

U277

STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-2085

Do P.O. MACHINES NOW READ?
(Stanford does !!)
ARE THESE FONTS
PLEASED ON THE SAME
PAGE - OR IS THIS FOR
DEMONSTRATION PURPOSES

DONALD E. KNUTH
Fletcher Jones Professor
Department of Computer Science

Telephone:
(415) 497-4367

WOULD YOU LIKE SOME OF
HIS ORIGINAL CODING
FOR THE 650?

July 22, 1983

IF YOU FOLLOW THE POST OFFICE STANDARD
FOR ZIPCODE, WHY NOT THE U.S. & ISO
STANDARDS FOR DATE?

1983 JULY 23
1983-07-23

Bob Bemer Software, Inc.
2 Moon Mountain Trail
Phoenix, Arizona 85023

Dear Bob,

Thanks so much for the bundle of goodies. I will try to keep them both safe and accessible to future historians.

Cordially,

Donald E. Knuth/pw

Donald E. Knuth

DEK/pw

I WOULD BE INTERESTED IN THE ALGORITHM,
IF NOT A DICTIONARY LOOKUP.

HOE
SHOE
SKI PHOE
FOENIX

AT LEAST ITALY
USES THE "F"

Bob Bemer Software, Inc.

1983 July 16

Prof. Donald Knuth
Stanford University
Stanford, CA 94305

Dear Don:

I was going to copy pages 104 and 105 of the enclosed journal for the references on timesharing. Then I remembered you are interested in "firsts" in computers, so you get the whole issue, because it is the world's first journal/magazine published with its microfiche version as an integral (also free) part.

It has other curiosities, such as my personal view of COBOL history. When I wrote it, I did not remember who had coined the term "COBOL" (p. 132). At the COBOL Pioneer's Day of a recent NCC we all had the chance to recount one anecdote. Grace Hopper's was about the time I coined "COBOL", while saying it had a nice round sound, accompanied by hand movements depicting a woman's body. So long ago I did not even remember it, but the actions were typical of me at the time.

Also enclosed -- a copy of my article in AUTOMATIC CONTROL. I can find no internal documents or IBM memos about it, although there must have been at least discussion. There was little clearance control for publishing then, and De Carlo trusted me (although a Franz <somebody> suggested to him that I should be fired for it, as such a thing would be against IBM policy). The New Yorker reprint (2 months earlier) mentions a "community computer", so I must have been promoting the concept for some time. Interesting that the letter from Grace says nothing about that aspect.

I also enclose copies of correspondence with the Scientific American (which the Honeywell Computer Journal once bested in a Printing Industries of America competition). I would have to say that my claim, if it holds, might be primarily for the concept of "commercial" timesharing. The utility, where services are sold to a party that does not run the computer. Not the concept of multiprogramming. The Dodds response was the only one received from my request for earlier papers, and doesn't seem to fit very well.

I remember that you asked, by telephone, for any other material suitable to your project. All my files on ASCII, FORTRAN, COBOL, etc. are at the Smithsonian. But I have saved a few tidbits. The CV is provided for time framework. It is fun to see how many of my interests paralleled yours.

Typography, for instance. My ¹⁹⁶⁹1969 Apr memo to John McPherson got nowhere. Mel Shader admits he made a mistake. But it kept my interest high enough to persuade Charlie DeCarlo to fund Mike Barnett's work on SHADOW at M.I.T., which eventually led to PAGE I and the clipping overlaid on my copy. The voice recognition part took only 20 years to come true!

And polynomials. At the 1965 ACM Conference I was sitting with Dick Hamming when Householder walked in. Hamming asked "anything earthshaking in there today?" Alston replied "No, but Bemer here caused a few tremors". My first paper in the computer business.

The exchange with Baer is thrown in to show that you were not the only victim of my autocratic habits as an editor.

The data compression method is of some interest. It is mentioned in Kahn's book "The Code Breakers". He assured me that the method was quite secure. The interesting thing was that I seemed to have found an extra 2 bits on Shannon. Neither he nor Brillouin ever explained it. IBM never used it, but a British firm did, and RCA much later.

Now to what I consider a more important concept than timesharing, escape sequences (paper 17). The Skelly letter of nomination sums the value, I think. The ISO document shows early effort to make a registry. I tell my grandson that without my work he might not be able to play PAC-MAN. Skelly has a printable database of all of the registered uses of escape sequences, even unto the Chinese set, which is a huge one. Would you like a copy?

Finally, a document to show another concept at its early stages. I believe the ISO symbol will provide the unifying factor for all encoded knowledge.

