt what he thought of non-scientists and their evaluations of computers.

Hank thought his companion had been doodling on the back of the check, but now he saw that a 3 by 4 array was beginning to take shape. John looked up from his scribbling. "I'll bet I can tell you how the various machines rank in speed, size and price."

"Don't see how you can do that; I haven't given you any figures. You must know these machines!"

"Not at all. I'll give you the relative standings of the machines in each category, and if I'm wrong, I'll pay for the coffee."

John showed the back of the check to Hank, who looked it over a few moments, then dug in his pocket, and summoned the waiter.

How did the machines compare?

(Solution next month)

"HONEY FROM NUMBERS"—COMMENTS

I. From Miss Georgia Nagle
Computation Center
Mass. Inst. of Technology
Cambridge, Mass. 02139

Congratulations on your delightful essay in the December "Computers and Automation" on numbers — from one who has never been able to cope with them in *any way.* I know the honey is there, but I've never been able to find it!

II. From the Editor, enclosing the December 1966 editorial, to
Monsieur Andre Maurois
Academie Francaise
Paris, France

Je prends du plaisir en vous envoyant quelque chose que j'ai écrite et que parle de vous et de vos idées — avec mes salutations distinguées.

III. From Andre Maurois

Je vous remercie de m'avoir envoyé cet article où vous me citez et où vous commentez mes propos de manière si intelligente et si neuve.

Bien cordialement à vous.

P.S. J'ai essayé 142857. It's amusing and easy to explain.

SUBSCRIPTION FULFILLMENT

I. From Margaret Milligan, Publisher
Data Processing Digest, Inc.
Los Angeles, Calif. 90035

We laughed ourselves sick (with sympathy) over your editorial about the subscription fulfillment problems in your November issue. No one who has never been in the situation could appreciate it as we do. How companies, and even government agencies, can send checks with no identification or accompanying documents is beyond me. And the agencies can get their clients' orders mixed up with amazing regularity. I guess we could exchange exasperating experiences for hours.

We still have less than 5000 subscribers, so we are not in the difficult position you are in, since ours is still managed entirely manually, and we can maintain cross files to help us locate people. But it is getting worse, and it is impossible to find good people to handle the problems (which we call bags of worms).

I would like to know something about your computer operation. We have been investigating the possibility of going on computer because the addressing plates available to us are so limited in character length. If you have the time, I hope you can give me a little information about your system.

II. From the Editor

Your letter delighted me.

By now you should have a letter from our computer fulfillment manager, and I hope he has answered your present questions.

IEEE COMPUTER CONFERENCE — CALL FOR PAPERS

Professor S. S. Yau
Dept. of Electrical Engineering
The Technological Institute
Northwestern University
Evanston, Ill. 60201

The First Annual Computer Conference of the Institute of Electrical and Electronics Engineers will be held in Chicago, Ill., Sept. 6-8, 1967. Papers describing results of original research in the following areas (or related topics) are invited.

1. Design Automation:
   This area includes automated design of electronics circuits, filters, logic circuits, large integrated arrays, computer systems, etc.
2. New Computer Elements and New Computer System Organizations:
   This area includes use of microelectronics, integrated electronics, optical devices, and other new elements in computer design, and future computer system organizations, etc.
3. Reliability:
   This area includes design of reliable computer systems from unreliable elements, organizations of reliable automata, self-fault-detection, self-diagnosis, self-repair, etc.
4. Pattern Recognition:
   This area includes new techniques for pattern recognition, pattern visualization, and description of their implementation, etc.
5. On-Line Computer Systems and Their Applications:
   This area includes on-line control of experimental equipment, industrial process control, inventory control, large message switching and circuit switching systems, etc.

Authors are requested to submit seven copies of a 75-word abstract and a 2000-word synopsis with any important illustrations, and a list of references to me by April 10, 1967. The author should arrange for any necessary company or security clearances before submission of his manuscript. Authors will be notified of acceptance or rejection of their papers by May 31, 1967. The deadline for submitting the completed manuscripts is September 6, 1967.

C OF P = CITY OF PRESCOTT

I. From the editorial "Subscription Fulfillment", Computers and Automation for November, 1966

As I write this editorial, I have in front of me a voucher which came off a check. The date is typed "JUN 20, 66." The "account number" is typed "103986," a number which does not agree with any invoice number we have. The "invoice amount" is typed $4.00. The "remarks" are typed [sic]:

subscription to computers & automation, November, 1966

At the bottom of the voucher, printed, is "C of P 14." On the voucher there is no name of a bank, nor name of an organization, nor name of a subscriber, nor any address. What do you do? We have no way of crediting this $4 to anybody. We have no way of inquiring of anybody whose $4 this is. (It is beyond my understanding how any modern organization could use printed vouchers for its checks that do not bear any name or address!)

II. From Donald E. Wilson, CPA
City Treasurer
P. O. Box 2059
City of Prescott, Arizona 86301

As I write this letter, I have in front of me your editorial in the November, 1966 issue of "Computers & Automation." As a matter of fact, I have enclosed a marked copy of that page, outlining this embarrassing subject in red. I am also enclosing copies of the voucher to which you refer, plus a copy of the duplicate payment which resulted from all this.

Had we issued a purchase order for this, as is our customary practice, we would not have had the duplicate payment. Had we had the City name and address printed on the voucher, this wouldn't have happened. Had we recorded your invoice number on the original check, this would have been avoided.

We now have on order a NCR500 system, the planning for which made your magazine desirable. I read each copy thoroughly, and have gotten much from it. As a matter of fact, I even read the editorial, as you can see.

Oh, by the way, the account number is that of our own accounting system. The checks are prepared on our present equipment, a NCR class 33 accounting machine. Perhaps our new computer will help to eliminate these deficiencies.

I will appreciate your returning the duplicate payment of $4.00 at your earliest convenience.

III. From the Editor

Thank you for your letter. It gives me much satisfaction to return to you as City Treasurer the $4 we owe you. I can see from your letter that this sort of problem, which has given us both such difficulty, is on its way out for the City of Prescott, Arizona.

LOST SUBSCRIBER SEARCH OPERATION

I. From Roger H. Geeslin
Assoc. Prof. of Mathematics
University of Louisville
Louisville, Kentucky 40208

My copy of the November issue of Computers and Automation had a second label affixed, addressed to the Library, Belknap Campus, University of Louisville. I assume this means that the Library did not receive their copy. Please send them one.

Also, re your subscription fulfillment editorial, is it too much to expect you to have a cross reference list by *name* of your subscribers?

II. From the Editor

Thank you for your letter, and for noticing and returning the second label pasted in error on your copy. Thanks to you, we can correct this error, and have done so by sending a copy of the issue to your library.

As a result of your suggestion and some analysis of possibilities with our magnetic tape file of subscribers, we are installing a "Lost Subscriber" search operation to look for "lost" subscribers. Although our file of subscribers is sequenced according to their geographic location, at a reasonable cost we can run an alphabetic search every second month for names not found geographically.

We shall begin this soon experimentally and hope in this way to reunite certain lost subscribers and the subscription record to "Computers and Automation."

Just the same, we still request that all correspondence about subscriptions should include complete name and address, present and previous.

765 AND 567

I. From William E. Roberts
Fort Worth, Texas 76112

This is a fast world we live in. Everything seems to be accelerating — even the rate of acceleration. I know that our newest jets flying west can beat the sun and thus arrive at their destination before they leave (timewise) from the point of departure. This doesn't upset me, probably because I haven't done any of that kind of jet flying yet. But, what does upset me is when my subscription to "Computers and Automation" expires before it starts.

In July of this year I subscribed to your magazine for a period of one year. I made a special request, however, and this is probably how things got fouled up. I asked that, if possible, my subscription should start with the June Directory issue. I preferred to have the Directory then instead of waiting for the 1967 Directory issue.

I didn't get the June 1966 Directory issue nor the July 1966 issue but did get the August 1966 issue. This was understandable as I figured it was too late and too troublesome to back the thing up as much as I had requested. But, the August 1966 is the only copy I have ever received.

I took a look at the subscription sticker on that one lonely issue I received and, sure enough, it appears to be coded to expire before the subscription was to start. The code is *DO 765. Usually these things are set up to give the actual expiration date. If that is the way your system is programmed, my subscription was set up to expire *before* I decided to subscribe.

In any event I really would like to receive your magazine and sincerely hope that someone around your shop can straighten it out.

Don't feel too badly, though. I have had similar trouble with other publications and have about reached the conclusion that one of two situations must exist; either I am "snake-bit" when it comes to magazine subscriptions or magazine subscription automation is a poorly developed art that is fraught with errors.

II. From the Editor

Thank you for your nice letter. From our office we are shipping you the missing issues right now.

We are sending your letter to our fulfillment office to have them correct your subscription record, and send you February through May 1967 in due course. May 1967 should be the right expiry. The code on your sub apparently changed from 567 to 765, due to Murphy's First Law, "If something can go wrong, it will." If there is any discrepancy, we shall get in touch with you again.

Thank you again for your friendliness towards us, in spite of the poor functioning of our subscription fulfillment art in your case.

CORRECTION

On page 9 of the January, 1967 issue of "Computers and Automation," the credit line for the article entitled "BOOK" should be amended to include: "Reprinted by permission of *Punch.*"

TIME-SHARED COMPUTER ACCESS — C&A CONTEST WINNER

"Computers and Automation" is pleased to announce the winner of the Readers Service Card drawing in the contest for a chance to use a time-shared computer free. This contest was announced on page 55 of the October, 1966, issue of "Computers and Automation". The winner is Mr. Francis J. Bartram of the Shell Oil Company, New York, N.Y. Mr. Bartram has won $100 worth (approximately 7 hours) of time-shared computer console time plus a free installation for the period of use. The winner has the right to delegate his prize if he is not in a position to use it.

The contest was sponsored jointly by "Computers and Automation" and the "Information Processing Centers Business" of General Electric, Computer Department, Phoenix, Arizona. C&A is providing the quantum of computer time; General Electric is providing the teletypewriter installation charge and rental, the telephone line charge, and the time for instruction to the winner.

We hope to publish soon a first-hand account of the lucky user's experiences and reactions with the time-shared computer.

ZERO-UN INFORMATIQUE

I. From Pierre-Jean Refregier

Editor
Zero-Un Informatique
Paris 10e, France

To read practically every publication dealing with EDP is a professional obligation for myself as well as for my staff. I have been wanting to tell you for some time that although we are "blasé", each number of "Computers and Automation" is eagerly awaited by us, because we believe that in its own style, it is the best journal in the EDP field.

I should even say that "Computers and Automation" is, in some respects, a model for our own publications: "Zéro-Un-Informatique" (monthly) and "Zéro-Un-Actualités" (fortnightly). Both are the only French EDP journals which are completely independent from any outside influence. Like you we think not only in terms of machines, but also in terms of men.

I should welcome any form of cooperation with you which you may think suitable.

It is my belief that it would be particularly interesting to acquaint French EDP circles with your ideas as regards the social and human repercussions of progress in the EDP field. As you certainly know, France is at present actively — I should even say, enthusiastically — interested in computers, but action in this field is somewhat chaotic for the time being.

II. From the Editor

We are glad that you like "Computers and Automation." Your proposal of cooperation is interesting to us, and we shall be glad to be of help to you if we can. It is quite possible that we have many common interests and purposes, particularly in producing understanding of the social implications of computers and automation and the social responsibilities of computer people.

MAS DE 1000 AREAS DE UTILIZACION DE LOS COMPUTADORES ELECTRONICOS

G. L. Forgnoni
Univac de Mexico
Mexico 7, D.F.
Mexico

We are enclosing some copies of "Over 1000 Applications of Computers" translated into Spanish, in accordance with the permission you gave us in your recent letter.

The distribution to computer users in Mexico and to Universities started in December.

Please accept again our thanks for your authorization.

There are three things to remember about Sigma 5.

First, it is multi-use, like its big brother Sigma 7. It does foreground real-time control, background general-purpose computation, and high-speed input/output. Simultaneously.

Second, Sigma 5's central processor is so powerful and sophisticated that it works even more efficiently with $500,000 worth of memory, peripherals and options than it does in its basic $90,000 configuration.

Third, Sigma 5 grows without fuss. Everything is modular—memory, input/output processors, peripherals, central processor options, software. Each upward transition is simple and logical until you reach Sigma 5's limit. Then if you want to keep on growing, just substitute a Sigma 7 CPU and behold! You have a Sigma 7 time-sharing system.

Two computers aren't better than one.

It no longer makes economic sense to have different computers for different kinds of jobs.

Sigma 5 will control your plant, do your scientific and engineering computation, and look after your accounting and inventory. All at once.

Nobody will have to stand in line. Everyone will be protected against loss or interference. Sigma 5 can deal with foreground real-time interrupts in 6 microseconds without losing control of any of its other jobs, yet every background user will get his answers faster than he needs them.

Nobody loves a lazy computer.

If your Sigma 5 ever has an idle microsecond it will be your fault.

Sigma 5 doesn't stop computing to wait for input/output. It doesn't reserve idle capacity to take care of on-line peaks.

Instead it dynamically and very rapidly shifts from one task to another in order to keep all its power working all the time. Input/output is managed independently by one built-in and five optional I/O processors, with up to 160 automatic I/O channels.

The bigger the better.

One reason why Sigma 5 gets more efficient as it grows larger is that when memory modules are added interleave and overlap occur. This not only increases the effective speed of the central processor but raises input/output capability too.

Sigma 5 won't wait for software.

Sigma 5 is program-compatible with Sigma 7. The software for both has already been developed and is being delivered now with Sigma 7's.

So you won't have problems with new and untried software. And you won't have to wait either.

Sigma 5 software is modular like the hardware. As your Sigma 5 grows the software grows too, in natural, simple, logical steps.

Software for Sigma 5 includes Basic Control Monitor, Batch Processing Monitor, three ASA-compatible FORTRAN IV compilers including a high-efficiency version, Symbol Assembler, Meta-Symbol Extended Assembler, SDS COBOL 65, Sort/Merge, and a library of mathematical, business and utility routines.

All Sigma 5 software is multi-use.

Here are some numbers.

Basic memory cycle time of 850 nanoseconds is reduced as overlapping of memory occurs.

32 memory sizes, from 4,096 to 131,072 words.

Memory uses 32-bit words, is addressable and alterable by 8-bit bytes, halfwords, words, and doublewords.

16 general purpose registers, expandable to 256. All memory is directly addressable without base registers.

8 automatic I/O channels in CPU, plus 5 optional I/O processors, either multiplexor IOP's (500,000 bytes/sec.) or selector IOP's (37 million bits/sec.) in any desired combination.

224 levels of priority interrupt. Priorities automatically recognized without program intervention.

How soon?

Order now and you'll get your Sigma 5 (with software) in August.

You can order the exact configuration that fits your needs. Or if you have budget problems, order what you can afford now and plan to add modules as more funds become available.

Growing with Sigma 5 is almost painless. All you feel is a slight twinge in the pocketbook.

SDS

Scientific Data Systems, Santa Monica, California

Designate No. 12 on Readers Service Card

SOFTWARE DEVELOPMENT AND ITS COSTS

Carl H. Reynolds, President
Computer Usage Company, Inc.
Mt. Kisco, N.Y. 10549

"It is impossible for us to see anything which we are unable to do with computers. Therefore, something of the human being inside of us takes over and says we must try to do at least part of what we can see as the possibilities."

In the past three or four years I have done a lot of work in development of software. I have met a lot of programmers; worked with a lot of programmers under both good and bad conditions; and have now decided to devote my life fully to the development of software. These past four or five years have started to synthesize in my mind some true statements about programming development and that lead to the so-called "software crisis." In this article I shall go over some of these points and, hopefully, this battle-scarred veteran will be able to give you a picture of the overall war which will be useful in developing the strategies you need for your own ends.

My subject is the costs of programming, development, and maintenance, and the payoff from software development. After looking quite deeply into the cut and dried figures of the matter, I have come to the conclusion that they were really beside the point; that understanding and controlling costs and pay-off of software development required an understanding in a direction different from accounting for or measuring the physical process of writing specifications, coding, and debugging.

Three Ideas to Understand Programming Development

The software area is in confusion today because there are very few fundamental concepts on which to base any assessment of the situation. It takes at least three ideas to understand the programming development process.

First of all one must have some method of evaluating the program. Generally it seems to me that we are talking about the software problems in the same way that we would tell the designers of the Volkswagen that they have a poor product because it doesn't go fast enough, it isn't big enough, and it costs a little too much. All of this without letting the VW designer know how fast is fast, how big is big, and how little is little. This is a problem of great magnitude, but which I will leave for others to discuss.

The second thing that we need to understand is that programming has a technological base just as hardware does. The technology that is used in particular programming can be in various stages of development. For example, it can be in research; it can be in the state-of-the-art; or it can be in available technology. It is difficult to be very specific about these definitions, but let's try.

Defining "Research"

In hardware technology, to define research is easy. It is work done in the laboratory by one or two people. It may be advertised widely in the newspapers as being the savior of everything in mankind, but nobody in his right mind will take an order for that device nor will a responsible manufacturing man say he can build them in quantity.

Defining "State of the Art"

The second stage we could call State-of-the-art. That is, a lot of people and a lot of laboratories can do this and they have made pilot production attempts and have gotten some data on what it costs to produce this item.

Defining "Available Technology"

Finally, it is Available Technology which means lots of people understand the process, many people are, in fact, producing this item, and almost anybody can go into the business with a fair understanding of what his costs and profits will be.

The Same States in Programming

I contend that in programming the same states exist, but they have never been differentiated or understood by perhaps even the programmers themselves — certainly not by their management, and certainly not by the customers. We can differentiate technology in programming in the same way.

Programming is based on research technology if only one or two people have tried to do this particular thing and have achieved at least some small success. State-of-the-art technology can be defined as a program performing a job which has been done by, say, half a dozen independent people with success at least to the extent that results generally determine the technical feasibility of what they are doing. For example, time-sharing a year ago was in this state. Half a dozen people had produced systems of general-purpose time-sharing nature but which had limitations with respect to their economies and their functional capabilities. Nevertheless, the general idea had been proved.

Finally, there is the state in which a particular program can be written by almost any competent person in the field. For example, today a company like Computer Usage Corp. can produce a FORTRAN compiler for standard hardware for a very nominal sum.

Application Technology

The third idea is that the technology I have been discussing really has two separate parts. Although the parts interact extensively, I think it is useful to try to differentiate them. First is what I call Application Technology. That is, how well do we understand what we are trying to accomplish with the program? The second I call Implementation Technology. Perhaps a couple of examples will clarify this idea.

I consider that writing a compiler is really an application program. I do not want to differentiate between so-called "systems programs" and applications programs. I am sure that there are many differences and the costs are different, but the concepts I am discussing here today are the same. I consider a FORTRAN compiler an application just as I consider a program to process payroll an application program. I would say, with respect to payroll, that we have that application pretty well defined. We know what the input is and we know what we want to get out. The government changes that periodically, but those are small perturbations and they are usually fairly well defined when they come out. They usually just reduce the output in terms of both time on the machine and in dollars in the pocket. FORTRAN, I believe, is in the same state. We know what the language is, pretty much. It has been implemented many times. We know what the output should be.

Management Information Systems

Management Information Systems today are not a well-defined application. Aside from saying that a manager shouldn't have to think, we know only in vague terms what a management information system is to provide. Now I don't mean to say that there aren't well defined applications which do specific, useful things for management. There are, and they can be classified as management information systems. But too often today, the generic term management information system, refers to "terminal in the office, push the button, and get the answer to the competitive problem or next year's profit."

Implementation Technology

In addition to application technology, there is a technology for writing a program for a particular class or generation of machine. Third generation equipment, by and large, is quite a bit different in organizational structure and is absolutely different in detail from the second generation machines. This means that the technology of doing things with new machines goes through a technological development stage; for example, there has been essentially a mass movement by one segment of the using public from tape storage to disc storage. The techniques of dealing with random-access devices are entirely different from those in dealing with tape. The bulk of people today have experience in and have put applications on tape and they know the problems. Without fail, they will have trouble in achieving their desires when they start using discs instead of tape.

Similarly, many people today are card-oriented. They are going to have a difficulty in revising their approaches and making their plans and requirements fit the different systems' storage medium.

The impact and importance of these technologies is shown by the history of IBSYS. IBSYS is the software support for the IBM 7090 series computers. In the 18 months after these programs were first delivered, the thruput for average size jobs was improved by more than a factor of 10! This was not because of improvements in the hardware — it resulted from the improvement of implementation technology, and improved understanding of the operating-system application technology.

Controlling Programming Costs

In four years of fulltime deep involvement in this field, the most commonly heard remark is that to get a handle on programming we have got to get a handle on a programmer. I go back and say that to get a handle on a cost we have to get a handle on what we are trying to do. I'll give an algorithm for determining the state of technological development of a particular program. If it is possible to write specifications for what is required in terms of what goes in and what comes out, if it is possible to define the performance parameters of the program, and if it is possible to provide numbers for those performance parameters based on measurement of existing programs on existing equipment, then one thing is certain: your application technology is available. It is very highly probable that your implementation technology is available or very close to it. I want you to be very careful to notice that I said it has to be measurable. Something like three years ago we set down what almost all the components in OS/360 ought to have in the way of performance, and we did this in some detail. My guess is that it is only in the last year or so that measurements have generally been available on existing components so that the targets could be verified and understood.

Now the problem in controlling programming development costs come down to the fact that the costs go just completely out of control when it is expected that the results of taking a state-of-the-art application and implementation technology will, in fact, produce results as good as using both application and implementation available technologies. This single factor is what accounts for the astronomical increase over early estimates of the costs of programming.

Now, third-generation programming to go with third-generation hardware has a couple of problems. I am talking about inherent problems, not delivery problems. Let's take a look at what third-generation software is trying to do. In this case I'm restricting the more specific term of software to mean specifically that provided by manufacturers. In general, major strides have been attempted in three areas in the third-generation software.

Monitors

First, in the monitor area, major efforts have been made to make them multi-programming. The reason for this is that there is lots of core available and the machines run so fast that it is inconceivable that any one program can keep them occupied very long. The computer design is such that it permits overlap of many kinds of operations. Thus, to fully exploit the machine designs it is necessary to permit multi-programming; i.e., that at more than one program be in memory at the same time. Execution can be switched between these programs at micro-second speed depending upon their changing requirements for system components.

Second, most of the monitors are attempting to handle with the back of their hand real-time or more specifically communications-time problems. This is stimulated again by the machines' capability and a universal need for faster transit time for certain bits of information. This need, and the reduction in costs of communication, have greatly pushed the development of communications technology. Everybody needs to get something faster than they used to get out of this machine. I contend that both of these attempts individually were no better than state-of-the-art on second-generation equipment; i.e., there were very few people who adequately could put these on second-generation equipment. They are just now approaching available technology in the third-generation equipment as the first implementations are coming out.

Languages

The second area of development is in languages. Now it is true that FORTRAN and COBOL are coming to fruition as technological developments in the third generation and the only problem facing them is the transition to the third-generation machinery. I believe that they can safely be said to be above the available technology stage. FORTRAN made it on the second generation and COBOL just barely did. PL/1 is probably about to move from the state-of-the-art to available. It isn't clear to me at all, because I haven't worked recently with the language myself, how long one can expect the transition to take. It is clear to me that FORTRAN and COBOL are inadequate for today's problems.

This point was brought home to me forcefully just last month when I tried to write a program to produce my '67 budget. Our business is fairly complicated in some senses, even though it is easy in others, and we do an awful lot of calculation to determine the guess as what our revenue and costs are going to be. It is based on the cost of people and the sales price of people, and so forth. In going through this, a program to do these calculations written in FORTRAN took something like 20 statements. In COBOL it took about 150. The difference, of course, is that if you did the calculations in FORTRAN you couldn't read them in and print them out. If you did them in COBOL you could read them in and print them out okay but it took you forever to get the answer. So it is obvious to me that PL/1 is needed; everything I have seen says that it is going to be more useful and less expensive to use, than either FORTRAN or COBOL when FORTRAN and COBOL do not precisely meet the needs of the application being programmed. Nevertheless, there is an application-technology cycle which will have to go on. It shouldn't take too long and there are many useful things that will be done with it before that cycle is complete, but the process will still go on.

Job Control Languages

Probably the most significant technological step in the language area in third generation support is the development and requirement on a large scale for something called job control language, or command language. These are programming languages which permit the programmer to program not the machine, but the system of support for that machine. It permits him to call programs into action, manipulate the files of the system and sequence the various functional capabilities of the system. While this was first developed in the operating systems and took the form of control cards, there has been a manyfold proliferation in the capabilities of the systems and in the language capabilities to use them. I would venture to say that this single development is much more important in the long run than all of the conferences on ALGOL, FORTRAN and PL/1. It is important in the sense that no attention is really paid to this area of system design by other than those who have to implement the system.

In general, these languages have just grown, and the problems have been solved by implementers to make life as easy as possible for them. Nevertheless, one has all of the problems of FORTRAN and COBOL, with the added complexity that the functional capabilities are not clearly understood. Furthermore, there is a much more detailed interaction between this language system design and performance that one would think to begin with. It is in this area that one can expect the most trouble and the most development over the next few years. Actual progress will be difficult, however, because changes in these areas require extensive changes in the monitor and supervisory parts of the system, and these are always difficult for manufacturers to carry out.

Data Management

The third big area of advance in third-generation software is in the so-called data management. Data management has gone through an evolution from the days when we all wrote our own I/O to the stage of full data-management capability in which all the problems of cataloging, storing, retrieving and editing all the data needed to run an installation are handled essentially by the system. This is an absolute necessity. It is an absolute necessity for two reasons. First is that the customers need it. They know the cost of putting data inside the system is very high. Once there, it ought always be accessible.

The second reason is that it has become economical for the system to handle these problems by the availability of relatively cheap storage in terms of random access devices at all kinds of price/performance levels.

Key Technologies Not Available

So I see that the major problem in development of manufacturers' third-generation software and users' third-generation software is that they key technologies are not in the available stage. That they aren't is deplorable but hardly surprising since the pushing of these technologies through the state-of-the-art into the available stage only started a year or so ago. When one remembers that the state-of-the-art technology upon which the machines of the third generation are based is only five or six years old, it is not surprising that there is a difference in availability of hardware and software technology.

The fact is that the manufacturers, and the people upon whom the manufacturers relied on, have worked at least 50% harder in trying to rush through this technological development, to try to match the availability of the system to the users. They have succeeded, in my opinion, to an unbelievable extent. The manufacturers put up the money and the programmers put up the blood, sweat, and tears and they have both succeeded magnificently. I certainly didn't think it was that easy to understand three years ago, nor even two years ago. Nor do I think it's wrong to push technology.

A couple of points: Third-generation equipment is tremendously better than second generation. Many people today have been clever in one way or another and whether they understood it or not, they took the approach of watching the technology and are making money out of installations of third-generation equipment. Several ways are available. One way is that known application-technologies were applied to the third generation equipment. Thus the only barrier and hurdle to be overcome were the differences in machine organization and instruction set. In most of the really profitable cases, if it was a tape application it was left tape on the new machine. I know of one client who took simple stand-alone concept and a subset of the COBOL language and we produced a compiler for this client which was running for him six to nine months before anything was available from IBM, although it doesn't provide all the features. IBM itself produced a FORTRAN compiler over a year and a half ago which by being careful not to overstep the application-technology fully exploited the capability of the 360 machine.

The places we're all in trouble is when we pushed both the new hardware which required a new implementation technology and made drastic advances in the application technology. Is this good or bad? It is certainly true that programming technology has been pushed ahead at two to four times the rate it would if that step had not been taken. Thus the whole world will benefit much earlier from these new application technologies than they would have if the effort had not been made. I contend, furthermore, that those who are committed internally to the gigantic steps in the application of the manufacturers' software to the running of your business can bear part of the responsibility for not knowing what you were getting into. But you should be praised because your company will be there first. Whether or not it's worth the pain and sweat, I don't know. I believe that it is. I believe that the decisions to try were valid, the risks were high, the payoffs will be high. It had to come anyway; it had to be faced anyway. In this process unfortunately those people who were trying not to take any risks of their own (or who said they weren't) and who have essentially abdicated their responsibility to understand what they were doing, are in a tough way.

The Future

So I believe the third-generation machinery push is a magnificent human endeavor. The third-generation machinery gives man the power that was inconceivable even five years ago. The ratio of what is available today to what was conceived of twenty years ago is just about infinite. Every one of us in the field knows that he has never faced a more challenging opportunity. But, it is impossible for us to see anything which we are unable to do with computers. Therefore, something of the human inside of us takes over and says we must try to do at least part of what we can see as the possibilities. If one confines himself to the available technologies, then one can write a meaningful discourse on the programming development cost versus pay-off. By definition, they are all able to be estimated before the work starts. We have a well-defined specification of both input and result. We have a scheme for understanding the significant performance parameters of this application; we have measurements of how it exists. It is relatively easy then to redesign it to take account of new things put on new machines, extend it in some ways and be fairly close in our estimates of what it costs to produce and how well it will perform. Given these, one can estimate what the pay-off in machine time, programming time, etc., is and come to a conclusion as to whether it is worth it.

When we are mostly dealing with state-of-the-art in the application, I believe it is almost impossible to tell where you are going to end up, and so the pay-off has to be very large. In any case where the pay-off isn't many times what one could expect to spend, it shouldn't be gone into or it should be gone into gradually with your eyes open.

U. S. BANKS HAVE $837 MILLION WORTH OF COMPUTERS IN USE: MARKET EXPANDING AT 20% PER YEAR

Compare these two statements:

- Fewer than 7% of the nation's banks have computer systems installed; thus market penetration is low.

- Banks have computer systems worth $837 million installed, an impressive 8.7% of the total value of computers installed in the United States; thus banks are a large market factor.

These seemingly contradictory statements are both true. They hold the key to the substantial market opportunity that the computer industry has in the banking field. The figures, and what they mean, come from an analysis of the 158-page results of a recent banking survey conducted by the American Banking Association.

There are about 14,000 banks in the United States, and the ABA calculates that 943 have computers installed and that another 2055 banks use off-premise computer services. This, however, is a significant improvement over the situation in late 1963, when the ABA made its first National Automation Survey. As Table 1 illustrates:

TABLE 1 - GROWTH OF AUTOMATION IN BANKING

	1963 (est.)	1966
Computer services on premises	485	943
Computer services off premises	585	2,055
Plans for automation	1,265	1,358
No plans for automation	11,665	9,638

But looked at in another way, the survey tells a different story. The banks that use computers have roughly 77% of the $303 billion deposited in commercial banks. All banks with deposits of over $500 million are using computers, and for those with deposits of $100 million or more, 91% have their own computer system and another 5% use an off-premise service. Indeed, taken together, these 397 large banks with assets over $100 million have 63% of the nation's total deposits. The breakdown of the outlay for computers is even more striking. As Table 2 indicates, only a a small percentage of the banks with computers spend the major share of what the banking industry pays for its computer systems.

TABLE 2 - ANALYSIS OF COMPUTERS INSTALLED IN BANKS

Monthly Costs	Banks Using Own Computers (est.)		Value of Computers (est.)	
	Number	%	$ millions	%
Under $5,000	146	15%	22	3%
$5,000 - $9,999	406	43	138	16
$10,000 - $14,000	123	13	74	9
$15,000 - $24,999	119	13	113	14
$25,000 - $49,999	81	9	146	17
$50,000 - $99,999	54	6	211	25
$100,000 and over	14	1	133	16
Totals	943	100%	$ 837	100%

Large banks still best customers

Interestingly enough, it is among this small percentage of banks that the computer industry still has its best potential market. The small banks with under $10 million in deposits number 73% of the total in the country, and those with deposits between $10 million and $50 million make up another 22%. Only a handful of the first group have their own computer, and only 11% of the second group do. Another 15% of these banks use off-premise services. Although this group constitutes 95% of the nation's banks, few are ripe for the computer salesman. In the ABA survey, 75% of the banks with less than $25 million in deposits do no computer processing and have no plans to order a computer or use off-premise services. And, when off-premise users were asked when they planned to buy their own computer, some 70% replied that they had no idea when this cut-over time would come. Indeed, only 12% (or 237) thought they would install their own computer by 1968.

Of course, there is only a handful of first computer accounts still available among larger banks. By next year, only 6% or 46 banks with deposits of over $50 million will not be using a computer. But the potential for upgrading and expanding computer equipment at larger banks is good. Almost 93% of computer-using banks with deposits of over $100 million are interested in converting to on-line/real-time processing for some of their applications.

Our estimates indicate an expansion rate of 18% next year in the value of computer equipment installed at current accounts in the banking field, offering approximately $150 million in add-on and replacement computer sales in 1967. New customer business is estimated at $70 million, yielding a total market for $220 million of computer equipment sales to banks during 1967.

Peripheral equipment

Predicting computer vendors' future success among banks is perhaps more difficult than for any market area except the Federal Government. Sales success in this market depends very heavily on peripheral equipment that reads, sorts, and lists checks and other certificates that transfer assets. In particular, an efficient reader/sorter for magnetic ink character recognition (MICR) has been a critical component in computer sales to banks. Now, as we move toward the proposed "checkless society", peripherals that automatically accept and process deposit, transfer, and withdrawal transactions will influence greatly which EDP vendors are successful with sales to banks.

Burroughs, as Table 3 indicates, is strong in the lower-priced equipment fields, while General Electric supplies larger computer configurations. The figures given are total monthly costs, and in many cases include multiple computer systems. As can be seen, IBM supplies almost all the banking customers with large data processing costs. (The 11% "other" figure in the largest category indicates RCA's giant installation at Chase Manhattan Bank.)

TABLE 3 - MONTHLY COMPUTER COSTS BY MANUFACTURER

Monthly Costs	IBM	Burroughs	NCR	GE	Other
Under $5,000	53%	17%	18%	8%	4%
$5,000 - $9,999	58	24	9	6	3
$10,000 - $14,999	64	13	14	9	-
$15,000 - $24,999	64	6	10	16	4
$25,000 - $49,999	66	3	9	19	3
$50,000 - $99,999	63	11	-	18	8
$100,000 and over	89	-	-	-	11

What the banks plan

Bank spending, naturally, tends to be proportional to the size of a bank, as Table 4 shows. The computer, however, is rapidly moving into use by smaller and smaller banks. By 1970, the ABA predicts, all banks with deposits of over $100 million will be using computers, as will 98% of those with $50 million and over in deposits. For even smaller banks, the ABA's survey indicates the computer will come into wider use. It predicts that 49% of all banks will be using a computer by 1970 and 55%, or 7665 banks, by 1971. Most of these, as we pointed out earlier, will be using off-premise computer service from a service bureau, a correspondent bank, or a joint venture.

TABLE 4 - MONTHLY COMPUTER COSTS BY SIZE OF BANK

(Deposits in Millions of Dollars)

Monthly Costs	Under $10	$10 to $49	$50 to $99	$100 to $499	$500 and over	Total
Under $5,000	87%	54%	14%	6%	1%	35%
$5,000 - $9,999	9	38	68	23	1	34
$10,000 - $14,999	4	3	13	24	1	10
$15,000 - $24,999	-	1	3	30	13	9
$25,000 - $49,999	-	3	1	14	33	7
$50,000 - $99,999	-	1	1	3	38	4
$100,000 and over	-	-	-	-	13	1

Banks don't expect to use their current equipment too long. 41% report their present system will be replaced after less than three years of use, and another 35% say the total use of current computers will be between 3 and 5 years. But the next computer banks install (and half have them on order) will see longer use: 38% plan to use them between 3 and 5 years; 56%, over 5 years. The survey results also indicate that the smaller banks will tend to rent computers while the larger ones will purchase them. Only 19% of the smallest size banks plan to use their next computer more than 3 years, while 65% of the largest banks plan to have their next computer more than 5 years. Thus the smaller banks will be good prospects for replacement systems, while larger banks will provide the primary market for add-on memory and peripheral business.

Our projections indicate that the sale of computer equipment to banks should grow at an average compounded rate of 20% during the remaining three years of this decade. Thus, the computer market in banking is growing strongly, and will be for some time. Today's new, small computers make many more banks potential customers, and the development of good application packages for these smaller customers will stimulate this acceptance. The banker, by nature, likes to keep close control on his operation, and a computer system he can afford to install and operate himself will let him move his business away from the service bureau or correspondent bank.

The complete ABA survey results, in tabulated form, are available at $20 per copy from Department of Automation, The American Bankers Association, 90 Park Avenue, New York, N.Y. 10016.

AN INTRODUCTION TO SORT TECHNIQUES

Edward Opengart, Vice President
Programming Methods, Inc.
New York, N.Y. 10006

"Many users spend a great deal of time and money investigating applications only remotely related to their particular needs, yet make no effort to understand sort techniques which tie up from 30 to 70 per cent of their computer time. . . . The approaches to solving this problem are new and dynamic."

In the excitement created by the introduction of new computers and computer languages, sorting has become a systems application people tend to take for granted. Many users spend a great deal of time and money investigating applications only remotely related to their particular needs, yet, make no effort to understand sort techniques which tie up from 30 to 70 per cent of their computer time.

Generalized Sorting Programs

Far from a simple application, the design of a generalized sort program requires a great deal of imagination. Consider the number of data variables that must be considered in sort design such as: record length, blocking, form, control key locations and size, file sequence and size, merge order, etc. In addition, add the variation in hardware considerations that form the environment in which the sort must work, such as: instruction repertoire, machine size, register capabilities, number and/or capacity of I/O units, overlap and multiprocessing capability, and so on. You can now begin to appreciate the task of designing a generalized sort program which can effectively handle all combinations of external data in the environment of the internal capability.

With these variables to consider, it is no wonder that no two generalized sort programs use the exact same procedure in sequencing files. The method best suited to a particular computer configuration must be based on an extensive analysis of the hardware and I/O capabilities. A number of common techniques have been developed in the past, however, and as the external parameters change, new approaches are sure to be introduced. This article will describe some of the most widely used techniques currently being applied.

Development of Logic in Sorting

The development of basic sort logic can be more easily understood by considering a familiar problem. Suppose we were to manually sort 1000 checks in ascending sequence by a five digit check number. What approach would be most efficient? Initially, we might try placing each check in sequence as it is introduced from the input stack. We will find that this procedure will work well for the first twenty or so checks, just as long as we can maintain our capacity to remember the check numbers in our hand. As soon as we lose this capacity, the time required to search for the sequential position of each new entry makes this procedure prohibitive. By the time we reach fifty checks the method becomes completely unwieldy.

We can make use of the most efficient portion of this procedure. By creating sequences of twenty checks each (the maximum number handled efficiently), we can create fifty separate sequenced decks, from the 1000 checks. The problem is now reduced to merging fifty sequenced decks together rather than handling 1000 separate checks.

We might initially think of placing each deck of twenty sorted checks in a separate pile with the lowest serial number in each group on top. By reviewing all fifty piles we can select the lowest of the entire group. As we remove the lowest from the winning pile, we bring the next lowest of that pile to the top and must again compare this new entry against all other piles to determine the new lowest serial number. We can continue this process until we have exhausted all of the input piles. This method is efficient as long as we have the capacity to handle the number of input piles. Suppose, for example, we had 10,000 checks, and thus five hundred piles after our initial grouping and sorting. It would be impossible to use this technique with 500 scpearate sequences.

Capacity of the Sorting Mechanism

The method of merging sorted groups must again be based on the capacity of the sorting mechanism. For our example, we will create five rather than fifty output stacks as optimum and discover later how alternating the number of stacks affects sort efficiency.

The first sequenced group of twenty checks will be placed on pile one, of the second on pile two and so on. Sequence group six will again be placed on pile one, and seven on pile two, etc., until all the checks are exhausted. The result of this distribution will be fifty sequenced decks in five stacks, each stack will contain ten sequences of twenty checks each. Figure 1 represents an example of the output stacks resulting from our initial sequencing.

Application to Computers

The method described for our hypothetical case is the basis for all generalized sorts currently in use. The input records are first grouped and sequenced by an internal sort technique and output or stacked on an I/O media. The number of records (size of initial sequence) that may be sorted internally is a function of available memory and internal sort technique. The number of stacks created is a function of available units. This internal sort is generally referred to as Phase 1 of the sort and the size of the initial sequence as G.

```
Pile 1             Pile 2             Pile 3             Pile 4             Pile 5

00001              00007              00018              00019              00012
00008              00080              00025              00036              00014
00015   Seq.1      00086   Seq.2      00037   Seq.3      00079   Seq.4      00093   Seq.5
  .                  .                  .                  .                  .
  .                  .                  .                  .                  .
  .                  .                  .                  .                  .
00134              00212              00193              00210              000181

00013              00002              00011              00081              00003
00026              00019                .                  .                  .
00031              00034                .                  .                  .
  .     Seq.6        .     Seq.7        .     Seq.8        .     Seq.9        .     Seq.10
  .                  .                  .                  .                  .
  .                  .                  .                  .                  .
00297              00300              00235              00280              00179
  .                  .                  .                  .                  .
  .                  .                  .                  .                  .
  .                  .                  .                  .                  .
```

Figure 1 - Sequenced Piles of 20 checks

By continually extracting the lowest check number of the five stacks, we can combine sequences 1-5 into one sequence. For example, we would extract check numbers 00001, 00007, 00008, 00012, 00018 and so on until all of the checks in sequences 1-5 are merged. The five sequences of 20 checks each would thus become one sequence of 100 checks. By repeating the same procedure with sequences 6-10, 11-15, and so on we create as output 10 sequences of 100 checks. If we again place these ten sequences on five separate stacks, as before, each stack would now contain two sequences.

The merge procedure could now be repeated with the five stacks just created. The first five sequence of 100 checks each can be merged into one deck to create a sequence of 500 checks. The second five, a second deck of 500 sequenced checks. The result of this second "pass" would be two stacks each containing one sequence of 500 checks. By repeating the "merge" procedure once more, the two sequences can be combined into one stack of 1000 checks.

The Order of Merge

If we review the "merge" procedure we will see that each pass increased the number of checks in sequence by five (20, 100, 500). In addition, the total number of sequences was reduced by five (50, 10, 2). If we would have used ten stacks rather than five the merge would have increased the sequence size and reduced the number of sequences by a factor of ten each pass. This factor is known as the order of the merge, and, excluding all other factors, the larger the order of merge, the faster the sort. Figure 2 shows the number of passes required to merge a given number of sequences based on the order of the merge.

The merging of internally sequenced records (Phase 2) is accomplished by successive passes of the files. The number of records in sequence (string) is increased by a factor of the order of the merge each pass, and the total number of strings reduced by the same factor until only one string containing all the records in the file remains. The number of passes, therefore, is a factor of the number of initial strings and the order of the merge. With this basic understanding we can now review some techniques used to both internally sort input records as well as procedures for merging sequenced strings to form a sorted file.

Internal Sort Techniques

A variety of techniques of grouping and internally sequencing input records have been found to be effective for particular computer configurations. The most common of these, the insertion, the bubble sift, the sift-merge and the tournament, will be described in this article. The function of each is to create as large a sequence (G) as possible thus reducing the number of merge passes required to order the file.

For ease of explanation, we will assume an internal sort capability of only eight records. For most computers and record sizes, the internal sort blocking is many times greater. In our examples, we will show only the control portion of the record, the remainder of the record, though present, is not essential for our explanation. In practice, most internal sorts manipulate only keys and do not resequence records until they are output.

Insertion Technique

The insertion technique exactly parallels the manual technique we originally described in sequencing checks. Each record is "inserted" into its proper sequential position as it is introduced. If you recall, this system was effective as long as we didn't try to go beyond our capacity. At that point the added time required to locate each new entry made the procedure undesirable.

Passes Required: Order of Merge	Number of Sequences 2	4	5	10
1 Pass	2	4	5	10
2 Passes	4	16	25	100
3 Passes	8	64	125	1,000
4 Passes	16	256	625	10,000
5 Passes	32	1024	3125	100,000

Figure 2

The computer has the advantage of being able to locate the insertion position much more quickly (and quite fortunately, much more accurately) than we can acomplish manually. Several methods of insertion exist, the two common procedures being straight and binary techniques.

The straight insertion technique compares each new record with the lowest of the records already in sequence. If the new is lower, it is "inserted" in front as the new winner. If the new is higher, it is compared against the next lowest, and so on until the proper "insertion" position is found. On the average, one half of the sequenced records, G/2, will have to be searched to locate the appropriate insertion position. Since the last position is automatically high and the need not be checked, the average number of compares is (G-1)/2.

Binary Search Technique

The binary search technique uses a faster, more sophisticated method for locating the insertion position. Instead of comparing each record sequentially, a binary search is made of the sequenced records to find the insertion position. This search continually halves the sequenced records to locate the insertion position. The disadvantage is that more coding is required, thus reducing the maximum size of G. For ease of understanding this technique consider inserting a record whose key value is 29 into a partially sequenced string of:

01 05 16 37 25 19 80 99

The binary search begins by comparing the new value with the middle value of the sequenced records. Immediately, half of the records are eliminated since the new value must either be higher than the mid value (and thus higher than the first half of records) or lower than the mid value (and thus lower than the second half of the records). In our example, the new value (29) is higher than the mid value (19) and only the second half of records need be considered for the proper insertion position.

The remaining records are again halved and the new midpoint (37) compared with the new entry. In our second compare, the second one-quarter is eliminated since the new (29) is lower than the mid (37). The record to be inserted is now defined as greater than 19 and less than 37. The last compare (25) locates the exact position.

In the example it took three compares to locate the insertion position in a group of eight records. Four compares would have been sufficient to locate the position in a group of sixteen records since the first compare would eliminate eight as either high or low. In general, the number of compares (N) required to locate the insertion position in a group of G records is $N = \log_2 G$ or $2^N = G$. For a G of 32 records, the number of compares required would be five. For the same G, an average 15.5 compares would be required using straight insertion.