Cordially,



R. W. Bemer

An IBM ad in a New York paper which recruited for "research programmers for digital computers" drew an unusual response. Miss Andy Logan, a reporter from the magazine, THE NEW YORKER, came to World Headquarters to interview R. W. Bemer, Assistant Manager of the Programming Research Department. Based on the interview, the write-up, which is reprinted below, appeared in THE NEW YORKER of January 5th.

Chess to Come

By permission, copr. 1957 The New Yorker Magazine, Inc.

THE International Business Machines people ran an ad in the *Times* a few weeks ago asking any "research programmers for digital computers" who might be interested in taking part in an "expanding research effort in the development and automatic translation of a multi-computer language" to apply to Mr. R. W. Bemer, assistant manager of the I.B.M. programming-research department. The ad suggested that programmers in related fields—language theory, logic, topology, and the like—might also be interested, and noted teasingly, "Those who enjoy playing chess or solving puzzles will find this work absorbing." Though we know practically nothing about digital computers except what we've seen of their fancy work on TV on Election Nights, and though we had never even heard of topology, we made bold to apply to Mr. Bemer. Not that we wanted a programming job, we told him; we just wondered if anyone else did. A fast-talking, sandy-haired man of about thirty-five, Mr. Bemer said that the ad, which was also run in the *Los Angeles Times* and the *Scientific American*, had brought a total of seven responses, and that although this might sound disappointing to us, I.B.M.

considered it excellent. There are some fifteen hundred digital computers scattered throughout the country (ninety per cent of them built by I.B.M.), and each of the larger models requires from thirty to fifty programmers—programmers being the clever fellows who figure out the proper form for stating whatever problem a machine is expected to solve. "All told, there are probably fifteen thousand trained programmers in the United States," Mr. Bemer said. "They're very well paid and always in short supply, so we

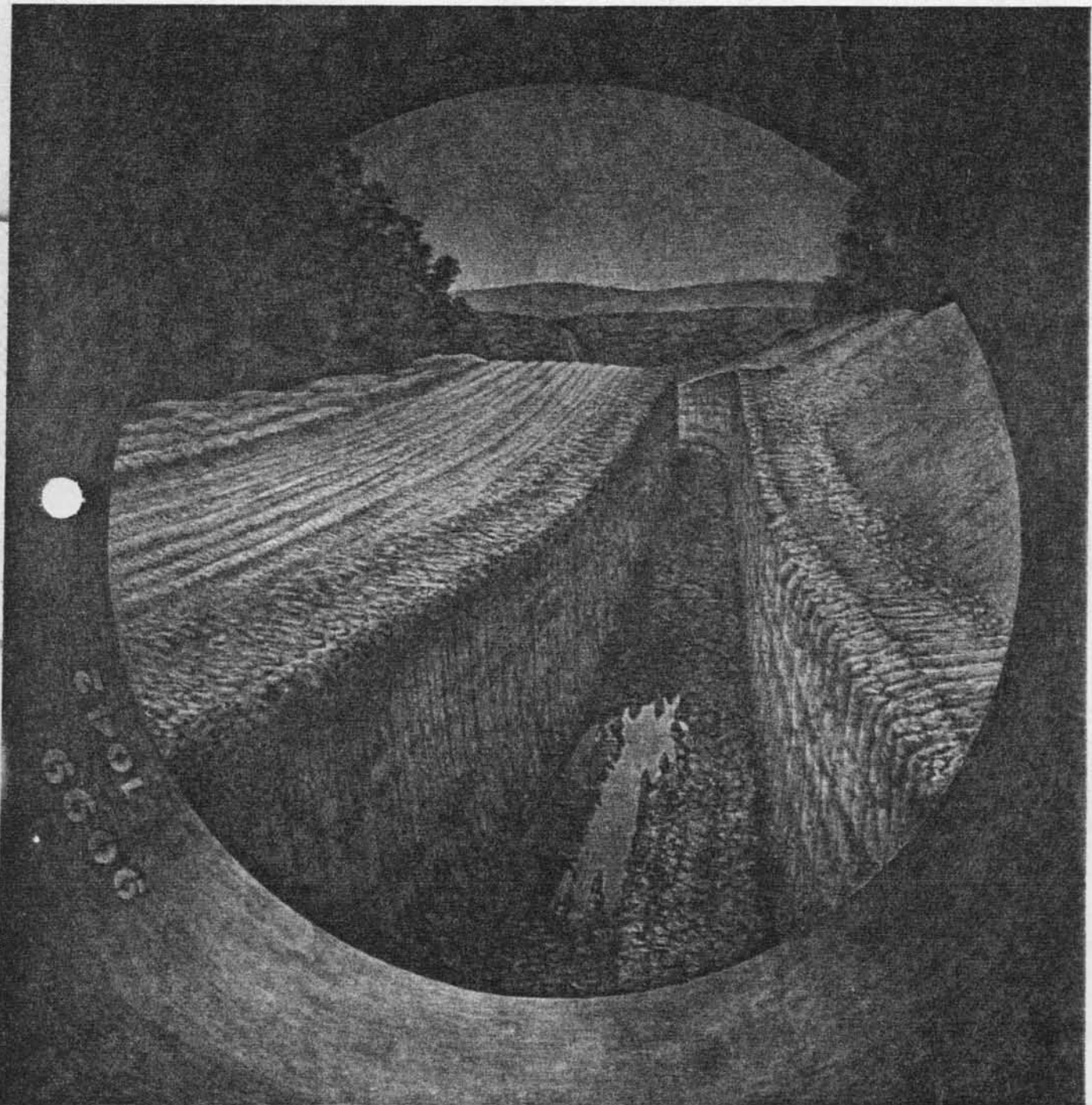
didn't expect many of them to be mooning over 'Help Wanted' ads. Of the seven who answered us, we hope to take on five in this department. The sixth man really *was* interested only in playing chess, and we let him go back to his board. The seventh man knew almost nothing about computing, but he had the kind of mind we like, and will no doubt be hired by some other department of the company. He has an I.Q. of a hundred and seventy-two, and taught himself to play the piano when he was ten, working on the assumption that the note F was E. Claims he played that way for years. God knows what the neighbors went through, but you can see that it shows a nice independent talent for the systematic translation of values."