Bubble Sift

The bubble sift technique acquired its unique name from the method it employs in internally sequencing records. Given G records, this technique bubbles the lowest record to the top of the pack, then the next lowest, the third lowest, and so on. The major advantage of this technique is that the lowest record of the group is available at once and may be output while the next lowest is being determined (simultaneity). In comparison, the insertion technique required that all records be considered before any record could be output since the last record in may be the lowest of the group.

Consider the same records as the preceding example input as:

25 05 16 99 29 37 01 19

The lowest of the group is determined by comparing the first record with each subsequent record in sequence. If the first is lower, it maintains its position as the initial record. If the first is higher, the two records are transposed with the lower record becoming the first record.

As example, 25 is compared to 05. Since 05 is lower the two records are switched and the sequence becomes:

05 25 16 99 29 37 01 19

Now 05 (the new first entry) is compared with the third record (16). No change results since 05 is lower than 16, as well as the next entries 99, 29 and 37. When 05 is compared with 01, the two are interchanged since the first is no longer lower, the sequence becomes:

01 25 16 99 29 37 05 19

The new first (01) is compared against the last record (19) and maintains its position as winner. The cycle is again repreated (this time starting with record two (25) against record three (16)) to determine the lowest of the remaining records. While this search is going on the winner can be output since it has already been proven the lowest.

Sift Merge

It is often possible to combine more than one technique to create an optimum internal sort method for a particular computer configuration. One such combination is the sift merge procedure. The group of records to be internally sorted are divided into two (and possibly more) groups. Each group is sequenced using either a sift or insertion technique and the sequenced group merged to form the output string.

Consider the group of G records as before, divided into two groups of four records each. The input would be:

Group 1: 22 05 16 99
Group 2: 29 37 01 19

By either method of internal sort previously described, we could sort each G/2 records to create two sequences:

Group 1: 05 16 22 99
Group 2: 01 19 29 37

The two groups could then be merged to create the sorted string, producing phase 2 of the sort:

01 05 16 19 22 29 37 99

A number of advantages may be realized by using this procedure. If, for example, the binary insertion technique were used to sequence each G/2 records, the lowest record could be made available before the sequencing is complete. The total number of compares (assuming a two-way merge) would be the same as the insertion method for G records.

Tournament Technique

The final internal sort technique to be discussed is one that is familiar to all sport fans. The basic approach is to play one record against the other, with the lowest being the winner. This winner is compared against the winner of a different contest with the lowest of these two representing all four records as the winner in the next contest level, and so on.

Consider again our eight records input as before. Each two records are considered a contest and the lowest, or winner, placed at the next level to play against another winner. The winner of this level is moved as challenger to the next level, and so on, until the tournament winner is chosen (see Figure 3).

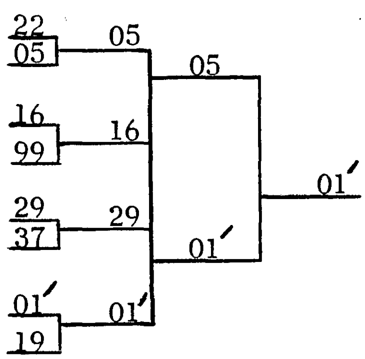


Figure 3

An advantage of this technique is that it is much faster to select subsequent winners once the first winner is chosen. There is no need to repeat all comparisons since only the branch containing the winning entry need be considered. For example, to choose the second lowest, we need only return to the branch from which 01 was chosen. Since 01 was selected it is eliminated from the contest and the new tournament is as shown in Figure 4.

Those contests containing XX need not be repeated since no change is possible as a result of selecting 01 as initial winner.

Replacement

A considerable advantage can be gained by replacing each winning entry with a new record or contestant. If the new record is higher than the record that just won, the new record may be considered as part of the same sequence and can be treated as an entry in this tournament. If the new record is lower than the winning entry, it may not be entered in this tournament since it will create a break in sequence if it is allowed to win. A simple solution is to provide a sequence number as major control to each new entry. If the new entry is higher than the previous winner, it is assigned the same sequence value as the winner. If lower, it will be assigned the next higher sequence value.

As an example, assume our initial sequence value to be A and the records that follow our initial A record to be 09, 02, 23. . . When 01 was declared the winner, its position in the tournament was replaced by the next entry, or 09. Since it is higher than the winner, it will be assigned the same sequence value and the second round of the tournament becomes as shown in Figure 5.

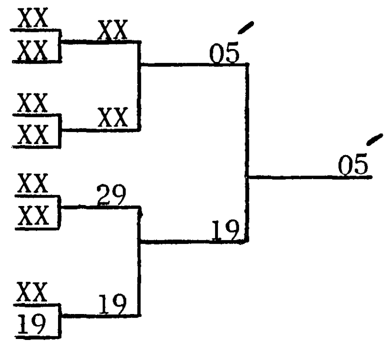


Figure 4

The primed A, A′, or primed number, such as 05′, represents the sequence value.

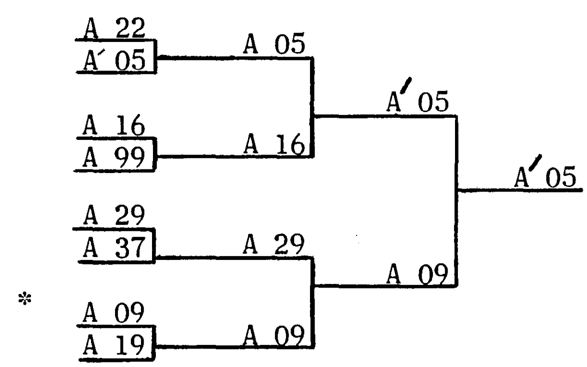
The new winner of the tournament, 05, is replaced by the next available record, which is 02. This entry is lower than the previous winner (05) and must be assigned the next higher sequence value B to avoid a break in sequence. The new tournament becomes as shown in Figure 6.

This procedure accomplished two objectives. It assured that all records still in sequence were output before the new low entry; and it allowed automatic initialization of the next sequence since the first time a record with a sequence value of *B* becomes the winner, the tournament must contain all records with a *B* sequence value.

The replacement procedure does a great deal to improve sort efficiency. In all non-replacement techniques, the greatest number of records that may be sequenced internally was limited by the capacity of memory. The size of G was the maximum number of records that could be held in core at any one time. By *replacing* the winning entries with new records, it is possible to internally sort more than the memory capacity. In fact, on the average, approximately 2 times G records may be sequenced using a replacement technique.

Merge Techniques

Once the input records are grouped and internally sorted to form strings of G records, the next step is to combine or merge these strings by successive passes to form a sorted file. Three methods are currently in use to accomplish this merge phase: the balanced merge, the cascade merge and the polyphase merge techniques. The selection of the optimum method is dependent on hardware and I/O capabilities.



*New Record

Figure 5

Each of these three approaches will be discussed separately, and as before, no justification or evaluation will be made. For simplification, we will again use an example. This time, we will assume approximately thirty-one strings coming from phase 1. The number of records in each string is not important since the merge is only interested in reducing the number of strings to one by repetitive merge passes.

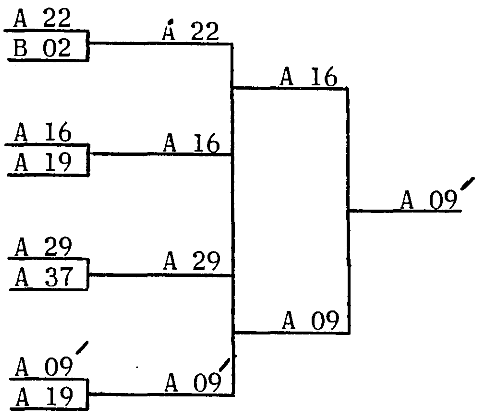


Figure 6

Balanced Merge

Because of the simplicity of design and implementation, the most common technique currently applied is the balanced merge.

Assuming N tapes, the balance technique distributes the phase 1 output of internally sorted strings on N/2 of the I/O units. These strings become input to the first merge pass which increases the length of each string by the order of the merge (N/2) and reduces the number of strings by the same factor. The output becomes input to the second pass of the merge and what was the input to pass-one becomes available as output. The records are thus ping-ponged back and forth until the merge creates a single output string.

Consider a simple example of a four tape merge of 32 sequences. For simplicity, assume that each string (G) contains only one record. The number of records in a string will be shown as N/xx where N represents the number of strings and xx the number of records in each string. See Figure 7.

	Tape 1	Tape 2	Tape 3	Tape 4	Total Records Processed
Pass 0	16	16	0	0	Initial distribution
Pass 1	0	0	8/2	8/2	32
Pass 2	4/4	4/4	0	0	32
Pass 3	0	0	2/8	2/8	32
Pass 4	1/16	1/16	0	0	32
Pass 5	0	0	1/32	0	32
					160

Tapes 1 and 2 are input on even number passes and output on odd passes. Tapes 3 and 4 are the opposite.

Figure 7

Cascade Merge

As we described previously, the higher the order of merge, the smaller the number of passes required to create a sequenced file. Assuming an N tape configuration, there are methods by which a merge order greater than N/2 may be realized. The first of these is the cascade merge.

The basic logic of this procedure is to distribute the output strings of phase I on N-1 of the tapes, leaving only one tape for output. The strings are distributed according to a predetermined algorithm such that no tapes have the same number of sequences. No attempt will be made to explain this algorithm except to mention that the distribution is determined by a linear combinaton of numbers in a Fibonacci sequence.

The N-1 tapes are then merged onto the remaining tape until the input tape with the shortest number of sequences reaches end of file. This emptied tape becomes an output file and the remaining tapes (N-2) are merged onto it until the next tape empties. An N-3 merge continues onto this new empty tape again until an end of file is reached. In general, the merge pass continues until all input tapes have reached their logical end. At this time the distribution of records is such that the next pass starting with an N-1 merge can be accomplished. An example is again in order to better describe the technique.

We will again assume a four tape system merging 31 strings from phase 1 where, for simplicity, each string contains only one record. An asterisk (*) denotes an input to the merge. See Figure 8.

From this example you will note that there were only 4 passes rather than five required to merge the phase 1 sequences. The total number of records passed were 124 as opposed to 160 required by the balanced merge procedure.

Pass	Tape 1	Tape 2	Tape 3	Tape 4	Order	Processed
0	11/1*	0	6/1*	14/1*	Initial distribution	
1	5/1*	6/3	0	8/1*	3 way	31
	0	6/3	5/2	3/1*	2 way	
	3/1*	6/3*	5/2*	0	1 way	
2	0	3/3*	2/2*	3/6	3 way	31
	2/5	1/3*	0	3/6	2 way	
	2/5*	0	1/3*	3/6*	1 way	
3	1/5*	1/14	0	2/6*	3 way	31
	0	1/14	1/11	1/6*	2 way	
	1/6*	1/14*	1/11*		1 way	
				1/31	3 way	31
						124

Figure 8

Thirty-one sequences were chosen since it easily fit into the algorithm for string distribution. In general, the cascade is far more complex to implement, and since input and output are interchanged for each merge. this technique requires some form of cross channel switching to be effective.

Polyphase Merge

Another approach to obtaining maximum merge from an N tape system is the Polyphase technique. Unlike the cascade, the polyphase always maintains an N-1 merge to create the sorted file. This distribution of strings from phase 1 is accomplished much the same as the cascade.

An algorithm based on the Fibonacci sequence is used to allocate the sorted strings among the N-1 tapes. These tapes are then merged always using an N-1 merge until the shortest tape is ended. This tape together with the remaining input tapes are merged onto the tape that just emptied until the next file is exhausted. The emptied tape becomes the output for the other N-1 input reels.

Returning to our example of 31 strings, each only 1 record long, a schematic of the polyphase would be as shown in Figure 9. The example illustrates the great saving in record handling. Since not all records are processed in each pass, the total records handled in the five polyphase passes is far less than the records processed in four cascade passes. Since we have assumed a sequence size from phase 1 of only 1 record (in reality it may be 100), and we have specified only 31 sequences, the magnitude of savings must be multiplied many times to relate to an actual sort.

Pass	Tape 1	Tape 2	Tape 3	Tape 4	Records Processed
	7/1	11/1	13/1	0	Initial distribution
1	0	4/1	6/1	7/3	21
2	4/5	0	2/1	3/3	20
3	2/5	2/9	0	1/3	18
4	1/5	1/9	1/17	0	17
5	0	0	0	1/31	31
					107

Figure 9

Several interesting advances have been developed in both hardware and software to make this approach more desirable. The cross-channel switching feature provides for dynamically assigning N-1 units to the input channel and the remaining unit to output. The read-backwards facility of current hardware allows for sorting without rewind. This is accomplished by alternating the sequence of the strings on the phase I output files. The first series of strings is in ascending order, the second descending, the next ascending and so forth. The merge (reading all tapes backwards) is accomplished by alternating the merge from descending to ascending for each new sequence. In this way, both input and output tapes are always properly positioned for merging with no rewind required.

The Future

The internal sort and merge techniques outlined in the previous portion of this article represent the more common procedures of current generalized sorts. As the capability of hardware improves, and the state of the art of software techniques advances, new and faster approaches are sure to be introduced.

Externally, the problem of sequencing files is an old and static problem. Internally, the approaches to solving this problem are new and dynamic.

CALENDAR OF COMING EVENTS

Feb. 16-17, 1967: The Association of Data Processing Service Organizations (ADAPSO), Sheraton Hotel, Chicago, Ill.; contact James Powell, United Data Processing, Inc., 1001 S.W. Tenth Ave., Portland, Oregon

Mar., 1967: Fifth Annual Symposium on Biomathematics and Computer Science in the Life Sciences, Shamrock Hilton Hotel, Houston, Texas; contact Office of the Dean, Division of Continuing Education, the University of Texas Graduate School of Biomedical Sciences, 102 Jesse Jones Library Bldg., Texas Medical Center, Houston, Texas 77025

March 1-3, 1967: Numerical Control Society, Annual Meeting and Technical Conference, Statler Hilton Hotel, Detroit, Mich.; contact Thomas C. Huber, DeVlieg Machine Co., Royal Oak, Mich.

Mar. 7-9, 1967: 8th Annual AFETR Range User Data Symposium, Air Force Eastern Test Range, Orlando Air Force Base, Fla.; attendance by invitation only; a SECRET clearance is required; contact Col. Asa P. Whitmire, Chief of Data Processing Div., Patrick Air Force Base, Fla. 32925

April 4-7, 1967: Honeywell H800 Users Association (HUG) Spring Conference, Bellevue Stratford Hotel, Philadelphia, Pa.; contact R. E. Hanington, Philadelphia Electric Co., 2301 Market St., Philadelphia, Pa. 19103

April 4-7, 1967: Joint Conference of the Univac Users Association and the Univac Scientific Exchange, Fontainebleu Hotel, Miami, Fla. Contacts: UUA — Murray Hepple, Harris Trust, 111 W. Monroe St., Chicago, Illinois; or USE — S. C. Bloom, Univac, P.O. Box 8100, Philadelphia, Pa. 19101

April 6-7, 1967: Atlantic Systems Conference, Americana Hotel, New York, N.Y.; contact Dr. Gibbs Myers, The General Precision Co., Wayne, N.J.

April 7, 1967: Association for Computing Machinery, San Francisco Bay Area Chapter, Jack Tar Hotel, San Francisco, Calif.; contact A. E. Corduan, Lockheed Missile & Space Co., P.O. Box 504, Sunnyvale, Calif. 94088

April 12-14, 1967: Electronic Information Handling Conference II, Flying Carpet Motel, Pittsburgh, Pa.; contact Allen Kent or Orrin E. Taulbee, Co-Chairmen, Conference on Electronic Information Handling, University of Pittsburgh, Pittsburgh, 15213

April 18-19, 1967: ECHO (Electronic Computing Hospital Oriented) Annual Meeting, American Hospital Association Headquarters, 840 N. Lake Shore, Chicago, Ill.; contact Howard Abrahamson, Director of Data Processing, Fairview Hospitals, 2312 South Sixth St., Minneapolis, Minn. 55409

April 18-20, 1967: Spring Joint Computer Conference, Chalfonte-Haddon Hall, Atlantic City, N.J.; contact AFIPS Hdqs., 211 East 43 St., New York, N.Y. 10017

April 20-22, 1967: Oregon Association for Educational Data Systems, Spring Conference, Portland State College, Portland, Oregon; contact Phil Morgan, Room 015, College Center, P.O. Box 751, Portland, Oregon 97207

May 3-4, 1967: Annual National Colloquium on Information Retrieval, Philadelphia, Pa.; contact R. M. Hildreth, Publicity Chairman, Auerbach Corp., 121 N. Broad St., Philadelphia, Pa. 19107

May 18-19, 1967: 10th Midwest Symposium on Circuit Theory, Purdue University, Lafayette, Ind.

(Please turn to page 55)

FUNDAMENTAL CONCEPTS OF PROGRAMMING LANGUAGES

Jean E. Sammet
IBM
Federal Systems Division
Cambridge, Mass.

"The widespread use of programming languages has given tangible proof that the advantages far outweigh the disadvantages in most cases."

In order to use a digital computer it is necessary to write a program to accomplish the desired task. Thus, a computer without a program is about as useful as an automobile without an engine. If programming is necessary to make a computer "go," then the language in which the program is written is the key to effective use of the computer. Hence, it seems fair to say that the communication between person and computer is fundamental — both as a problem and as a technique. Since the advent of the computer, the endeavors of many people have been — and continue to be — devoted to trying to ease this communication problem. Probably the single most important tool is what has become known as a "programming language," sometimes called a "higher level language" (to distinguish it from the normal machine codes or assembly languages which are more closely related to the hardware).

Unfortunately, there does not appear to be any universally accepted definition of a programming language, and it is therefore easier to provide characteristics rather than a specific definition. It is of course taken for granted that a programming language is some set of characters and rules for combining them by which the user can communicate with the computer to cause useful work to be done; this communication takes place through another program which is normally called a compiler, and whose purpose is to translate the user's program (called the source program) into machine code (called the object program) which can then be executed by the computer. This contrasts with the use of a language which is the same as, or very similar to, the direct language used by the computer, namely an assembly language.

This article seeks to characterize programming languages, point out their advantages and disadvantages, provide some classifications of programming languages with proposed definitions, and discuss the major issues in programming languages. The latter are subdivided into nontechnical and technical characteristics of programming languages. Although much of the material in this paper may be familiar or even well known, it is hoped that the various classifications and ways of grouping concepts will prove useful.

This is the first of two articles based on a talk delivered to an American Management Association meeting on October 3, 1966. The material in both articles is condensed from chapters in a book on programming languages which the author is writing.

Characteristics that Define Programming Languages

A programming language is a set of characters and rules for combining them which has the following four characteristics:

It requires no knowledge of machine code by the user. In other words, the user need only learn the particular programming language and can use this quite independently of his (perhaps non-existent) knowledge of any particular machine code. This does not mean that the user can completely ignore the actual computer. For example, he may wish to take advantage of certain machine facilities which are known to him and which provide more efficient programs, and in particular he obviously cannot use input/output equipment which does not exist on a particular computer configuration. However, the fundamental point is that he does not need to know the basic machine code for the given computer.

A programming language must have some significant amount of machine independence. This means that there must be some reasonable potential of having a source program run on two computers with different machine codes without completely rewriting the source program.

When a source program is translated to machine language, there is normally more than one machine instruction per executable unit created. For example, an executable unit in a programming language might be something of the form "A = B + C*D" or "MOVE A TO B." Normally, each of these executable units would be translated into more than one machine instruction.

A programming language normally employs a notation which is somewhat closer to the specific problem being solved than is normal machine code. Thus, for example, the example "A = B + C*D" might be translated into a sequence of instructions such as:

```
CLA   C
MPY   D
ADD   B
STO   A
```

which is clearly less understandable than the programming language form.

Advantages of Programming Languages

As always, one cannot obtain something for nothing, and therefore there are both advantages and disadvantages to programming languages, where the alternative is some type of assembly language. Let us consider the advantages first.

The primary advantage of a programming language is that it is easier to learn than a machine language. It must be emphasized that there is a relative aspect involved in this advantage. An extremely powerful programming language might be harder to learn than an assembly language on a computer with only a dozen instructions. However, given programming and assembly languages of approximately the same complexity in their relative classes, the programming language will be easier to learn. This actually has two facets to it. The programming language may itself be extremely complex, but its ease of learning often comes because the notation is somewhat more related to the problem usage than is the machine code; furthermore, more attention can be paid to the language itself rather than to the idiosyncrasies of the physical hardware which are necessary when one deals in machine code.

A problem written in a programming language is generally easier to debug for two major reasons. First, there is actually less material which needs to be written, because of the explosion factor indicated as the third characteristic of a programming langage. Thus, in comparison with a program written in assembly language, the source program will be physically shorter. Since the number of errors is roughly proportional to the length of the program, obviously there will be fewer errors. A second reason for the program being easier to debug is that the notation itself is somewhat more natural and therefore more attention can be paid to the logic of the program with less attention paid to details of the machine code.

A program coded in a programming language is generally easier to understand and to transfer to someone other than the originator because of the notational advantages and relative conciseness already mentioned.

Fourth, the notation of a programming language automatically provides certain documentation because the notation is easier to understand and the logic is easier to follow.

Finally, the above advantages tend to accumulate into one general advantage which is that the total calendar time required for the problem solution is generally reduced significantly.

Disadvantages of Programming Languages

The primary disadvantage is that the advantages do not always exist in specific cases, and the person might be *worse* off than with a very simple assembly language, as was indicated under the first advantage. Thus, the programming language might be extremely difficult to learn, and unless proper attention is paid to the compiler and other facets of the overall system, the other advantages may not themselves accrue. Fortunately, this seldom occurs.

Second, the additional process of compilation obviously requires machine time, and this may require more than the machine time saved from easier debugging.

Third, the compiler might produce very inefficient object code. This would significantly affect production runs, i.e. programs which are run repeatedly and whose machine time requirements are increased significantly by any inefficiencies. (The counter argument to this of course is the fact that compilers nowadays generally produce code that is at least as good as the average programmer can produce and there are only a few really expert programmers who can write the most efficient machine code.)

Finally, the program may be much harder to debug than an assembly language program if the user does not know machine code and the compiler does not provide the proper type of diagnostics and debugging tools. A user who must look at a memory dump in octal, which he does not understand, is going to have more trouble than debugging an assembly language program in which he understands what is happening.

Classifications of Programming Languages and Proposed Definitions

As indicated earlier, it is very difficult to define a programming language. However, it is a bit easier to propose definitions for classes of programming languages. The terms to be defined are the following: procedure oriented, non-procedural, problem oriented, application oriented, special purpose, problem defining, problem describing, problem solving. Note that some of these are overlapping and that a particular language may fall into more than one of these categories.

A *procedure oriented* language is one in which the user specifies a set of executable operations which are to be performed in sequence; the key factor here is that these are definitely executable operations, and the sequencing is already specified by the user.

The term *non-procedural* has been bandied about for years without any attempt to define it. It is my firm contention that a definition is not really possible, because non-procedural is really a relative term in which decreasing numbers of specific steps need be provided by the user as the state of the art improves. Thus, before such languages as FORTRAN existed, the statement $Y = A + B*C - F/G$ could be considered non-procedural, because it could not be written as one executable unit and translated by any system. Right now, the sentences "calculate the square root of the prime numbers from 7 to 91 and print in three columns" and "print all the salary checks" are non-procedural because there is no compiler available that can accept these statements and translate them; the user must supply the specific steps required. As compilers are developed to cope with increasingly complex sentences, then the nature of the term changes. Thus, what is considered non-procedural today maybe definitely procedural tomorrow. One of the best examples of a currently available non-procedural language is a report generator in which the individual supplies essentially the input and the output without any specific indication as to the procedures needed.

The term *problem oriented* has been used in many ways by different people, but it seems that the most effective use of this term is to encompass any language which is easier for solving a particular problem than machine code would be. Any current programming language illustrates this, and thus the term "problem oriented" is a kind of catchall.

The term *application oriented* seems to apply best to a language which has facilities and/or notation which are useful primarily for a single application area. The best illustrations of this are such languages as APT for machine tool control, and COGO for civil engineering applications. Notice that both of these are of course problem-oriented languages. On the other hand, FORTRAN and COBOL are problem oriented but much less application oriented than APT or COGO. Here again, the term is somewhat relative because

Guns, courtesy of Abercrombie & Fitch. For quality reproductions of this photograph, write us at Memorex.

THE TWO GREAT TAPES, I

W
g
T
W
c
r
I
p
T
p

LIKE THE TWO GREAT SHOTGUNS, ARE NOT IDENTICAL.

When you buy a Purdey or a Holland, you buy the finest in guns. They cost about forty-five hundred dollars each. They last for generations.

When you buy the two great tapes—ours or the one the computer company makes—you buy the most durable, most reliable computer tape around. Naturally, they cost more. It's not surprising, therefore, that they make pass after pass after pass without a drop-out.

The two great guns are pretty much alike in total performance. But although each is exquisitely crafted to perform unerringly, they are not identical. You'd have to use them both to know exactly what each can do.

It's the same with the two great tapes: they aren't exactly alike, either. In fact, some tape-users insist that our great tape out-performs the other.

Could be.

(Brochures? Specs? Write us at 213 Memorex Park, Santa Clara, California 95050.)

MEMOREX

FORTRAN is suitable for applications involving numerical mathematics, whereas COBOL is obviously suited for business data processing and the overlap between these is relatively small. The wider the application area, the more general the language must be.

A *special purpose* language is one which is designed to satisfy a single objective. The objective might involve the application area, or the ease of use for a particular application, or pertain to efficiency of the compiler or the object code.

A *problem defining* language is one which literally defines the problem, and may particularly define the desired input and output, but does *not* define the method of transformation. Here, the best illustration is a report generator. A somewhat secondary example is sorting routines, except that the input to sort routines usually is not in the form of a language per se.

A much more general type of language classification is that referred to as *problem describing* in which the objective is described only in very general terms, e.g. "calculate payroll", All such a statement does is describe, in the most general way, the problem which is to be solved but gives no indication whatsoever as to how to solve it. We are an extremely long way from this type of language.

Finally, a *problem solving* language is one which specifies a complete solution to a problem. Like the term non-procedural, this is a relative term which changes as the state of the art changes.

Non-Technical, i.e. Functional, Characteristics

Most articles and talks about programming languages tend to throw in a great many concepts and characteristics without any very careful delineation of their relationships to one another. Thus when people try to decide which language to select for a given task, they often run into difficulty because of a lack of clearcut factors against which to make their evaluation. In particular, insufficient attention is usually paid to the difference between non-technical and specific technical, characteristics. While it is *not* the purpose of this paper to tell readers how to select a particular programming language, it is hoped that some of the fundamental background material contained herein might be useful in such a situation.

Purpose of a Programming Language

Three different types of purposes must be determined in defining any programming language. The first and most important is the particular application area for which the language is designed. Thus, it must be determined whether this is to handle such things as numerical scientific work, or graphic displays, or business data processing, or engineering design, etc.

The second purpose is to specify the type of language, where in this context the following things are meant. First: into which one or more of the classifications previously cited does the language fall? Secondly: what is the form of the language relative to succinct and/or formal notation, versus naturalness? Third: is the language meant for direct input to a computer or primarily as a publication or reference language? Finally: what type of user the language is intended to assist? Generally speaking, programming languages tend to aid the application programmer who is not necessarily a full-time professional programmer, or in other words, most programming languages tend to help the individual who has a specific problem to solve rather than somebody whose major interest is in computer programming.

Conversion and Compatibility

There has probably been more talk and discussion about conversion and compatibility than almost any other aspect of programming languages. It would be impossible to discuss these topics in any great detail here; so the main thing is to point out some of the varying types of compatibility that exist. The first is compatibility across individual machines, i.e. how machine-independent is a particular language? The second type of compatibility relates to the compiler; unfortunately, the same language is not always implemented the same way even for the same machine; there exist cases in which compilers for the same language, on the same machine, do not produce the same results. A third type of compatibility is related to the problem of dialects, where major and minor changes and/or subsets and extensions are all made in the name of the same language.

With regard to ease of conversion, the first and easiest way of converting is based on compatibility which hopefully exists. Two other ways are direct translation into another language, and semi-translation whereby the user does some of the work.

Standardization

The purposes of standardization are fairly well known. Basically, standardization of programming languages eases conversion problems, permits compatibility across machines, eases the training problem, and generally reduces economic costs from many points of view. The problems in standardization are a little less obvious. They actually fall into three classes — conceptual, technical, and procedural.

In the conceptual category falls the major difficulty of determining when it is time to establish a standard, and how one establishes a standard and at the same time permits new developments to proceed. This type of conceptual problem is not unique to programming languages, and exists in many fields.

The technical problems in standardizing programming languages are enormous. For example, in the more than five years of existence of Committee X3.4 (which is the subcommittee on standardizing programming languages), only one language has been standardized and that only recently — namely FORTRAN; this was by far the best known and most widely available language. In essence, it is very difficult to write down the definition of a language in unambiguous terms.

The third problem in standardization is the method of establishing standards and the procedures associated therewith. These are long and laborious, and have the advantage of requiring a consensus, so that when a standard actually does come into existence, it is one which will really be obeyed, even though compliance with USASI (formerly ASA) standards is completely voluntary.

Types and Methods of Language Definition

When one thinks of language definition, one almost invariably thinks of various technical ways in which languages can be defined. However, although it is often overlooked, the administrative framework within which a language is defined tends to have a significant effect on the language. For example, the question of who designed the language and in what organizational framework, is of paramount importance. Unfortunately, language design is still an art and not a science, and different people often have very strong personal views which can't be defended on any other grounds than "I (don't) like it". The organizational framework is also of major importance, because a language which is designed by an intercompany group will have many more compromises than that designed by a single organizational unit.

However, I wish to state very strongly that those who oppose language design by committee are putting their heads in the sand, because I do not know of any major fully implemented language which was actually designed by a single person. The moment that several people become involved, there is in fact a committee, even though it may be a single organizational unit with a person designated as being in charge. Thus, the mere fact that a language is designed by an intercompany committee is not bad in and of itself. The more significant problem in a situation like that is that the people usually have other job assignments and thus their time is limited. Furthermore, the interconnection between the language designers, the implementers, and those who maintain the language (i.e., settle tricky points of interpretation and perhaps try to extend it) is a crucial part of the administrative framework. If the people who implement the language are not the ones who designed it, or conversely, there is always trouble. Similarly, if the people who maintain the language are not those who designed it then there is difficulty arising from the fact that the newcomers always have (or think they have) better ways of doing things. Sometimes the language is redesigned each time a new group of people becomes involved, and chaos results. The interconnection between the language maintainers and those who do the implementation plays a key role in the orderly development — or lack thereof — of a language.

A number of technical problems are associated with language definition. First and foremost is that we do not yet know how to give a completely formal definition of a programming language. The three components of a language definition are: (1) syntax, which specifies the legal strings of characters (e.g., A + B, not + AB); (2) semantics, which specifies the meaning (e.g. add, not subtract); and (3) pragmatics, which specifies the interaction between the user and the system. We have fairly good ways of defining the syntax, but ways of defining the semantics are only now being developed, and we have made no progress on the pragmatics except to recognize that it is a problem.

Evaluation Based on Use

Very often languages are evaluated on the basis of manuals, speculation, and offhand opinion, rather than on the basis of use. It may be logical for somebody to evaluate a language manual in this way — but he should not evaluate the language itself until it has undergone some usage. Part of usage is training, learning, writing programs, debugging them, etc., and each is fair game for objective evaluation. However, one must be very careful to distinguish between evaluating the language and evaluating the compilers which translate it. The language may be good and the compilers very bad; in cases like this, it is almost always the language which is blamed, although it simply may be a matter of ineffective compiler writing.

* Technical Characteristics

Technical characteristics of a language, mean the sum total of what actually constitutes the language. The main technical characteristics of a language are the form of the language, the structure of the program, the data types and units, the executable statement types, the declarations and nonexecutable statements, and the structure of the language and the compiler interaction.

Form of Language

The form of the language involves many facets. The first is the actual character set which is used, and the second is the ways in which the characters are combined to form names (e.g., TEMP), operators (PLUS, AND), executable commands (COMPUTE, GO TO), and non-executable declarations (DIMENSION, INTEGER). The third facet of language form is the specific rules for forming names and words. The rules for names involve such things as the formation of data names and statement labels, the handling of subscripts and qualification, the use of reserved words, and the definition of literals. The methods of combining words involve the significance of blanks, the use and meaning of punctuation, the presence or absence of noise words, and ways of combining operands and operators. Finally, the form of language obviously involves the physical input which is used, as well as the conceptual form (e.g., fixed format versus a string of characters, English-like versus highly symbolic, etc.). Although a language is really independent of the input medium, when punch cards are meant to be the prime input source, the language tends to be defined as a partially fixed format. Similarly, if paper tape or direct typewriter input is involved, the language is usually designed as a string of characters.

Structure of Program

The structure of the program is based on the types of subunits that are permitted, their characteristics, and the ways of intermingling them. The types of subunits that are normally allowed are declarations (including data descriptions); executable units; loops; functions, subroutines and procedures; the complete program; and the method of interacting with the operating system and the environment. The characteristics of subunits involve the methods of delimiting them, whether they are recursive and/or reentrant, the types of parameter passage required, what types of executable statements and declarations can be combined, and what other languages, including possibly machine language, can be included.

Data Types and Units

The different types of data variables include the following: arithmetic, Boolean, alpha-numeric, formal (=algebraic), strings, lists, vectors and arrays, hierarchical. A key characteristic is what types of data units can be accessed by the commands in the language. Thus one needs to know what data units of the hardware, as well as what variable types, can be obtained through the use of a particular command. Many different types of arithmetic can be performed, e.g. fixed and floating point, rational arithmetic, multiple precision, complex number, etc. Along with all these different types of arithmetic and different types of data go various rules on modes such as rules on which data types can be combined, and conversion and precision rules for computation.

Executable Statement Types

Many people feel that a programming language is completely determined by the executable statements it contains. Certainly, the executable commands play a very significant role, but they are not the only parts of a language. This is rather a narrow view.

The major classes of executable statements are assignment statements, those for handling alpha-numeric data, and those for sequence control and decision making. Almost all programming languages have all of these in some form or other. An increasing number of languages include executable statements for handling various types of symbolic data, e.g. algebraic expression manipulation statements, and statements to handle lists, strings, and/or patterns. Finally, with the increasing complexity of hardware and operating systems,

(Please turn to page 37)

* Some of the terms used here may not be self-explanatory, and lack of space prevents adequate definition. It is hoped that the reader will get the general concepts even if a few specific words are not understood.

MULTI-PROGRAMMING: WHO NEEDS IT?

Brooke W. Boering
Systems Analyst and Senior Programmer
Talman Savings and Loan Assn. of Chicago
Chicago, Ill. 60629

"To a medium-scale commercial user who is considering going on real-time, the issue of reliability takes on a dimension quite different from that encountered previously."

Numerous papers have dealt with the admittedly significant successes of large scale multi-programmed installations of recent date. As a result there seems to be developing a cult whose credo is that "If it isn't multi-programmed, it's out of date". It is perfectly true that both time-sharing systems and the behemoths of batch-processing can and are benefiting mightily from the maturing of multi-programming (henceforth called "m-p"). In the case of time-sharing m-p is, of course, central to the very concept. Devotion to expanding the understanding and use of m-p in such installations is virtually a "must". With these remarks, I have now made the appropriate reservations, set forth the more obvious exceptions, and generally rendered unto Caesar that which is Caesar's.

Potential Dangers

But now I must point out that there are potential dangers in assuming that m-p applies in *all* installations. Manufacturers of computers today are promoting m-p for medium-scale (half megabuck) systems, especially where there are plans for both real-time and batch processing. So let's re-examine the fundamentals of systems design and of return on costs, in the light of not only the advent of m-p but also current changes in equipment, prices and capacities.

First, let us look closely at what m-p does for time-sharing and for very large-scale users. In time-sharing, each request for processing by a remote (or local) terminal results in a "job" to be handled by the system in the manner most efficient to the total environment based on the demands of the moment and the condition of previously commenced "jobs" from other terminals. Neither the timing nor structure of such jobs can be determined at the time of systems design, so it makes good sense to provide an extensive set of gear, and an "executive" program which can adapt on a dynamic basis to rapidly changing demands of the system. The technique of "slicing" a job into sub-jobs which has received much attention is secondary to this prime purpose of "gear sharing" which, of course, preceded time-sharing in development.

Time Slicing

The "time-slicing" technique is necessary where jobs can use a sizable chunk of time of the central processor unit. In the commercial counterpart to time-sharing, namely, real-time business systems, the individual job demands on the CPU are seldom extensive enough to require temporary abandonment unless there is "wait" time for an I/O operation. Nearly all real-time systems are multi-programmed to permit overlap of various real-time "jobs" emanating from many terminals within the system. However it is not *this* definition of m-p that concerns us but rather the broader concept that involves both real-time (whether time-shared or not) and "batch" processing within the same CPU. This concept is, in effect, the most complex yet; for it purports to handle in a sensitive on-line system "batch" programs written without concern for the continuity requirements of real-time. Such systems exist, of course, and do the job as advertised. What is not quite so apparent is the real costs to the medium-scale user looking at such a system as compared to a more conventional approach.

Myth of Precious Central Processor Time

The myth of "precious" CPU time requires almost immediate examination, or it will continue to distort more practical considerations throughout our appraisal. With the advent of "unlimited use" leases and the increasing portion of total gear cost being borne by the peripherals in the latest systems, it is becoming apparent that CPU "pulsing" costs are no longer the dominant consideration.

Requirements for Multi-Processing

The elements required to operate a m-p environment for simultaneous real-time and batch processing are:

- Enough core to contain the resident real-time system, the batch program, and the m-p executive. Trade-offs are possible to reduce core requirements, which generally involve penalties of increased secondary memory, reduced efficiency of response times and/or need for increased processor speeds, all of which cost money, as does core.
- Sufficient dual-purpose peripherals to enable both systems to overlap and to minimize "idle" units. In actual practice, few real-time and batch multi-programmed systems utilize either common input/output devices or a common data base except for the m-p executive itself.

From this it can be seen that the m-p advantage of dynamically allocating such equipment is of little value here.

- Possibly increased CPU speeds to absorb the housekeeping operations of m-p. (Check the price differences between a mod-30 and mod-40 or a 415 compared to a 425 mainframe.)
- The latest hardware "goodies" in the form of base register modification, memory protect, interval timers, nonstop mode, etc. Again, nothing's free, nor should it be.
- The "complexity" introduced to the user's installation. This is probably the most difficult element to assess properly. Although promised (and even delivered) by the manufacturer, the m-p executive for an installation of the size under consideration here must be just as complicated as that required for the most sophisticated (and expensive) installations going. Typically, medium-scale users have limited staffs who are close enough to the user problems to do a good systems job but seldom have extensive background in software design or system maintenance. This would not be a factor if all software were delivered letter-perfect by the manufacturer and/or his subsequent support of said software was consistently beyond reproach. Experience however indicates otherwise.

Unreliability and Headaches

The factor of complexity brings us to the element of reliability. A software failure of the m-p system that stops the real-time system from operating must be viewed differently from our batch-oriented headaches of bygone days — when we had all night to find the bug.

To our hypothetical medium-scale commercial user who is going real-time, the issue of reliability takes on a dimension quite different from that encountered previously. If something goes wrong with the data-processing environment (software or hardware) which cuts off real-time service, he has extended his troubles *right this minute* to the front line of the enterprise. In the case of an airline, it's the reservation system that does not work; for a bank, it's probably the savings accounts which are now inaccessible; a large wholesale distributor finds he no longer can determine his inventory condition. The data processing manager in such situations has simply *got* to suffer some degree of anguish, embarrassment, or worse, as a result of such occurences. Naturally, manual "back-up" systems are devised — but, like all such unpractised systems, they are all but forgotten at the time they are most needed. Even when usable, the manual systems are only barely tolerable for only the very shortest times. This entire sensitive mater of reliability takes on a degree of importance not easily equated directly to the costs of the system alone.

Dividing the Day

Let's now turn to the obvious alternatives to m-p for the medium-scale user who needs both real-time and batch capability. If the data processing demands are modest enough and there are no time conflicts, he might devote the system to real-time processing during regular business hours, and leave the batch processing to the night shifts. Such an approach has some disadvantages. There is the psychological one of requiring operating and programming personnel to work undesirable shifts on a regular and continuing basis. There is the possibility of losing the best people to other installations not having such problems. Another limitation is that the real-time system cannot operate 24 hours per day and also do batch work in a non m-p mode. Such round-the-clock applications for real-time leave no time for after hours "batching." Even when some time remains, say 6 or 8 hours per night, it might prove insufficient to accomplish the off-line processing needed to support the next day's operations.

The Duplex Installation

One configuration that deserves serious consideration as an alternative to m-p is commonly referred to as a "duplex" installation. Here the principal feature is the presence of two main frames, one normally devoted to the real-time operation while the other normally does "batch" processing.[1] Each can operate independently of the other under normal circumstances; and yet the "batch" CPU stands ready to abandon its non-time-critical work to take on the real-time operation should the other processor fail. For this back-up function to operate properly requires equal mainframes in core size, communication capability, etc. Once such a concept of separation, simplicity and back-up is given serious consideration, what seemed to be impossibly costly configurations often come into focus as the cheapest in view of *total* costs.

So we assert that a 2% usage of CPU time just might be economical under certain conditions, and thereby we incur the wrath of the entire Association for Computing Machinery. But we will now attempt to make peace with the manufacturers, who have perhaps been treated rather shabbily, by recommending to medium-scale users contemplating new real-time, commercial systems to "take two, they're small"

[1]See Robert V. Head, "Real Time Business Systems".

FUNDAMENTAL CONCEPTS — Sammet

(Continued from page 35)

the programming language itself must include a number of statements to provide interaction among these. Specifically, there are apt to be statements dealing with input/output, library references, debugging, storage allocation and segmentation, and finally statements which deal directly with the operating system and/or machine features.

Declarations and Non-Executable Statements

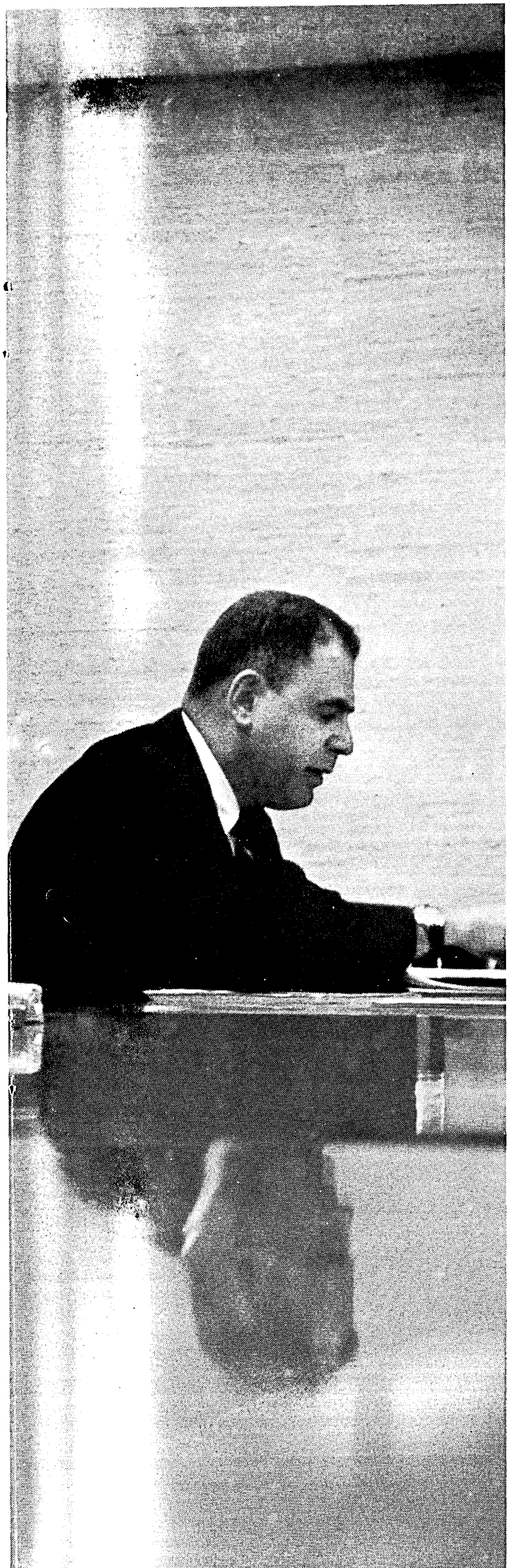
Although the executable statements are obviously essential in order to accomplish anything, in general they cannot operate without knowing something about the data on which they are to act. For this reason, most programming languages have either primitive or complicated declarations to cover such things as data and/or file descriptions, format descriptions for input/output data, and storage allocation.

Structure of Language and Compiler Interaction

Among the characteristics of programming languages are the abilities for self-modification of programs, or self-extension of the language. Many languages contain directives to provide the compiler with useful information. Consideration must be given to the effect of the language design on compiler efficiency, and to the inclusion of debugging aids and error checking. A key characteristic of more theoretical than practical interest is whether the language can be used to write a compiler for the language. Finally, an interesting characteristic of the language is whether or not it is useful outside of its primary application area.

Conclusions

The widespread use of programming languages has given tangible proof that in most cases the advantages far outweigh the disadvantages. While obviously there are specific instances in which no existing higher level language will serve an existing need and therefore an assembly language must be used, these situations are becoming fewer. Careful examination of languages with respect to their non-technical as well as technical characteristics should help avoid incorrect choices by users.