We expressed modest astonishment that a profession we'd never heard of should have as many as fifteen thousand members. "Whole thing happened overnight," Mr. Bemer said, in a consoling voice. "I've been programming for eight years now and I'm considered an old man with a long beard." Small digital computers were used during the war, and the first giant, Mark I, was built by I.B.M. for Harvard in 1944. The latest I.B.M. giant, completed in 1955, is twice as big as Mark I and approximately fifty times as subtle. Since giant computers cost a couple of million dollars apiece to build and are quickly outmoded, I.B.M.'s usual practice is not to sell them but to rent their electronic services by the month. (Rental fees run from thirty thousand to fifty thousand dollars a month.) Computers are used by scientists, for working out abstruse calculations that it would take a man a lifetime, or many lifetimes, to work out by himself, and also by commercial enterprises. At the moment, the best computer customers are airplane manufacturers, who, in effect, can feed

the carrying capacity, fuel load, speed, and maximum range of a projected plane into a computer and in a few minutes' time will be given back what amounts to a complete design; moreover, each item of the design, down to the smallest rivet, can then be tested by the computer for all imaginable stresses and strains. Shipbuilders and bridge designers use computers in the same fashion.

We asked Mr. Bemer about the multi-computer language referred to in the *Times*. He said that I.B.M. has already developed two synthetic languages for its computers: Fortran, which is strictly for scientific use, and Print I, which can handle both scientific and commercial information. By I.B.M.'s strict standards, both languages leave much to be desired (what strikes us as miraculous is mere irritating clumsiness to I.B.M.); for example, though the point has been reached where computers can translate scientific data from Russian into English at the rate of four or five sentences a minute, the data requires pre-editing and post-editing. "We're out to develop a language that will let computers think pretty much as we do—make ready use of their stored memories and be capable of free association," Bemer said. "A computer has been designed that plays checkers and has beaten all comers so far. Chess is still beyond it, but won't be for long. There's no telling how many ticklish problems computers will someday be able to solve. I foresee the time when every major city in the country will have its community computer. Grocers, doctors, lawyers—they will all throw problems to the computer and will all have their problems solved. Some people fear that these machines will put them out of work. On the contrary, they permit the human mind to devote itself to what it can do best. We will always be able to outthink machines." Triumphantly, Bemer turned a sign on his desk in our direction. It read, "REFLEXIONE."

SCIENTIFIC AMERICAN



PIPELINES

SIXTY CENTS

January 1967

LETTERS

Sirs:

I was pleased to see William C. Livingston's article on solar magnetic fields in your November issue and to read his lucid account of the elements of the theory of the sun's magnetic cycle that I had published in 1961.