THE CASE FOR COBOL

S. M. Bernard
Systems Engineer
IBM Data Processing Division
Newark, N.J.

"Here is one high-level language with many features integrated in it, that, if used appropriately, can provide significant assistance to any commercial computer users in the areas that in the past have caused the bulk of implementation problems."

With the advent of third generation computers, there has been increasing emphasis on higher-level commercial programming languages. In particular, the improved quality of COBOL compilers on third-generation equipment has caused many accounts to select COBOL as their primary commercial language. This in turn has renewed, with increasing vigor, the arguments for and against COBOL as a useful and worthwhile language.

At this point in time, the debates on this subject are reminiscent of discussions held a few years back on symbolic language coding versus machine language coding. As was true in the latter case, many are analyzing the newer approach merely as a modification or extension of the existing system, rather than taking an overview of the new technique prior to making any sort of comparison.

If a one-for-one element-by-element comparison is made between COBOL and a previous assembly language, COBOL may very well suffer by comparison. The following conclusions may legitimately be drawn from this kind of an analysis:

- COBOL takes longer to code.
- COBOL is much too wordy.
- A good assembly-language programmer can produce more efficient code than a COBOL programmer.
- An assembly language with powerful macros would be better than COBOL.

All of these conclusions may be valid for installations which attempt to use COBOL without first appreciating the fact that COBOL is indeed a different type of language and that it must be utilized differently before its benefits can be truly realized.

In proving this thesis I will describe how COBOL applications should be implemented. To do this I will use System/360 DOS COBOL as a frame of reference. This particular language is my selection for a few reasons:

- It is the one with which I am most familiar;
- It is probably the most frequently used COBOL compiler released to date; and
- It contains certain optional elements which I feel are vital if one is to use COBOL effectively.

The language against which I will compare this compiler will be 1401 Autocoder. My reasons for selection here are the same as the first two reasons above.

The Typical 1401 Autocoder Installation

If we were to represent a typical 1401 Autocoder installation activity, assigning relative time units to each element involved in converting an application, it might look something like Figure 1. In this example, three units of time are spent in application flow, file design, and the like.

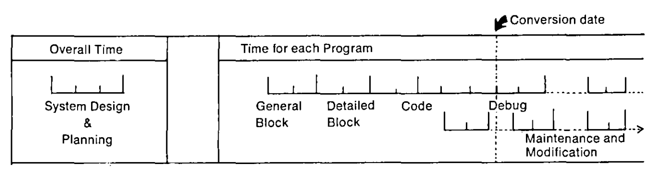


Figure 1. 1401 Autocoder Installation

For each program in the system, two units of time are each spent for general blocking, detailed blocking and coding. At this point initial debug time is started and will of course be overlapped with other programming being done. It does however continue for a considerable time (even after conversion) and will never be done until all program legs (including modifications and patches) are completely tested.

S. M. Bernard is a B.S. from Rutgers University (1960, Economics) and an M.B.A. from Fairleigh Dickinson University (1964, Business Management). He was an instructor in data processing at Fairleigh Dickinson University. Mr. Bernard, who has been a Systems Engineer with the New Jersey Manufacturing Office of IBM since 1961, has installed eighteen data processing systems including 360's. He has written several articles and lectured often on subjects pertaining to System/360, COBOL, and direct access devices. He is currently preparing a COBOL Programmers Guide soon to be published.

Maintenance time will vary considerably but can take a significant amount of time, and, on a given program, may continue indefinitely.

Using COBOL as an Improved Autocoder

Now, let us assume that this application were to be done in COBOL by one who concluded that COBOL was a panacea — a language that would allow us to achieve the same goals in basically the same way, but in a shorter period of time. Our bar chart might look something like Figure 2. Note that there are significant changes in certain areas and no change in others. System design would remain unchanged, as would general and detailed block time. Coding time would increase slightly due to the wordiness of COBOL — especially in the Data Division of the program. But, because of its self-documentation and machine independence, debugging and maintenance time would decrease. Total time for implementation would be about the same but the critical areas would be cut down: modification and maintenance would improve, debugging someone else's program would be far easier, careless errors would be reduced, communication with nonprogrammers would improve, and machine-to-machine compatibility would increase. The cost for these benefits would be increased coding time and perhaps less efficient programs (more core, slower execution).

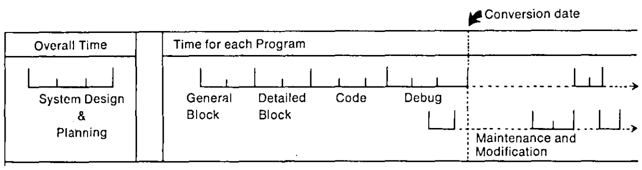


Figure 2. System/360 DOS COBOL used as if it were an improved 1401 Autocoder

Using COBOL as COBOL

Prior to describing how these relative times can be dramatically altered, I must expand a point which was made before: that System/360 DOS COBOL contains items vital to effective COBOL use. (It is true that many other compilers may have some or all of these features, as well as some additional and very useful ones not described herein.) These items must be described and understood prior to a discussion on how COBOL should be used.

Vital Elements of a COBOL Compiler-Source Library Facilities

It is true that COBOL requires lengthy record descriptions. It is also true that the same files and work areas are used from program to program. If the record descriptions can be coded once and placed in a source statement library, then all each programmer would have to do in a given program would be to code single statements that would copy selected descriptions from the library. Any programmer could use any standard record description by using a single 'COPY' statement that would retrieve the entire lengthy description from the library.

Since the Data Division of a COBOL program can take from 30 per cent to 75 per cent of the total coding time, this will result in considerable coding time saved. It will also reduce maintenance time drastically. File changes, such as new fields, or making a field larger, or making one longer requires only that the source library be modified and existing programs be recompiled. No program changes need be made. We can therefore reduce coding time, debugging time and maintenance time and improve standardization and documentation by using source library facilities.

To do this, however, we must be certain that our files are completely designed, coded with standard, non-conflicting names, and put in the source library prior to program coding. This means that an additional investment is required in the planning and design phase to improve the quality of the remainder of the effort.

Similar benefits can accrue from source statement subroutines which can be coded once and placed in the library for common use. This, too, requires a one-time investment in the initial planning stages.

Object Library Facilities

The ideal way to write a program is to break it down into a series of independent modules or subprograms. If these modules can be separately compiled modules, the benefits which accrue are immediately obvious:

- Programs can be tested prior to all modules being written.
- Work can be divided among programmers easily.
- Routines that can be handled better in other languages can be written in other than COBOL.
- Compilation time decreases.
- Maintenance on a given module can be done independently of unaffected modules.

COBOL gives this facility via "CALL'S" and linkage handling. Consider the feasibility of writing a payroll program with certain state taxes as independent modules, or of writing a common module to handle a specific I/O device such as the printer (counting lines, inserting headings, putting printer output on disk for later 'spooling', etc.), or a subprogram for table look ups or for other commonly used functions. Savings could be significant, and will be if an application is planned well.

Debugging Features

To cut down debugging time, it is mandatory that there be COBOL words for that function. You cannot have source-language debugging without these facilities. Programmers who debug COBOL programs with object code dumps might be better advised to use an assembly language.

Debugging a program consists of two phases: getting a successful compilation; and getting successful executions. The first task is aided by the COBOL compiler; the second by COBOL debug words.

It is my opinion that no DOS COBOL program should be submitted for its first test without a "READY TRACE" instruction at the beginning of the procedure division. This debugging aid, which prints each paragraph name as that paragraph is executed, can help cut debugging time down to almost nothing.

Similar benefits are realized from other COBOL words (EXHIBIT, ON COUNT, DISPLAY) which help to relegate debug time to a minor portion of the installation cycle. Anyone who discusses COBOL without an appreciation for its debugging words is obviously not doing justice to the language.

Powerful Words

All COBOL compilers have powerful words, but they are useless until applied to an application. Coders with previous programming experience will often code down to an unnecessary level of detail.

Too often one may write:

```
MOVE FLD1 to FLD2
ADD 10 TO FLD2
MULTIPLY FLD2 BY 13.5
SUBTRACT FLD1 FROM FLD2
```

Instead of: COMPUTE FLD2 = ((FLD1 + 10) * 13.5)-FLD1

Similarly, programmers may shy away from GO TO DEPENDING, 88 CONDITIONALS, TRANSFORM and many other words which could be used very effectively in certain situations. To use COBOL well, one must understand when they should be used as well as why.

Coding COBOL Effectively

One must also realize that program structure in COBOL is most important. Certain rules should be followed in COBOL as in any other language. It is a good idea to take time during the planning stages to develop guide-lines and to make certain that all programmers are aware of these rules and how to apply them.

The use of the PERFORM should be very carefully analyzed. This word allows you to divide a given program into a mainline and a series of independent sub-routines. Yet, experienced coders too often use GO TO where PERFORM might better have been used. This is because they are accustomed to "branching" in previous languages.

This is not a minor point. A complete understanding of IF statements in conjunction with PERFORM will reduce block diagramming time. It should eliminate detailed blocks completely. It will also cut down debugging time by allowing the COBOL compiler to generate "Branch" and "Return" instructions as opposed to the programmer being responsible for these functions. Well-written programs will obviously take less time to debug or to modify; they will also be self-documenting.

Additional Considerations

Other rules should be developed by the installation to help realize these goals. Standard techniques for naming fields and paragraphs should be devised so that all can understand them. Coders who "GO TO P0/3A04" are obviously not documenting their program as well as those who "GO TO 17-COMPUTE-FICA."

Spacing and numbering of statements should also be in accordance with predetermined rules.

Apply These Concepts

Now let us assume that an installation utilized COBOL as described above; how will their implementation cycle change and why? Their cycle should look like Figure 3.

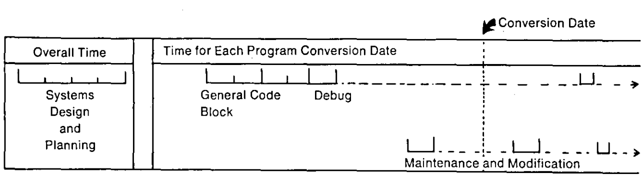


Figure 3. Using COBOL as COBOL

Systems Design. This time will increase due to additional efforts required for standardizing, for setting up file descriptions, for developing guidelines, etc.

Block Diagramming. Only General Block Diagrams need be done. There should only be a series of Block Diagrams each describing a module (main-line, PERFORMed sub-routine, or CALLed sub-program)

Detailed Block. Unnecessary if those who do the general block understand COBOL.

Coding. Coding will decrease slightly due to the use of source libraries, standard naming, powerful COBOL words, and well-structured programs.

Debug. Because of structured programs, COBOL self-documentation, DEBUG words, and the like, debugging time should be drastically reduced. Source-level debugging is a reality.

Maintenance and Modification. Everything described above lends itself to improved maintenance facilities. Documentation, source library usage, structured programs, CALLed and PERFORMed independent modules, source debugging, installation standards, and the rest provide an excellent base which will facilitate future changes — even by persons other than the original programmer.

Conclusion

It is unfair to conclude without mentioning that there are, as in any languages, disadvantages in System/360 DOS COBOL. Certain things cannot be done easily or efficiently — such as subscripting, manipulating bits, accessing machine addresses, sign handling, clearing large areas, handling interrupts, etc. Other things that can be done require too many words, or excessively long words. Nor can I propose that it be the only language to be considered by a given user. RPG, assembly language, and PL/1 on this particular system should be used where they achieve a given goal more satisfactorily. But what is important is a realistic evaluation of the language as a whole and of those elements which are built into it for us to use. All the things described in this article have been implemented in other languages, at least in certain installations, but here is one high-level language with many features integrated in it, that, if used appropriately, can provide significant assistance to any commercial computer users in the areas that in the past have caused the bulk of implementation problems.

THE SECOND INDUSTRIAL REVOLUTION

"Your suggestion on how to weld our gas pipes automatically was brilliant, Jackson. And I wish you luck on finding another job."

ACROSS THE EDITOR'S DESK

Computing and Data Processing Newsletter

TABLE OF CONTENTS

APPLICATIONS

STATE OF IOWA HAS STATUTE RETRIEVAL SYSTEM

The Iowa Legislative Research committee has announced details of an electronic statute retrieval system that puts the research power of a computer at the fingertips of the state's 61 senators and 124 representatives. Sen. John P. Kibbie, D-Emmetsburg, chairman of the bipartisan joint committee, said, "The 2988 pages of state laws and the 30 pages of the Iowa Constitution have been indexed and electronically stored in an IBM computer. Any section pertinent to pending legislation now may be called out merely by coding in key words. The computer searches the law and prints out, at the rate of one typewritten page per second, those sections dealing with the subject under consideration." When the General Assembly convened January 9th (less than a year from the start of the project), any lawmaker, committee, agency or state university had access to the working system.

Key item in the law retrieval system is an IBM System/360 Model 40. Gene Reyhons, Legislative Research Bureau director, said: "In addition to tremendous time savings, the new computer is completely accurate in its reporting for the Legislature if the system is used properly." The new system frees researchers from laborious, manual searching. "Our staff members will devote their time to more demanding tasks, such as preparing and sharpening questions to be put to the System/360," Mr. Reyhons said. "The appropriateness of the answers is directly related to the degree of accuracy and explicitness of the question.

"We have a 250-page index of words and phrases," Mr. Reyhons said. "From it we can locate those sections of the law and the constitution pertaining to the question at hand. The system requires no knowledge of computers, but our researchers must be familiar with the law and with traditional search techniques." For example, he said, a lawmaker asking for Iowa statutes concerning fences will receive, overnight, printed sections of the law containing the words "fence," "enclosures," and related terms.

The computer is under the auspices of State Comptroller Marvin R. Selden, Jr. and will be used for many tasks in addition to statute searching. "Our centralized data processing services are used by 36 agencies of state government," he said...."We add more applications every year as we continue centralizing State of Iowa data processing activities."

PAN AMERICAN'S NEW AIR FREIGHT TERMINAL

The largest and most sophisticated air freight terminal in the world, costing in excess of $8.5 million, has been opened by Pan American World Airways at New York's Kennedy International Airport. The terminal, a culmination of over five years planning and two years in construction, will permit Pan Am to handle cargo volume projected at nine times the airline's current volume in ten years. This new facility utilizes a computer system in conjunction with a network of three automated mechanical cargo handling systems.

The cargo terminal is geared for both export and import freight, capable of moving cargo direct from truckside on pallets or containers into 90,000 pound capacity Pan Am jet freighters. With two jet freighters positioned at docks extending from the building, the system provides for simultaneous loading or unloading of each aircraft. Both jet freighters can be unloaded in 20 minutes — involving a movement of 45 tons for each aircraft. The cargo terminal provides 22 complete enclosed truck delivery and pickup docks.

The cargo system was designed by Abbott, Merkt & Co., New York, to process more than ten times the amount of cargo the airline could in its old Kennedy freight terminal. Heart of the cargo handling system is a package conveyor arrangement, an automatic tow cart network, and an AirPak pallet system.

Nerve center for the terminal is the airline's multi-million dollar computer system, located in the Pan Am Building. Backbone of the system is an IBM 7080 computer and an IBM 7750 programmed transmission control unit. This equipment is linked by high speed telephone lines to two Bunker Ramo control units at the cargo terminal. The Bunker Ramo control units in turn are connected to some 26 input/output sets (also manufactured by Bunker Ramo) which are strategically placed throughout the terminal.

Primary aim of the computer/handling system network is to speed the flow of freight through the terminal by providing instant read-out of freight inventory, handling of reservations, and simplifying the dispatch and tracing of cargo movements.

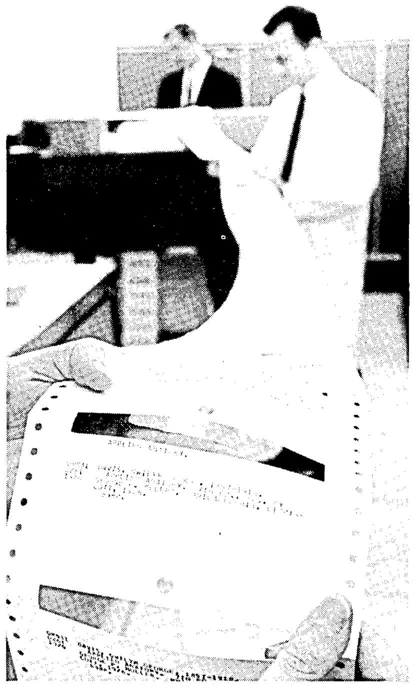
MEDICAL COLLEGE OF VIRGINIA COMPUTER TO PRINT LIBRARY CATALOG, INDEX FOR DOCTORS

The Medical College of Virginia, Richmond, Va., has begun using an IBM 1410 computer to index its 80,000 library books and ultimately plans to put the index into the hands of the state's doctors. The computer is being used to produce library catalog cards, and by reclassifying the same information into page form, also to produce a publishable book catalog.

The computer-aided recataloging of the library, which is still in the pilot project stage, is using subject headings designated by the National Library of Medicine in Bethesda, Md., to meet the needs of the medical profession. The first step in the recataloging of a volume begins at the library where Mrs. Louise Bryson, project supervisor for computer cataloging, determines the information to be entered on each card, appropriate coding, and required cross-reference headings.

The library information is punched into cards at the computer center. The punched card system simplifies updating or correcting single lines or entire entries. The computer sequences the information and automatically orders printing of additional cards according to the coded variety of subject headings entered for each volume.

The index cards, giving the titles, subjects and call numbers of the books in the Tompkins-McCaw library, feed out of the computer's high-speed printer — as fast as 40 a minute. The cards have been especially die-cut to fit standard library catalog drawers and travel through the printer in streamers, carried along on perforated tracks that are stripped off afterward.



— A high-speed printer directed by an IBM 1410 computer produces streamers of index cards for the Thompkins-McCaw library

The computer also provides an abridged listing without cross-reference subject headings as the format for a publishable book catalog. The first index has been published and circulated to the staff of the Medical College of Virginia.

"This book catalog simplifies the search for information by the staff outside the library," said Mrs. June Leath Huntley, library director. "We decided to recatalog according to the latest standards, and began work in May 1966. We hoped then we could simultaneously produce a card and book catalog, and thus far the system has demonstrated it can."

CONTROL DATA COMPUTERS REGULATE TRAFFIC IN DETROIT EXPRESSWAY TESTS

R. D. Schmidt, vice president of sales for Control Data Corporation, has announced that one of his firm's electronic computers is being used to regulate traffic on test sections of a Detroit (Mich.) expressway. The equipment is being studied as part of a national program designed to develop new methods of easing metropolitan traffic problems. The Detroit tests are being partially sponsored by the U. S. Bureau of Public Roads, in cooperation with city, county and state agencies.

The traffic surveillance system in Detroit involves a 3.2 mile portion of the John C. Lodge Freeway. It employs a Control Data computer system and closed-circuit television for visual monitoring and regulation of traffic. Electronic sensors located out on the expressway report the movements of vehicles on the freeway to the computer. Using this information, the computer then calculates the volume of traffic passing specific points every minute, the percentage of pavement occupied, and the actual speed of vehicles as opposed to the posted speed limits.

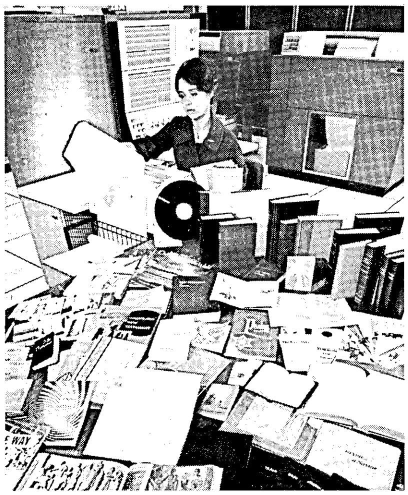
Armed with this information and visual checks, via TV, of the changing traffic picture, the system's human operator can regulate freeway traffic with electronically controlled signs. These signs alert motorists to changes in minimum and maximum speed limits, and indicate open and closed lanes. For example, 'lane closed' signs divert traffic into other lanes long before it reaches the congested area. If the freeway is loaded beyond its capacity or an emergency situation exists at a certain point, the human operator at the system's controls pushes a button to activate 'ramp closed' signs. By obeying the signs, motorists, in effect, regulate themselves and restore the freeway system to maximum efficiency.

The control center which houses the computer for the Detroit experiment has become an important source of information for the city's Traffic Central. One of Traffic Central's function is providing radio stations in the area with current information on freeway conditions. The control center also maintains a direct line to the Freeway Patrol Division of the Detroit Police Department.

Data collected by the computer system in Detroit will be used in continuing traffic engineering studies. These studies are aimed at developing an automated traffic control system, in which the computer will directly control the traffic signals. Proponents of computer-controlled traffic systems predict such equipment will provide new levels of speed, service and safety for metropolitan motorists.

COMPUTER-AIDED BIBLE DISTRIBUTION

The American Bible Society, New York, has an IBM System/360 at its headquarters to handle the complex logistical problems involved in speeding millions of Bibles to people throughout the world. The Society is the world's largest publisher and distributor of Bibles, Testaments and portions of Scripture — 75-million copies to 150 countries in 1966. The computer will keep track of the more than one-million copies of the Scriptures which are mailed out each week from the Society's World Distribution Center in Wayne, N.J.



The total supply of Bibles and selections from the Bible at the Wayne warehouse is maintained at about 27-million.

At least one entire Book of the Bible has been published in 1250 languages and dialects, in Braille (in 50 languages) and on records. American Bible Society publications also are available in many forms — from the standard King James Version of the Bible to modern translation of the Scriptures featuring contemporary photographs of the Holy Land. When the stock-level of any version in any language has dropped below a certain point, the System/360 Model 30 automatically will notify the Society's management and will handle all administrative details such as the preparation of invoices, packing slips and shipping labels.

Everett Smith, president of the American Bible Society, said that the use of a computer to help distribute one of man's oldest and most venerated books represents the Society's drive to make the Bible as accessible and meaningful as possible to today's generation.

CHRYSLER CORPORATION'S DYNAMIC INVENTORY ANALYSIS SYSTEM (DIAS)

Chrysler Corporation (Detroit, Mich.) has unveiled an instant information computer network which supplies up-to-the-minute inventory and shipment data on any of more than 13,500 parts used in its 1967 passenger cars. The new system, called Dynamic Inventory Analysis System or DIAS, is linked to 77 independent suppliers, 26 Chrysler parts manufacturing plants and the company's seven car assembly plants with the nerve center at the Car Assembly Group headquarters in the Hamtramck (Mich.) Assembly Plant. The plants linked into the computerized network are located in 18 states and the Province of Ontario Canada.

"We now have the means to obtain information in a fraction of a second on how many parts are on hand at key supplier plants and in each of our assembly plants, how many are on the road in shipment and to which plant they are being shipped," said Joseph F. Kerigan, Chrysler Corp. vice president and group executive - Car Assembly Group. "We believe this is the world's largest tele-processing network for gathering and disseminating commercial-manufacturing information."

The nerve center of the DIAS network is an IBM 360 computer that has the capability to simultaneously receive 31 input data transmissions over regular telephone lines. A key part of the computer control is an electronic data bank that can receive input data at more than 156,000 characters per second and has a memory core that can store 100 million characters to call upon for its decisions.

In describing the operation, John J. DiCicco, manager of operations analysis and development staff for Chrysler's Car Assembly Group, said the computer system is so fast that it often starts transmitting the answer to a query back to the operator at the very instant the operator has finished transmitting the query to the nerve center.

DiCicco pointed out the ease and simplicity with which a supplier can tie into the network. The system permits the supplier to use normal business methods of recording production and transmitting punched card information over normal telephone lines to the Chrysler computer.

DIAS is so well disciplined that it can detect its own errors or malfunctions such as the breakdown of one of its tele-processing units. It can scan all information for accuracy as it is being received and locate errors such as wrong part numbers, invalid supplier codes or even typing errors. The system will respond with a printed notation of the error and ask the supplier to review his input data for correctness.

COMPUTERS AID IN SEARCH FOR MALARIA CURE

New forms of drug-defying malaria, the ancient disease once thought conquered by wonder drugs, is again taking its toll at an alarming rate in Southeast Asian battle zones. Today, however, doctors and chemists have a new and powerful ally in their search for a more effective cure for the newly developed strains of malaria.

Researchers at the Walter Reed Army Institute of Research in the nation's Capitol are using computers to probe the effects of over 100,000 organic chemical compounds in such areas as malaria, radiology and schistosomiasis.

Using a Compound Structure File, a computerized information-retrieval system developed by The Service Bureau Corporation (SBC), researchers can query their computers about relationships between complex chemical compounds. To assist them, the computer's answers are printed in a chemical typeface — the geometric shapes which represent recognizable chemical structures to doctors and chemists.

With the SBC file, a researcher who is having some success with a particular compound can check the computer for a complete listing of all chemical compounds which use that particular structure in their chemical makeup. Using the SBC file, a researcher could pinpoint compounds which may be more effective as drugs, or which may not have undesirable side effects.

Meanwhile, Walter Reed has some 300 commercial and hospital laboratories forwarding the results of various drug experimentations to them. From this data, SBC continuously updates the computer's master files.

NEW CONTRACTS

FROM	TO	FOR	AMOUNT
French Navy, France	Sperry Rand Corp.'s UNIVAC Defense Systems Division	Shipboard computing equipment: some already delivered feature UNIVAC 1206 computer; new system on order designated "B-2" by the North Atlantic Treaty Organization features the UNIVAC CP-642B computer	over $4 million
Lockheed-Georgia Co., Marietta, Ga.	Northrop Corp., Palos Verdes Peninsula, Calif.	Sixty-seven airborne computers for the U. S. Air Force C-5A military transport airplanes	about $3 million
Bell Telephone Laboratories	Ampex Corp., Redwood City, Calif.	Development of a videotape system to be used to store and retrieve maintenance data for the NIKE-X Weapons System	about $2 million
General Electric, Computer Equipment Dept., Phoenix, Ariz.	Data Products Corp., Culver City, Calif.	Linear positioning actuators to be used on the DSU-200 disc type random access memory storage system in production by GE	over $1.5 million
A. B. Scania-Vabis, Sodertalje, Sweden	Westinghouse Electric Industrial Systems Scandinavia A.B.	Design and installation of fully automated, computer-controlled warehouse	over $1 million
Flying Tiger Line, Los Angeles, Calif.	Link Group, General Precision Inc., Binghamton, N.Y.	A DC-8 digital flight simulator	about $1.5 million
Lockheed Missiles and Space Co., Sunnyvale, Calif.	Computer Division, Electro-Mechanical Research, Inc., Minneapolis, Minn.	Fifteen computer systems including EMR ADVANCE 6020 and 6040 computers which will be used in Lockheed system for checkout of the Poseidon missile	—
Navy Purchasing Office, Washington, D.C.	The Bunker-Ramo Corp., Canoga Park, Calif.	The fabrication of a Remote Query Display System which will interface with an existing 1410 data processing system	$467,000
Goodyear Aerospace Corp.	Systems Engineering Laboratories, Inc., Fort Lauderdale, Fla.	SEL 840A general purpose computer which will be used in Device 2H87, Aircraft Carrier Landing Trainer	$200,000
Louis Breguet Aviation Manufacturing Corp., Breguet Research Laboratories, Velizi-Villacoubly (near Paris) France	Electronic Associates, Inc., West Long Branch, N.J.	EAI 680 Analog/Hybrid Computing System for aviation research in fields of aircraft design and flight control, as well as for general simulation of aeronautical problems	$122,000
Chemical Abstract Service, Columbus, Ohio	Mohawk Data Sciences Corp., Herkimer, N.Y.	Thirty more Data-Recorders to be used for recording chemical information directly on computer magnetic tape	—
Rome Air Development Center, Griffiss Air Force Base, N.Y.	Sylvania Electric Products Inc., Sylvania Electronic Systems, Waltham, Mass.	Studying methods to simplify data processing by enabling computers to "read" hand-printed information	—
Telecredit, Inc., Los Angeles, Calif.	The Service Bureau Corp.(SBC), Los Angeles, Calif.	Total data processing support for Telecredit's check verification and insurance services	—
University of Illinois, Dept. of Computer Science	Fabri-Tek Inc., Minneapolis, Minn.	Two core memory systems for the Illiac III computer	—

NEW INSTALLATIONS

AT	OF	FOR
NASA Computation and Analysis Division Facility, Houston, Texas	EAI 8900 Hybrid Computing System valued at $1.7 million	Simulating critical docking maneuvers for project Apollo
Parkview Memorial Hospital, Fort Wayne, Ind.	GE-115 computer system	Maintaining and analyzing medical records and to process financial accounting data; also to assist in special diet and menu planning, and to schedule preventive maintenance on the physical plant
Mack Trucks, Inc., Parts Service Operation Center, Somerville, N.J.	GE 415 computer	Keeping track of some 80,000 spare parts and supplies for shipment to customers around the world
Northwestern Drug Co., Tacoma, Wash.	Two NCR 315 systems	Handling data processing of firm's wholesale operation; also to serve area retail druggists. One system is installed in Tacoma, the other at firm's Portland facility
Fedder Data Centers, Inc., Baltimore Md.	NCR 315 computer	Additional capacity to its lineup of data processing equipment
Electricite de France (French Power Bureau), Paris, France	Control Data 6600 Computer System	Wide range of applications including electrical network power distribution studies, power plant engineering, economic studies, as well as for scientific computations

AT	OF	FOR
NASA's Marshall Space Flight Center, Huntsville, Ala.	EAI 8900 Hybrid Computing System valued at $1 million	Application to studies in the Saturn booster program; also for studies on reclaiming/re-use of boosters, and for multi-staging control studies
The Fairview Hospitals, Minneapolis, Minn.	IBM System/360 Model 40	Patient and laboratory records-keeping
U. S. Navy Ships Parts Control Center (SPCC), Mechanicsburg, Pa.	IBM System/360 Model 65 valued at $1.8 million	Assistance in world wide management of 300,000 line items valued at over $3 billion and ranging from ships and ordnance parts to ammunition and guided missiles
Brookhaven National Laboratory	Two Sigma 7 computers	Helping to explain some recently discovered inconsistencies in the laws of nature
C-E-I-R, Inc., Manhattan, N.Y.	IBM 360/65 computer valued at about $2 million	Replacement of IBM 7094
Michigan State University, East Lansing, Mich.	SDS Sigma 7 computer	Studies aimed at understanding the physical forces that hold atomic nuclei together
Texas Electronic Company (TRACOR), Austin, Texas	Sperry Rand UNIVAC 1108 Computer System valued at approximately $2 million	Serving all of TRACOR's scientists; eventually becoming accessible to all of company's four subsidiaries, two divisions and five branch offices and laboratories via telephone lines and remote communications devices
Cerritos College, Los Angeles County, Calif.	Honeywell 200 system	Administrative and instructional purposes; primary use will be instructional
Goldblatt Bros. Inc., Chicago, Ill.	IBM System/360 Model 30	General accounting, accounts payable, employee discounts and general inventory control
Van Waters & Robers, Inc., San Francisco, Calif.	IBM System/360	Controlling inventory of more than 30,000 items ranging from laboratory glassware to $50,000 electron microscopes
Investors Diversified Services (I.D.S.), Minneapolis, Minn.	IBM System/360 Model 50	Expansion of existing computer system
Radio Specialties Co., Inc., Detroit, Mich.	IBM System/360 Model 20	Providing up-to-date stock control of some 46,000 items ranging from tiny resistors to large-scale testing equipment; also billing, statements and sales analysis
Electrical Engineering Dept., University of Minnesota	Two Model 240 Simulator general purpose analog computing systems	Training students in the application of analytical and simulation techniques
Booth Fisheries, Chicago, Ill.	Honeywell 120 computer	Centralizing world-wide cost accounting and doubling speed of handling over 20,000/month invoice transactions
J. Walter Jones, Jr., and Associates, South Boston, Va.	LGP 30 Computer	Performing subdivision design and surveying computations

ORGANIZATION NEWS

NCR AND RCA SIGN PACT ON COMPUTER PATENTS

The National Cash Register Company has announced that it has entered into an agreement with the Radio Corporation of America under which each company grants rights to the other under its patents in the electronic data processing field.

This cross-licensing agreement is designed to free both companies from patent infringement conflicts which could arise in the increasingly complex computer field, NCR pointed out, but it does not involve any exhange of know-how.

The agreement covers not only patents on NCR and RCA computers, as such, but also patents pertaining to other products of the two companies when those products are employed as direct, "on-line" components of computer systems.

CONTROL DATA RECEIVES EXPORT LICENSE TO SHIP A SECOND 6600 COMPUTER TO FRANCE

William C. Norris, President of Control Data Corporation has announced that his company has received an export license to ship a super-scale Control Data 6600 Computer System to S.I.A. (Societe d'Informatique Appliquee) in Paris.

This is the second export license granted to Control Data (see Computers and Automation, January 1967, p.52) under terms of the agreement recently reached by the United States and French Governments on computer exports to France. A Control Data 6600 was installed at the French Power Bureau (Electricite de France), also in Paris, during December.

ADAMS ASSOCIATES FORMS NEW SUBSIDIARY

Charles W. Adams, President of Charles W. Adams Associates, Inc. (Cambridge, Mass.) has announced the formation of a new wholly-owned subsidiary, Adams Associates Incorporated. This move was made, according to Mr. Adams, in recognition of the need for a separate organization, with its own corporate identity, to handle the rapidly expanding computer consulting and programming activities. Its other subsidiary, KEYDATA Corporation, provides time-shared on-line data processing services for a variety of business management and accounting requirements. The new company will be headed by John T. Gilmore, Jr., who in 1959 founded the parent firm with Mr. Adams and has been its Vice President since.

GT&E SUBSIDIARY NEGOTIATING FOR ACQUISITION OF ULTRONIC SYSTEMS

General Telephone & Electronics Corporation and Ultronic Systems Corp. have announced that a GT&E subsidiary, Sylvania Electric Products Inc., has been negotiating with Ultronic Systems Corp. on a plan for acquisition of Ultronic Systems.

Under terms of the plan, GT&E would issue one share of $50 par, 5% convertible preferred stock in exchange for each five shares of Ultronic Systems common stock, except for the 178,390 common shares already owned by Sylvania. The GT&E convertible preferred stock would be convertible into common stock initially at $55 per share. Based upon the present number of Ultronic Systems common shares and warrants outstanding, approximately 317,874 shares of GT&E preferred stock would be issued. In addition, preferred shares of GT&E would be issued for outstanding preferred shares of Ultronic Systems.

GT&E President, Leslie H. Warner said that upon approval of the plan by the Board of Directors of Ultronic Systems, the plan would then be submitted to the GT&E Board of Directors for their approval. Robert S. Sinn, President and Chairman of Ultronic Systems, said that upon such approval, the proposal then would be presented to Ultronic Systems shareowners for approval on a date yet to be decided.

Sylvania is a principal producer of electronic systems and components, lighting products, and television-stereo-radio sets. Ultronic Systems (Pennsauken, N.J.) is engaged in the development, manufacture, lease, and servicing of electronic quotation systems for securities and commodities markets.

TABULATING & BUSINESS SERVICES, INC. ACQUIRES AMERICAN COMPUTING CENTERS

Tabulating & Business Services, Inc., a diversified data processing service company, has acquired American Computing Centers Corp., also of New York. TBS president Murray Lee said the merger of American Computing Centers into Tabulating & Business Services has been approved by the boards of directors of both companies.

Jack Rothschild, president, and Alan Steinberg, executive vice-president, American Computing Centers, will join TBS as vice-presidents. American Computing Centers, a four-year old data processing service center, has specialized in operations research techniques using computers in industrial applications for the solution of industrial and management problems.

COMPUTING CENTERS

MAJOR EXPANSION OF CSC'S REMOTE COMPUTING NETWORK

A major expansion of Computer Sciences Corporation's remote computing network has been announced by William R. Hoover, president of CSC's Computer Sciences Division. Under the expansion program, Hoover said, CSC will install a Univac 1108 computer at Richland, Wash., center of the remote network, this month. In addition, CSC will install 52 of Univac's recently introduced DCT-2000 data communication terminals for customers of the remote service, called Remotran.

The enlarged system will offer advanced time-sharing capabilities, and will be accessible from anywhere in the United States and Canada via standard telephone circuits. The new computer will be one of the first 1108's with a core memory of 131,000 words of 36 bits, twice the size of the 1108 models now in operation. CSC will develop its own software for the larger-capacity machine.

UNITED NUCLEAR CORPORATION OPENS COMPUTING CENTER IN WESTCHESTER COUNTY

The United Nuclear Corporation has established a Computer Center in Elmsford (near White Plains), N.Y., that makes available a large scale, scientific and business data processing facility in Westchester County.

The Center, offering expanded customer services, operates a Control Data 1604A 32K with 8 magnetic tape drives and a 1000 lpm printer. Peripheral equipment and private offices for clients are provided. The Center's full complement of software — some 350 off-the-shelf programs — are available to the public. These include commercial and scientific routines (PERT, COBOL, FORTRAN, Linear Programming) and numerous engineering and statistical codes.

The Computer Center is a part of the Development Division of the United Nuclear Corporation. United Nuclear is the largest independent company in the nuclear energy field.

WASHINGTON-AREA COMPUTER CENTER OPENED BY THE MATRIX CORPORATION

A new Washington-area computer center, establishing a new terminal in its east-coast computer network centered in New York, has been opened by the Matrix Corporation in the Westgate Research Park at McLean, Va. The new center offers to business, industry, and government the use of the high-speed IBM 7094 in the New York center, by way of the McLean installation of the UNIVAC 1004, the vital link in teleprocessing.

On order for the New York center of the east-coast network, for spring installation, is a more powerful IBM 360, model 65-40, which will greatly increase the capacity of the east-coast network. This McLean installation is being augmented also by the new IBM 360, model 40, for teleprocessing.

The east-coast network now extends from New York and Farmingdale, Long Island, to Washington. Plans for the immediate future also call for extension of this network to Boston and other cities. The company also operates a major computer network on the west coast, with its main center in the Los Angeles area. A mid-west network is in the planning stages.

EDUCATION NEWS

COMPUTER PROGRAMMING COURSE FOR THE BLIND

A special nine-month computer programming course adapted for the blind has been undertaken by System Development Corporation (SDC), Santa Monica, Calif., under an institutional tuition arrangement with the Division of Rehabilitation of the Blind, Dept. of Rehabilitation, California Health and Welfare Agency.

One blind student already has successfully completed a rigid course and, after weighing several offers, has accepted employment with a computer firm in the Los Angeles area. The same basic course, with some modifications, is being offered to the first class of 14 blind students.

Students for the SDC program were selected from top candidates and were required to make high scores on a general aptitude and intelligence test battery. Their academic backgrounds include a science interest leaning heavily toward mathematics, and most have at least two years of college.

Materials for training the blind student are not substantially different from those presently used by SDC's Professional and Technical Training staff. Texts have been transcribed on tape and tests will be given in braille. From his class notes and taped lessons, the student types his personal braille notebook. The blind student can review printouts converted to braille by a computer program, reading the raised impressions made on the back of the paper by the computer line printer.

According to SDC project head and principal instructor, Miss Connie Walker, no student will graduate from the class unless he has met the grade requirements established for SDC's courses. Blind students who fall short of a passing grade at the completion of the course will be evaluated on the basis of what they have achieved and will be certified for an appropriate level of computer operation.

ARMSTRONG ADULT EDUCATION OFFERS JOB TRAINING WITH NEW COMPUTER

As the new semester started last month, students at the Armstrong Adult Education Center, Washington, D.C., began using a new IBM 1130 computer. The desk-sized computer, designed for a number of accounting and research applications, was installed for the Center's data processing classes. The course is the first in computer training to be offered in public schools here. Classes, which are given at night, are based on a curriculum developed by the United States Office of Education.

To prepare for actual computer use, students are instructed in basic programming principles, including languages used to communicate with computers. The "hands-on" computer training will include a number of representative applications.

At present, 78 students are enrolled in the course. The data processing course is one of 18 occupational training programs offered along with a full high school academic program at Armstrong. Tuition is free to District residents.

CURRICULUM 'SHOPPING' VIA COMPUTER, TV CONSOLES

School teachers may soon be using television and computers to "shop" for classroom teaching aids. A pilot project introduced by the Portland Public School District, Portland, Ore., makes use of both devices to help teachers immediately locate lists of instructional materials, training aids and reference data to aid them in planning their curricula. The project, operating under a $129,000 grant from the Hill Family Foundation of Minneapolis, is seeking answers to two major problems of education — the keeping of voluminous pupil personnel records, and the storing, maintenance and retrieval of school curriculum.

District officials feel they are finding the answers through the use of their Honeywell 120 computer and the television display devices, which have keyboards that allow information stored in the computer to be retrieved instantly and flashed on the screen for a teacher's use. The first of the display units will be installed at Rice Elementary School and will be ready for use by early spring.

The science curriculum will be placed into the computer's memory as the initial phase of the pilot project. Then, at Rice Elementary, teachers can go curriculum 'shopping' by using the computer to help plan the best possible class programs for the their students, geared to their particular learning levels. For example, if a teacher is beginning instruction in rock classification, she determines the learning readiness of her class, keys that information and the subject title into the computer and immediately sees on the screen a list of all instructional materials and aids that will be beneficial to her students. If she desires a printed list, pressing another button activates a teletypewriter which will give her a permanent copy.

Educators have long been aware that teachers work under severe handicaps in curriculum planning, since necessary information often is not properly indexed for reference or is stored someplace not readily accessible. "Through our program, vast amounts of information, centrally stored in the computer, will be available to teachers quickly and completely," said Thomas Crowder, project coordinator.

NEW PRODUCTS

Digital

SEL REDESIGNS 810A COMPUTER

The SEL 810A computer, manufactured by Systems Engineering Laboratories, Inc., Fort Lauderdale, Fla., has been redesigned, reduced appreciably in size, given increased reliability, and reduced in price, the company announces. The redesigned system is identical logically to the previous one — hence, the retention of the name SEL 810A.

The computer remains a total integrated circuit machine. Careful redesign has reduced the 378 logic cards previously employed to a startling 124 cards. Wiring connections have been reduced from 22,000 to 4000.

The 16 bit parallel computer has an internal memory cycle time of 1.75 μseconds, and a core storage of 4096 words which can be expanded to 32,000 words. The new 810A will use the field proven set of software developed over the last two years for the earlier generation of 810A's. Delivery of the first unit is scheduled for March with 30 to 60 day delivery thereafter.
(For more information, designate #43 on the Readers Service Card.)

INTERDATA MODEL 3 COMPUTER

Interdata, of Farmingdale, N.J., a 6 month old firm in the digital computer market, has disclosed the development of a low cost, high speed digital computer. The machine, Interdata Model 3, has been designed for the control field. It has definite applications for process control, real time counting, sequence control and data acquisition.

The same device with a different display panel can be used in the education field for teaching computer usage, mathematics, physics,

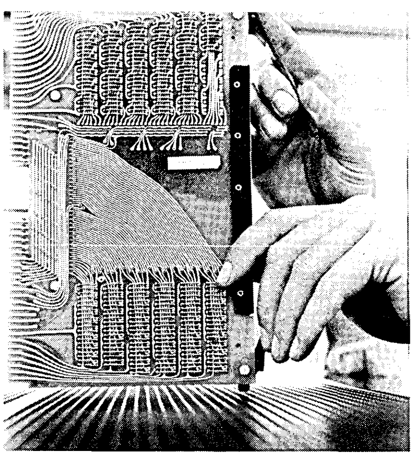
logic and programming. Model 3 for control is a 16 bit unit selling for $6000, while the education unit sells for $6700.
(For more information, designate #42 on the Readers Service Card.)

SIGMA 5 COMPUTER ANNOUNCED BY SDS

Sigma 5, a new medium-priced, multi-use computer, was announced last month by Max Palevsky, President of Scientific Data Systems, Santa Monica, Calif. It is the middle member of the new Sigma line of low-cost, multi-use computers announced in 1966.

The Sigma 5 is fully compatible with the larger Sigma 7, and data and input/output compatible with Sigma 2. The new Sigma 5 has been designed especially for scientific and business applications which require a high throughput for general purpose applications and real time systems control.

Memory cycle time of the Sigma 5 is 850 nanoseconds, which is reduced to 600 nanoseconds when overlapping of memory units occurs. Memory size ranges from 4096 words



— Memory module: each module contains 4096 eight bit bytes. Modular concept allows Sigma 5 computers to be field-expanded to 131,072 words

of 32 bits to 131,072 words. Like the larger Sigma 7, all of memory is addressed directly or indirectly and can be altered in bytes, half-words, words or double words.

Up to eight input/output processors, each with capacity for 32 I/O channels, can be provided with the Sigma 5. The computer is delivered standard with eight input/output channels and 16 general purpose registers, which can be expanded to as many as 256 registers. Up to 224 priority interrupt levels are available with Sigma 5.

Software supplied with Sigma 5 includes two operating monitors, three levels of FORTRAN IV, two assemblers, SDS COBOL-65 and other business programs, and a complete set of utility and diagnostic programs.
(For more information, designate #41 on the Readers Service Card.)

Software

COMPUTER PROGRAMMING TECHNIQUE USED TO PLAN FLEET OPERATIONS

Steamships sailing on the Great Lakes are following shipping routes and hauling cargoes that were pre-determined by a computer long before the shipping season began. Using a linear programming computer technique developed by The Service Bureau Corporation (SBC), New York, N.Y., Great Lakes shipping executives are getting answers to such key questions as how many ships will be needed for the season, when marginal ships should be activated, what cargoes to haul and sailing patterns to follow.

For every ship in the fleet, its capacity and shipping time between ports will vary according to cargo, distance, port facilities, water levels and other factors. During the season, which extends from April to December, a vessel will load at one port and sail to any number of others. The number of possible routes that could be taken by any one ship, said Dr. Leon Gleiberman, who helped develop the mathematical model at SBC's Scientific Computing Center, is "astronomical, and probably over a billion".

Moreover, with the programming technique, experiments can be made on the possible effects of unexpected occurrences such as labor strikes, accidents or unusual weather. The results of purchasing new ships, selling older ones and signing new tonnage contracts also can be analyzed.
(For more information, designate #44 on the Readers Service Card.)

Data Transmitters and A/D Converters

TELLER REGISTER MACHINE FROM HONEYWELL

Honeywell's electronic data processing division (Wellesley Hills, Mass.) has added a new Teller Register machine to its computer product line. Benjamin W. Taunton, in charge of Honeywell's computer marketing effort for banking and finance, described the machine as the most advanced bank teller computer equipment on the market because "it freely exchanges information with a central computer without the restrictions found in other equipment of this type."

Taunton said the new desk-top register is "limited only by the imagination of the bank using it," because the functions the machine performs are completely controlled by a flexible teller-unit monitoring computer program which may be modified easily by the bank to meet its particular requirements. He stated that initially the register, which has a 9-by-9 numeric keyboard and 12 control key indicators, will be used largely to process savings deposits or withdrawals, issue checks or money orders and record mortgage payments, but he added the machine can perform many other teller functions for banks.

If communications are cut off between the computer and the register, the machine can caluclate information indefinitely on an "off-line" basis for later entry into the computer. Taunton stated that the control unit, which operates the Teller Register, handles up to 10 registers at a time. He said prices are competitive with other computer equipment of this type.
(For more information, designate #45 on the Readers Service Card.)

Numerical Control

NUMERICAL CONTROL TEACHING EQUIPMENT

Equipment for teaching numerical control techniques is claimed by its British manufacturer to be the first simple, low-cost system suitable for technical colleges and universities.

The firm, Feedback Ltd. of Sussex, England, has combined its Logikit Primer LK 225 logic tutor with additional plug-in elements and its Digital Encoder SE 254 to produce an apparatus that illustrates the principles of numerical control; it can be assembled and understood by an average student during a single laboratory period.

In its simplest form the equipment is used by the student to construct a position control system which responds to a numerical demand in the form of a 3-bit binary number set on three switches. The digital encoder is coupled to the output shaft, and this produces a Gray Code to identify any of eight equal angular segments within a 360^0 rotation.

The logic circuits made up by the student from simple plug-in elements convert the Gray Code to natural binary and compare input demand with encoder output. The differences (greater or less) in binary form are converted into an error signal suitable for the servo. This drives the encoder in the correct direction to reduce the difference to zero.

Input demand and the natural binary response are monitored continuously by a bank of indicator lamps so the student can see the accuracy and speed of the system's response. The logic tutor, additional plug-in elements and digital encoder are available from the U.S. agent, Muirhead Instruments Inc., and cost $885 FOB Mountainside, N.J. The equipment can be expanded for larger experiments at extra cost.
(For more information, designate #46 on the Readers Service Card.)

Input-Output

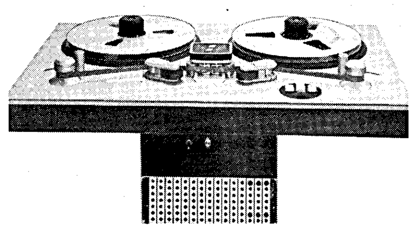
FACIT TAPE READER MARKETED BY POTTER

Potter Instrument Company, Plainview, N.Y., now is marketing Facit's PE 1000 Tape Reader under a marketing agreement concluded between the two companies. A recent demonstration showed a PE 1000 Tape Reader connected to an IBM 1460 computer. In this mode it is possible to read in punched tape directly without any conversion to punched cards.

The PE 1000 is used for reading punched tapes into data processing systems, data processing control of automatic processes, data transmission, as well as duplicating punched tapes. It is convertible for 5, 6, 7 or 8 tracks, and features a radical new design which eliminates risk of reading mistakes. All tape colors can be read, even transparent tape. The tape reader is unaffected by dust, dirt and incident light.
(For more information, designate #47 on the Readers Service Card.)

G-12 TAPE DUPLICATING SYSTEM HAS SPEEDS OF 240 IPS

The G-12 Tape Duplicating System, with all solid-state plug-in modules, has duplication ratios of 32:1 and 16:1 with recording and duplicating speeds of 120 and 240 ips. The exclusive "Focused Gap Field" magnetic recording technique, which is a beamed RF bias, improves the quality of all tapes because it very closely matches the ideal anhysteretic magnetic recording process. The equipment is produced by Gauss Electrophysics, Inc., Santa Monica, Calif.



The G-12, because of modular design, can be matched for any combination of up to 20 duplicating slave recorders and can be patched in or out of the common bus without system readjustment. Two, four or eight track magnetic heads may be plugged in quickly to handle a wide variety of program formats.

Several configurations are available from 2 to 8 track ¼-inch and the new 0.150 in. (3.8 MM) tapes. Rewind fast forward speed of a 2400 foot reel is 80 seconds on NAB metal or plastic reel or open hubs.
(For more information, designate #48 on the Readers Service Card.)

CALCOMP TELEPLOTTER

The CalComp Teleplotter, introduced by California Computer Products, Inc., Anaheim, Calif., extends high-speed digital plotting to users of time-shared computer systems. The Teleplotter, composed of a CalComp plotter and controller, operates with a standard Data-Phone Adapter and teletypewriter, allowing two-way communication with the computer and automatic switching, under program control, between plotter and teletypewriter. The Teleplotter drives any 500 or 600 Series CalComp plotter at speeds up to 280 incremental steps per second.

The user dials the computer for service and inputs all instructions, data and plot programs through the teletypewriter. In response, the computer prepares the plot program, including identification codes for that particular plotter, and outputs signals for remote online plotting.

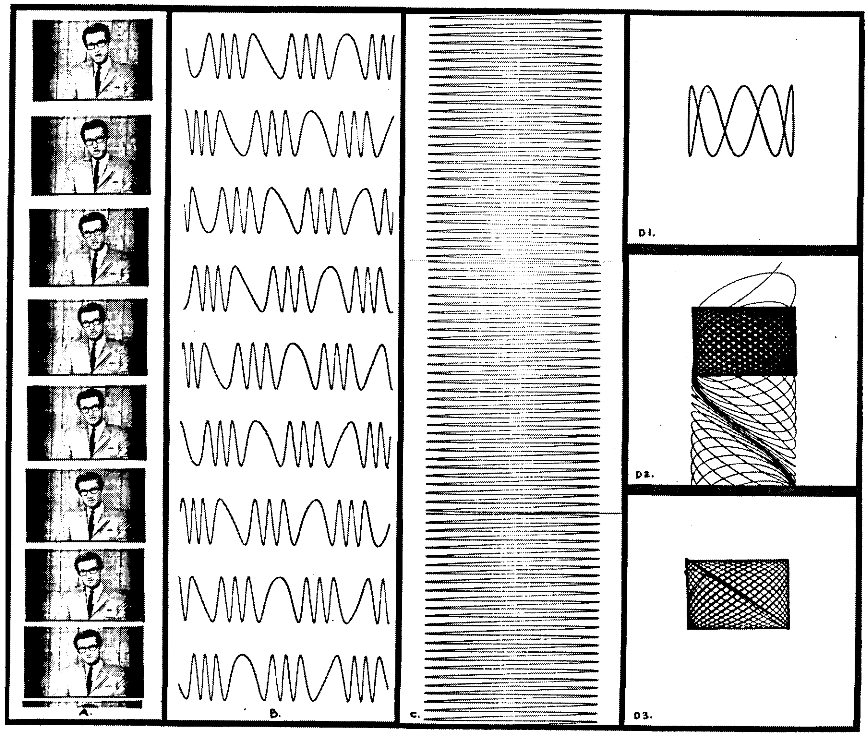
The Teleplotter decodes an 8-bit character which contains instruction codes and plotter drive signals. These contain all necessary data for direction of movement, plotter select, teletypewriter select, pen up or pen down, and incremental plotter steps to be performed (up to a maximum of 28 per character).

While plotting and other remote station output functions normally proceed automatically under computer control, a manual interrupt capability in the Teleplotter permits the user to override and interrupt computer controlled operations when required.
(For more information, designate #49 on the Readers Service Card.)

HONEYWELL OSCILLOGRAPH

Honeywell Inc. has introduced an oscillograph instrument that it says will measure and record high-frequency analog data at up to one million cycles per second, and has a data printout capability nearly 100 times faster than any direct-write system on the market, with a writing speed measured at more than a million inches per second. The new oscillograph — designated as the Model 1806 fiber-optic CRT (cathode-ray tube) Visicorder — was developed and built by the firm's Test Instruments Division in Denver, Colo. and made its debut in the instrumentation market last month.

A unique transverse recording technique permits writing across the paper, as well as in the standard downward mode. The 4-axis capability of Honeywell's 1806 oscillograph is indicated by the collection of analog and video signals recorded by the instrument as shown in the picture. Its capabilities include recording video pictures (A) in a series of individual frames; transverse (B) or longitudinal (C) recording of signals in a continuous mode; and combining both X and Y signals to obtain single 3x5-inch records (D1) or continuous records of X-Y plots and Lissajous patterns (D2 and D3).

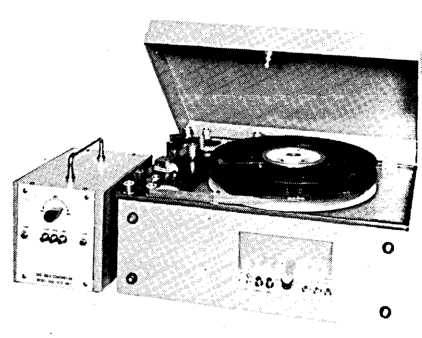


The compact and versatile Model 1806 Visicorder is said by the automation systems firm to have "virtually unlimited" uses. The high-frequency analog and video recorder incorporates operational features of both oscillographs and oscilloscopes to provide the user with immediate and permanent data records. Conventional oscilloscope controls were used for easy setup and operation of the 1806 for standard laboratory use.
(For more information, designate #50 on the Readers Service Card.)

Components

INCREMENTAL RECORDER TEST UNIT

Digi-Data Corporation's (Bladensburg, Md.) new Model 1400 Test Unit is capable of exercising incremental recorders by supplying all external command functions including Step and Record and IR Gap signals and monitoring all recorder outputs such as End IR Gap, End Tape, Echo Check Correct, and Echo Check Incorrect Output signals. Programming capabilities include a wide variation in recording rates, a choice of record lengths, odd or even parity, and the type of repetitive pattern recorded on tape. Powered by the recorder, the unit is but 8-3/8"L x 6"W x 6"H and sells for $350 including mating cable.



(For more information, designate #51 on the Readers Service Card.)

AUTOMATION

AUTOMATED CATTLE FEEDING

A completely automated cattle feeding system developed at the University of Illinois agricultural Experiment Station may revolutionize the feeding of dairy cows. A joint project of the Departments of Agricultural Engineering and Dairy Science, the system now being tested is based on the concepts of group handling of cows for milking, feeding and housing, and bulk handling of feed and milk. It is designed to eliminate the labor and equipment problems usually associated with mechanized feeding of individual dairy cows.

From the feed processing center — the heart of the system — grain-forage rations are processed automatically and delivered to four separate feed bunks, each accommodating groups of 20 cows. The feed processing center includes three silos or upright storages, a small feed factory with an automatic hammer mill, and a feed processing building housing a specially designed control panel manufactured by General Electric's General Purpose Control Dept. The control panel enables one man to select and regulate the proportion of each of the feed ingredients.

A number of safety features incorporated into the system prevent delivery of an improperly constituted ration, or overloading of conveyor motors. This feeding system removes the various ingredients from storage in the proper quantity and at the proper rate, mixes them together, and finally feeds them to each of four lots. The make-up of the ration and the quantity discharged are regulated by the control system. Frequency of feeding is controlled by a 24-hour time clock. In the event of equipment failure, the system automatically shuts off and a warning device is actuated.

Rations are collected from the four sources by means of a common 9-inch auger. The auger serves as both a conveyor and a mixer, discharging the ingredients into an inclined chain flight elevator that raises them approximately 10 feet above the ground to the first feed-bunk distributor. The feed is diverted from the inclined elevator into a 90-foot cross-conveying auger. There is a clutch between the first 10-foot section and the remainder of the auger. If feed is desired in the first feeder, the auger turns one way, operating only the first 10-foot section and dropping the feed into the first feeder. If feed is required at the other mechanical feeders, the auger motor is reversed and the whole 90-foot section turns, vonveying feed to the second feeder.

By adapting this automated system to the group handling concept, it is anticipated that feeding in the milking parlor can be eliminated and that one man will be able to manage a herd of high-producing dairy cows producing 500,000 to 700,000 pounds of milk annually.

BUSINESS NEWS

HONEYWELL EDP HAS RECORD REVENUES

Honeywell's computer business achieved record revenues of $182 million in sales and rental income during 1966, according to James H. Binger, board chairman of Honeywell Inc. This is an increase of some 79% from revenues of $104 recorded in 1965. Honeywell's computer business now represents 20% of Honeywell Inc.'s total income. It is the fastest growing segment of the firm.

Binger also stated that Honeywell passed a major milestone in 1966 when its domestic computer business became profitable for the first time. "The achievement was especially meaningful because it came at a time of rapid growth requiring continued investment of large sums in our computer business," Binger noted.

Long-term leasing and the sale and leaseback of computers have contributed significantly to financthe company's growth, according to the Honeywell chairman. He said that during 1966 Honeywell sold and leased back computers valued at $40 million. The result of these transactions were to lower by a substantial amount Honeywell's capital expenditures to finance computer equipment on rental to customers. Honeywell's total capital expenditures in 1966 amounted to $84 million.

Binger reported that the value of computers shipped in 1966 rose sharply to approximately $300 million, compared with $205 million in 1965. "We expect 1967 sales and rental revenue to exceed the 1966 level," Binger said. He added that at year end there were over 2000 Honeywell computers installed or on order around the world.

AMPEX SALES RISE

Ampex Corp. had record sales of $101,265,000 for the six months ending October 29th. This is an increase of 37% from sales of $73,715,000 reported for the same period last year. Earnings also rose to $4,453,000 or 30% from $3,437,000 for the period a year ago.

COMPUTER APPLICATIONS UPS SALES, EARNINGS

Computer Applications, Inc., New York, one of the largest software and service bureau firms in the country, had sales of $17,576,119 for the fiscal year ended Sept. 30. This is an increase of 59% from sales of just over $11 million reported in the last fiscal year.

Earnings were up 112% from $243,545 in FY65 to $517,352 in the fiscal year just completed.

Computer Applications has relied on mergers and acquisitions for a substantial part of its business growth.

UNIV. COMPUTING DOUBLES EARNINGS

The earnings of University Computer Co., Dallas, Texas, doubled in the nine months ended Sept. 30, from $186,532 (30¢ per share) in the first nine months of fiscal year '65 to $608,294 (66¢ per share) for the current period.

Sales increased 210% during the same period, from $1,483,843 to $3,118,808, in comparable nine month periods.

University Computing Co. is a diversified computer service firm offering software marketing, service bureau, and computer leasing services.

WORLD REPORT — GREAT BRITAIN

Although the Polish deal of International Computers and Tabulators is not going as smoothly as was last month expected — only one out of the four machines has been signed for, as the Poles are trying for lower prices than ICT will give — the success story of the United Kingdom in Eastern Europe continues.

Now Elliott-Automation has revealed four contracts for a number of "Iron Curtain" countries, worth about $1.5m. Significant among these is one for a small process-control machine for the Soviet Union. Elliott's has a good name in Russia, having supplied the equipment to control the world's largest ethylene cracker there.

Also significant is the contract to supply a large system to the Czechoslovak central computing authority, to be used by it in evaluating systems offered from outside the country for use by Czech organizations.

At a time when the United States State Department is reported to have relaxed its opposition to the sale of System-4 computers to Eastern bloc countries, this development could be of particular importance, since it could involve evaluation of 3rd generation machines by computer equipment which, though well ahead in the cost-effectiveness stakes, is barely "2½ generation". System-4 is, of course, Spectra 70 plus a further degree of microminiaturisation.

The makers of System-4, English Electric, are being very cautious about this apparent U.S. leniency in translating COCOM strategic export rules. They will barely admit to a slight thaw. At the same time it would surprise no one in the UK computer world to hear of several System-4 sales in the near future to the Czechs and possibly the Hungarians. At the same time, English Electric now admits that it is highly unlikely that the KDF-9 computer it had contracted to send to Peking will find its way there. This is a machine which is comparable in power with the 7094. It is a moot point whether there has been some horse-trading.

The Ferranti company, which many regard as leaders of British electronic technology, have sprung a surprise on the computer community. A year ago they announced two microminiature process control computers, the Argus 400 and 500; now to go with them they are building a whole range of auxiliary equipment whose circuits are based entirely on the company's integrated circuitry.

The company has demonstrated the degree to which its equipment can be made more compact; as a result, but more important still, a major client has ordered 600 of the micromin display screens for its seat reservation scheme, destined to be the most advanced in the world. The client is British Overseas Airways Corporation and the displays will be linked through some 50 Argus computers to two IBM 67's.

Ferranti pioneered the Direct Digital Control concept in 1962 with an Argus system working the Imperial Chemical Industries fertilizer plant at Fleetwood. Its latest range of equipment is aimed primarily at the industrial control field and while the two small computers are the central units, the company has also designed about 100 modules to permit the easy construction of input/output systems to suit any application.

High reliability and low cost have been sought by using virtually nothing but integrated-circuit logical elements and replacing conventional wiring looms with printed wiring systems. A standard interface highway is not modified by the number or types of input/output modules, which are simply connected to the highway. Module cards and buffer cards which use multilayer boards are plugged into main bore units. Plant connections are made directly to the module sockets.

Analogue input equipment can be built into a system able to handle up to 512 input signals and there is an analogue/digital converter capable of some 80,000 conversions a second.

Analogue output control of up to 256 integrator output circuits is possible. Pulsed digital input unit allows the computer to check and determine the state of 3072 contact closures on a plant. Output signals to drive external units are controlled by solid-state or reed switches and one output unit will give a total of 128 points controlled.

This is Ferranti's answer to the explosion in process control and in the year ending in November it sold $14m worth of the new equipment. The company's position as a components manufacturer may be strengthened very shortly by the injection of $12m to $15m of government money through the State-controlled investment organisation known as the National Research Development Corporation, which is committed to establishing a sound, competitive microcircuit industry in Great Britain.

Ted Schoeters
Stanmore
Middlesex
England

CALENDAR OF COMING EVENTS

(Continued from page 29)

May 23-26, 1967: GUIDE International, Americana Hotel, New York, N.Y.; contact Lois E. Mecham, Secretary, GUIDE International, c/o United Services Automobile Assoc., 4119 Broadway, San Antonio, Texas 78215

June 20-23, 1967: DPMA International Data Processing Conference and Business Exposition, Sheraton-Boston Hotel, Boston, Mass.; contact William J. Horne, Conference Director, United Shoe Machinery Corp., 140 Federal St., Boston, Mass.

June 26-27, 1967: Computer Personnel Research Group Fifth Annual Conference, University of Maryland, College Park, Md. (near Washington, D.C.); contact Dr. Charles D. Lothridge, General Electric Co., 570 Lexington Ave., New York, N.Y. 10022

June 28-30, 1967: 1967 Joint Automatic Control Conference, University of Pennsylvania, Philadelphia, Pa.; contact Lewis Winner, 152 W. 42nd St., New York, N.Y. 10036

July 31-August 4, 1967: MEDAC '67 Symposium and Exhibition, San Francisco Hilton Hotel, San Francisco, Calif.; contact John J. Post, Executive Secretary, AAMI, P. O. Box 314, Harvard Square, Cambridge, Mass. 02138

August 22-25, 1967: WESCON (Western Electronic Show and Convention), Cow Palace, San Francisco, Calif.; contact Don Larson, 3600 Wilshire Blvd., Los Angeles, Calif. 90005

Aug. 28-Sept. 2, 1967: AICA (International Association for Analogue Computation) Fifth Congress, Lausanne, Switzerland; contact secretary of the Swiss Federation of Automatic Control, Wasserwerkstrasse 53, Zurich, Switzerland

Aug. 29-31, 1967: 1967 ACM (Association for Computing Machinery) National Conference, Twentieth Anniversary, Sheraton Park Hotel, Washington, D.C.; contact Thomas Willette, P.O. Box 6, Annandale, Va. 22003

Sept. 6-8, 1967: First Annual IEEE Computer Conference, Edgewater Beach Hotel, Chicago, Ill.; contact Professor S. S. Yau, Dept. of Electrical Engineering, The Technological Institute, Northwestern University, Evanston, Ill. 60201

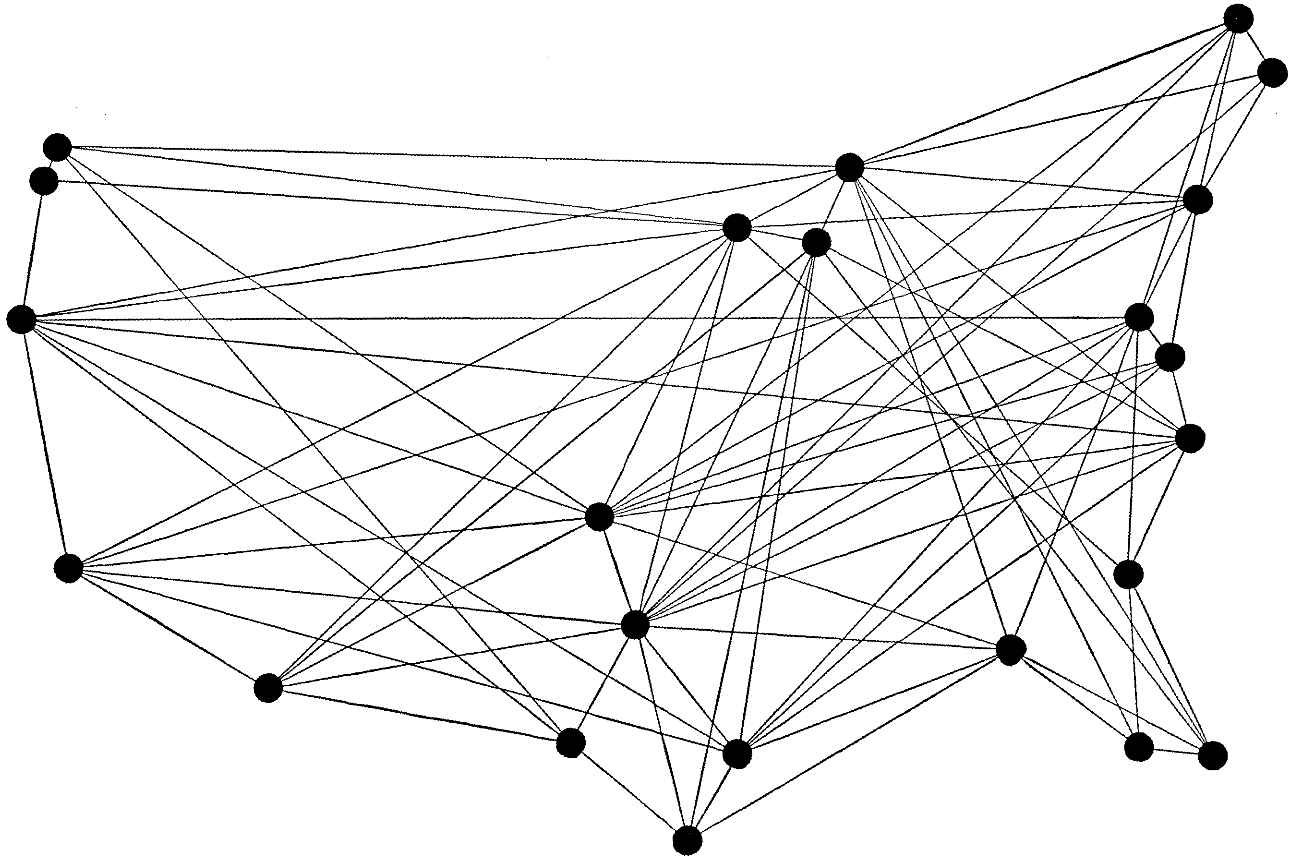
Sept. 11-15, 1967: 1967 International Symposium on Information Theory, Athens, Greece; contact A. V. Balakrishnan, Dept. of Engineering, U.C.L.A., Los Angeles, Calif. 90024

Sept. 25-28, 1967: International Symposium on Automation of Population Register Systems, Jerusalem, Israel; contact D. Chevion, Chairman of Council, Information Processing Association of Israel, P.O.B. 3009, Jerusalem, Israel

Nov. 14-16, 1967: Fall Joint Computer Conference, Anaheim Convention Center, Anaheim, Calif.; contact AFIPS Headquarters, 211 E. 43rd St., New York, N.Y. 10017

May 21-23, 1968: Spring Joint Computer Conference, Sheraton Park/Shoreham Hotel, Washington, D.C.; contact AFIPS Headquarters, 211 E. 43rd St., New York, N.Y. 10017

Aug. 5-10, 1968: IFIP (International Federation for Information Processing) Congress 68, Edinburgh, Scotland; contact John Fowlers & Partners, Ltd., Grand Buildings, Trafalgar Square, London, W.C. 2., England



Your decentralized company

MONTHLY COMPUTER CENSUS

The number of electronic computers installed or in production at any one time has been increasing at a bewildering pace in the past several years. New vendors have come into the computer market, and familiar machines have gone out of production. Some new machines have been received with open arms by users — others have been given the cold shoulder.

To aid our readers in keeping up with this mushrooming activity, the editors of COMPUTERS AND AUTOMATION present this monthly report on the number of general purpose electronic computers of American-based companies which are installed or on order as of the preceding month. These figures included installations and orders outside the United States. We update this computer census monthly, so that it will serve as a "box-score" of progress for readers interested in following the growth of the American computer industry, and of the computing power it builds.

In general, manufacturers in the computer field do not officially release installation and on order figures. The figures in this census are developed through a continuing market survey conducted by associates of our magazine. This market research program develops and maintains a data bank describing current computer installations in the United States. A similar program is conducted for overseas installations.

Any additions, or corrections, from informed readers will be welcomed.

AS OF JANUARY 10, 1967

NAME OF MANUFACTURER	NAME OF COMPUTER	SOLID STATE?	AVERAGE MONTHLY RENTAL	DATE OF FIRST INSTALLATION	NUMBER OF INSTALLATIONS	NUMBER OF UNFILLED ORDERS
ASI Computer	ASI 210	Y	$3850	4/62	25	0
	ASI 2100	Y	$4200	12/63	7	0
	ADVANCE 6020	Y	$4400	4/65	13	5
	ADVANCE 6040	Y	$5600	7/65	8	6
	ADVANCE 6050	Y	$9000	2/66	6	6
	ADVANCE 6070	Y	$15,000	10/65	5	6
	ADVANCE 6130	Y	$1000	11/66	3	21
Autonetics	RECOMP II	Y	$2495	11/58	35	X
	RECOMP III	Y	$1495	6/61	7	X
Bunker-Ramo Corp.	BR-130	Y	$2000	10/61	160	2
	BR-133	Y	$2400	5/64	29	40
	BR-230	Y	$2680	8/63	15	X
	BR-300	Y	$3000	3/59	35	X
	BR-330	Y	$4000	12/60	30	X
	BR-340	Y	$7000	12/63	20	X
Burroughs	205	N	$4600	1/54	42	X
	220	N	$14,000	10/58	34	X
	E101-103	N	$875	1/56	122	X
	B100	Y	$2800	8/64	172	12
	B250	Y	$4200	11/61	84	1
	B260	Y	$3750	11/62	231	2
	B270	Y	$7000	7/62	165	3
	B280	Y	$6500	7/62	128	5
	B300	Y	$10,000	7/65	135	78
	B2500	Y	$5000	1/67	0	45
	B3500	Y	$14,000	5/67	0	30
	B5500	Y	$22,000	3/63	61	11
	B6500	Y	$33,000	2/68	0	11
	B8500	Y	$200,000	2/67	0	2
Control Data Corporation	G-15	N	$1600	7/55	295	X
	G-20	Y	$15,500	4/61	25	X
	LGP-21	Y	$725	12/62	160	X
	LGP-30	semi	$1300	9/56	136	X
	RPC-4000	Y	$1875	1/61	65	X
	160*/160A/160G	Y	$2100/$5000/$12,000	5/60;7/61;3/64	461	2
	924/924A	Y	$11,000	8/61	29	X
	1604/1604A	Y	$45,000	1/60	59	X
	1700	Y	$4000	5/66	23	105
	3100	Y	$11,000	12/64	85	36
	3200	Y	$14,000	5/64	66	X
	3300	Y	$15,000	9/65	50	52
	3400	Y	$25,000	11/64	19	X
	3500	Y	$30,000	9/67	0	7
	3600	Y	$58,000	6/63	45	X
	3800	Y	$60,000	2/66	15	13
	6400	Y	$50,000	5/66	12	20
	6600	Y	$85,000	8/64	21	18
	6800	Y	$130,000	4/67	0	4
Data Machines, Inc.	620	Y	$900	11/65	38	25
Digital Equipment Corp.	PDP-1	Y	$3400	11/60	60	X
	PDP-4	Y	$1700	8/62	57	X
	PDP-5	Y	$900	9/63	116	X
	PDP-6	Y	$10,000	10/64	23	2
	PDP-7	Y	$1300	11/64	110	30
	PDP-8; 8/S	Y	$525; $300	4/65	610	520
	PDP-9	Y	$1000	12/66	1	60
	PDP-10	Y	$9000	7/67	0	7
El-tronics, Inc.	ALWAC IIIE	N	$1820	2/54	14	X
Electronic Associates, Inc.	8400	Y	$12,000	6/65	13	9
General Electric	115	Y	$1800	12/65	260	540
	205	Y	$2900	6/64	44	X
	210	Y	$16,000	7/59	48	X
	215	Y	$6000	9/63	54	X
	225	Y	$8000	4/61	203	X
	235	Y	$10,900	4/64	70	2
	415	Y	$9600	5/64	205	55
	425	Y	$18,000	6/64	85	44
	435	Y	$25,000	9/65	32	17
	625	Y	$50,000	4/65	20	16
	635	Y	$56,000	5/65	18	17
	645	Y	$90,000	7/66	2	10
Honeywell	DDP-24	Y	$2500	5/63	88	3
	DDP-116	Y	$900	4/65	148	30
	DDP-124	Y	$2050	3/66	27	38
	DDP-224	Y	$3300	3/65	50	9
	DDP-516	Y	$700	9/66	10	90
	H-120	Y	$3900	1/66	360	300
	H-200	Y	$8400	3/64	1020	150

NAME OF MANUFACTURER	NAME OF COMPUTER	SOLID STATE?	AVERAGE MONTHLY RENTAL	DATE OF FIRST INSTALLATION	NUMBER OF INSTALLATIONS	NUMBER OF UNFILLED ORDERS
Honeywell (cont'd)	H-400	Y	$8500	12/61	115	X
	H-800	Y	$28,000	12/60	89	1
	H-1200	Y	$8000	2/66	68	100
	H-1400	Y	$14,000	1/64	12	X
	H-1800	Y	$42,000	1/64	21	1
	H-2200	Y	$12,000	1/66	22	64
	H-4200	Y	$20,500	3/67	0	10
	H-8200	Y	$35,000	3/68	0	3
	DATA-matic 1000	N	$40,000	12/57	2	X
IBM	305	N	$3600	12/57	138	X
	360/20	Y	$2000	12/65	1300	6400
	360/30	Y	$7500	5/65	2750	4450
	360/40	Y	$15,000	4/65	1450	1500
	360/44	Y	$10,000	7/66	24	150
	360/50	Y	$26,000	8/65	150	590
	360/62	Y	$55,000	11/65	1	X
	360/65	Y	$50,000	11/65	32	220
	360/67	Y	$75,000	10/66	6	56
	360/75	Y	$78,000	2/66	17	35
	360/90 Series	Y	$140,000	6/67	0	10
	650	N	$4800	11/54	170	X
	1130	Y	$1200	11/65	1000	4500
	1401	Y	$6600	9/60	7650	X
	1401-G	Y	$2300	5/64	1620	X
	1410	Y	$14,200	11/61	805	65
	1440	Y	$4800	4/63	3400	90
	1460	Y	$11,500	10/63	1760	X
	1620 I, II	Y	$4000	9/60	1670	20
	1800	Y	$7600	1/66	120	320
	701	N	$5000	4/53	1	X
	7010	Y	$22,600	10/63	215	6
	702	N	$6900	2/55	6	X
	7030	Y	$160,000	5/61	6	X
	704	N	$32,000	12/55	31	X
	7040	Y	$22,000	6/63	120	4
	7044	Y	$32,000	6/63	127	5
	705	N	$38,000	11/55	52	X
	7070, 2, 4	Y	$27,000	3/60	322	X
	7080	Y	$55,000	8/61	85	X
	709	N	$40,000	8/58	9	X
	7090	Y	$63,500	11/59	45	X
	7094	Y	$72,500	9/62	115	2
	7094 II	Y	$78,500	4/64	130	4
National Cash Register Co.	NCR - 304	Y	$14,000	1/60	25	X
	NCR - 310	Y	$2500	5/61	10	X
	NCR - 315	Y	$8500	5/62	390	145
	NCR - 315-RMC	Y	$12,000	9/65	36	43
	NCR - 390	Y	$1850	5/61	725	25
	NCR - 500	Y	$1500	10/65	900	850
Philco	1000	Y	$7010	6/63	16	X
	2000-210, 211	Y	$40,000	10/58	16	X
	2000-212	Y	$52,000	1/63	12	X
Radio Corporation of America	RCA 301	Y	$7000	2/61	644	2
	RCA 3301	Y	$17,000	7/64	69	5
	RCA 501	Y	$14,000	6/59	96	X
	RCA 601	Y	$35,000	11/62	5	X
	Spectra 70/15	Y	$4100	9/65	78	100
	Spectra 70/25	Y	$6700	9/65	45	55
	Spectra 70/35	Y	$10,400	7/66	18	100
	Spectra 70/45	Y	$17,400	11/65	24	95
	Spectra 70/55	Y	$40,500	1/67	0	12
Raytheon	250	Y	$1200	12/60	175	X
	440	Y	$3500	3/64	16	3
	520	Y	$3200	10/65	22	6
Scientific Control Corporation	650	Y	$500	5/66	3	7
	655	Y	$1800	10/66	0	2
	660	Y	$2000	10/65	6	2
	670	Y	$2600	5/66	1	2
Scientific Data Systems Inc.	SDS-92	Y	$1500	4/65	73	20
	SDS-910	Y	$2000	8/62	188	6
	SDS-920	Y	$2900	9/62	140	8
	SDS-925	Y	$3000	12/64	32	8
	SDS-930	Y	$3400	6/64	137	12
	SDS-940	Y	$10,000	4/66	9	9
	SDS-9300	Y	$7000	11/64	33	5
	Sigma 2	Y	$1000	12/66	2	190
	Sigma 7	Y	$12,000	12/66	2	25
Systems Engineering Labs	SEL-810/810A	Y	$1000	9/65	30	8
	SEL-840/840A	Y	$1400	11/65	4	8
UNIVAC	I & II	N	$25,000	3/51 & 11/57	24	X
	III	Y	$20,000	8/62	75	X
	File Computers	N	$15,000	8/56	16	X
	Solid-State 80 I, II, 90 I, II & Step	Y	$8000	8/58	238	X
	418	Y	$11,000	6/63	100	35
	490 Series	Y	$35,000	12/61	115	52
	1004	Y	$1900	2/63	3200	50
	1005	Y	$2400	4/66	490	200
	1050	Y	$8000	9/63	295	35
	1100 Series (except 1107)	N	$35,000	12/50	10	X
	1107	Y	$55,000	10/62	35	X
	1108	Y	$65,000	9/65	34	68
	9200	Y	$1500	6/67	0	800
	9300	Y	$3400	6/67	0	250
	LARC	Y	$135,000	5/60	2	X
				TOTALS	41,196	24,366

X = no longer in production.

* To avoid double counting, note that the Control Data 160 serves as the central processor of the NCR 310. Also, customers ordering a new computer model intended to replace a computer model in the same product line may continue to use their current peripheral equipment, which can account for 30-70% of the value of the total computer system.

NEW PATENTS

Raymond R. Skolnick
Patent Manager
Ford Instrument Co.
Div. of Sperry Rand Corp.
Long Island City, N.Y. 11101

The following is a compilation of patents pertaining to computers and associated equipment from the "Official Gazette of the U. S. Patent Office," dates of issue as indicated. Each entry consists of: patent number / inventor(s) / assignee / invention. Printed copies of patents may be obtained from the U. S. Commissioner of Patents, Washington, D. C. 20231, at a cost of 50 cents each.

November 8, 1966 (Continued)

3,284,644 / Keith Henry Dormer, Harrisburg, and Charles Thomas Wyrick, Camp Hill, Pa. / assignors to AMP Inc. / Driver Circuit For Magnetic Core Device.

3,284,772 / Robert M. Stewart, Los Angeles, Calif. / assignor to Space-General Corp. / Data Correlation Apparatus Employing Cathode-Ray Tube Input and Variable Resistance Data Storage and Comparison.

3,284,773 / Joseph John Saykay, Sea Cliff, N.Y. / assignor to Fairchild Camera and Instrument Corp. / Magnetic Coding Apparatus.

3,284,774 / Eugene Leonard, Sands Point, Edward M. Richards, East Northport, Miles Skrivanek, Jr., Glenwood Landing, Edgar Wolf, Floral Park, and Marvin Shapiro, Huntington, N.Y. / assignors to Digitronics Corp. / Information Transfer System.

3,284,775 / Ralph J. Koerner, Canoga Park, and Samuel Nissim, Pacific Palisades, Calif. / assignors, by mesne assignments to the Bunker-Ramo Corp. / Content Addressable Memory.

3,284,780 / Genung L. Clapper, Vestal, N.Y. / assignor to International Business Machines Corp. / Adaptive Logic System.

3,284,781 / Shigeru Takahashi, Tokyo, Japan / assignor to Hitachi Ltd., Tokyo, Japan / Semi-Permanent Memory Device.

3,284,782 / Joseph R. Burns, Trenton, N.J. / assignor to RCA / Memory Storage System.

November 15, 1966

3,286 083 / Gunnar Nielsen, Johnson City, N.Y. / assignor to IBM Corp. / Information Storage System.

3,286,235 / Robert S. Sinn, Seaside Park, N.J. / assignor to Ultronic Systems Corp. / Information Storage System.

November 22, 1966

3,287,698 / Theodore Sapino, Framingham, Mass. / assignor to Honeywell Inc. / Data Handling Apparatus.

3,287,700 / Thomas Harold Flowers, London, England / assignor to Her Majesty's Postmaster General, London, England / Core Matrix Having Input and Output Lines Connected In a Priority Arrangement.

3,287,702 / Walter C. Borck, Jr., Arbutus, and Robert C. McReynolds, North Linthicum, Md. / assignors to Westinghouse Electric Corp. / Computer Control.

3,287,703 / Daniel L. Slotnick, Baltimore, Md. / assignor to Westinghouse Electric Corp. / Computer.

3,287.707 / Gerald Horace Perry and Sydney John Widdows, Malvern, England / assignors, by mesne assignments, to IBM Corp. / Magnetic Storage Devices.

3,287,708 / John R. Anderson, Los Altos, Calif., and Richard M. Clinehens, Dayton, Ohio / assignors to the National Cash Register Co. / Magnetic Data Storage Devices.

3,287.709 / Eric W. Moulton, Malibu, Calif. / assignor to Avco Corp. / High Speed Memory.

3,289,180 / William Albert Edward Loughhead, Beeston, Nottingham, England / assignor to Ericsson Telephones Limited, Beeston, Nottingham, England / Magnetic Core Matrices.

3,289,181 / Albert Zaretsky, Brooklyn, and Ottavio C. Cataldo, East Northport, N.Y. / assignors to American Bosch Arma Corp. / Multiaperture Core Memory Matrix.

3,289,182 / James C. Suits, Mount Kisco, N.Y. / assignor to IBM Corp. / Magnetic Memory.

3,289,184 / Douglas M. Brown, Bronx, N.Y. / assignor to Sperry Rand Corp. / Magnetic Core Memory Readout.

3,289,186 / George F. Bland, White Plains. N.Y. / assignor to IBM Corp. / Low-Noise Memory.

3,289.179 / Robert F. Elfant, Yorktown Heights, and Kurt R. Grebe, Beacon, N.Y. / assignors to IBM Corp. / Magnetic Memory.

November 29, 1966

3,288 919 / Henry H. Abbott and Renato D. Fracassi, Middletown, Edward G. Hughes. Leonardo, and Chester W. Lonnquist. Eatontown, N.J. / assignors to Bell Telephone Lab. / Data Transmission System.

3,288,985 / George Richard Hoffman, Sale, and Peter Lumsden Jones, Bramhall, Stockport, England / assignors to National Research Development Corp., London, England / Digital Information Storage Apparatus.

3,289.010 / James R. Bacon, Philadelphia, and George H. Barnes, West Chester, Pa. / assignors to Burroughs Corp. / Shift Register.

C&A CLASSIFIED COLUMN

Use economical C&A Classified Ads to buy or sell your computer and data processing equipment, to offer services to the industry, to offer new business opportunities, to seek new positions or to fill job vacancies, etc.

Rates for Classified Ads: 90¢ per word — minimum, 20 words. First line all capitals — no charge.

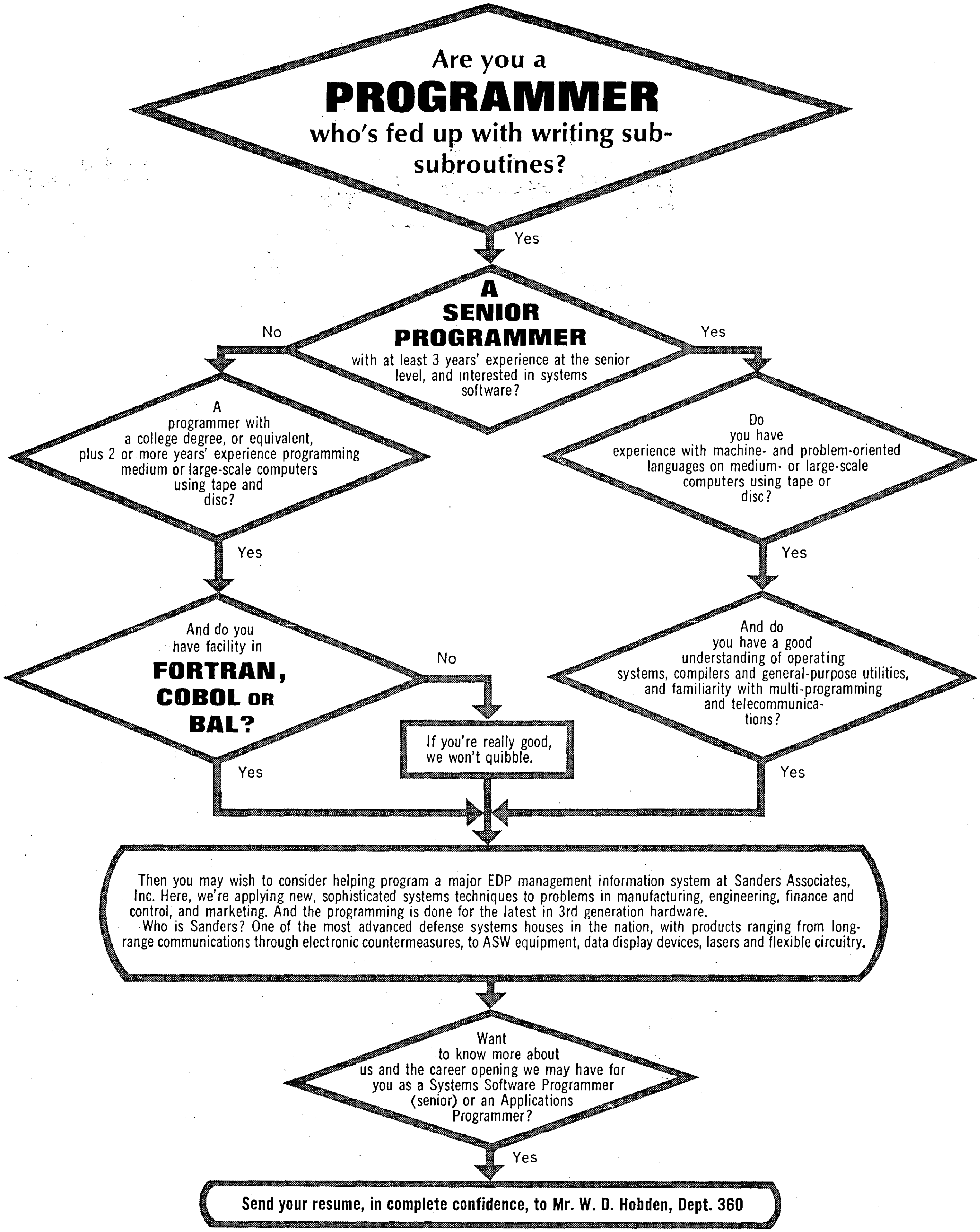
Blind Ads: Box Numbers acceptable at $4.00 additional to cover costs of handling and postage.

Send copy to: Computers and Automation, 815 Washington Street, Newtonville, Mass. 02160. Telephone: 617-332-5453.

Deadline for Classified Ads is the 10th of the month preceding issue.

ADVERTISING INDEX

Following is the index of advertisements. Each item contains: Name and address of the advertiser / page number where the advertisement appears / name of agency if any.



SANDERS ASSOCIATES INC.

NEW DIRECTIONS IN ELECTRONICS SYSTEMS

An Equal Opportunity Employer (M&F)

Nashua, New Hampshire