The purpose of this letter is mainly to call attention to a draftsman's error that originated in the second of three diagrams in Figure 5 of my paper (*Astrophysical Journal*, Vol. 133, pages 572-587, 1961) and that seems to have been carried over to Figure 6 (a, b, c) of Livingston's article (page 58). According to the theory, the twist of the submerged magnetic-flux ropes should have at all times predominantly one sign (that of a right-hand screw) south of the sun's equator, and vice versa; the twist should not reverse where the flux rope breaks the surface and a bipolar magnetic region is formed, as in the diagram.

According to the model, the flux ropes become twisted through the shearing or overriding effect of the (supposedly) shallow material of the equatorial belt of the sun (the well-known "equatorial acceleration"). This part of the theory finds confirmation, as I pointed out in my paper, in the finding of G. E. Hale

(*Nature*, Vol. 110, page 708, 1927) and later R. S. Richardson (*Astrophysical Journal*, Vol. 93, page 24, 1941) that a majority of chromospheric filamentary whirls surrounding sunspots in the Northern Hemisphere have the same sense of spiraling independent of the magnetic polarity of the spot or of the parity of the 11-year sunspot cycle, and that the opposite sense of spiraling prevails in the Southern Hemisphere. Hale (1927) showed that the prevailing sense of rotation of the chromospheric whirls is as if the equatorial edge of the central spot moves ahead in the direction of the sun's rotation.

Perhaps I may also point out that several features of the theory were developed as a result of extended observations with the magnetograph by Harold D. Babcock between 1952 and 1960. One crucial finding was that bipolar magnetic regions on the sun's surface are compact and intense when first formed, but that they disappear by expanding; this, as was emphasized in 1961, necessarily implies that the sun is continually expelling magnetic-flux loops that expand outward into interplanetary space.

HORACE W. BABCOCK

Mount Wilson and Palomar
Observatories
Pasadena, Calif.

Sirs:

In the article "Time-sharing on Computers," by R. M. Fano and F. J. Corbató [*SCIENTIFIC AMERICAN*; September, 1966] it is stated that Christopher Strachey first proposed a time-sharing system in 1959. This implies a faster development than actually occurred.

I have been unable to find a reference earlier than my article in the March 1957 issue of *Automatic Control*, which stated:

"One or several computers, much larger than anything presently contemplated, could service a multitude of users. They would no longer rent a computer as such; instead they would rent input-output equipment, although as far as the operation will be concerned they would not be able to tell the difference. Using commutative methods, just as motion pictures produce an image every so often for apparent continuity, entire plant operations might be controlled by such super-speed computers. Few computer manufacturers will deny the feasibility, even today, of super-speed and interleaved programs."

There is another reference prior to

Strachey's paper of June, 1959. This is an article by W. F. Bauer in *Proceedings of the Eastern Joint Computer Conference* (December, 1958), where he proposed:

"Each large metropolitan area would have one or more of these super computers. The computers would handle a number of problems concurrently. Organizations would have input-output equipment installed on their own premises and would buy time on the computer much the same way that the average household buys power and water from utility companies."

If any of your readers do know earlier references, I would appreciate being made aware of them.

R. W. BEMER

Computer Department
General Electric Company
Phoenix, Ariz.

Sirs:

It has been brought to our attention that our article entitled "Time-sharing on Computers" in the September 1966 issue of *Scientific American* gave the incorrect impression that the work at the M.I.T. Computation Center had its sole root in the paper presented by Christopher Strachey at the 1959 UNESCO Congress. In fact, the idea of time-sharing a large computer grew simultaneously and independently at the M.I.T. Computation Center. The implementation of a time-sharing system was proposed in an internal memorandum by Professor John McCarthy, dated January 1, 1959, entitled "A Time-sharing Operator Program for Our Projected IBM 709." Strachey's paper is, to our knowledge, the first formal publication that proposed and discussed in substantial detail the design of a general-purpose time-sharing system. Of course, computers already had been time-shared for special purposes, as in the SAGE air defense system in the early 1950's. General-purpose systems, however, did present a host of new, difficult problems. Thus there is no question that Strachey's paper had a very significant effect on the development of general-purpose time-sharing systems at M.I.T. as well as elsewhere.

R. M. FANO
F. J. CORBATÓ

Massachusetts Institute
of Technology
Cambridge, Mass.

Scientific American, January, 1967; Vol. 216, No. 1. Published monthly by Scientific American, Inc., 415 Madison Avenue, New York, N.Y. 10017; Gerard Piel, president; Dennis Flanagan, vice-president; Donald H. Miller, Jr., vice-president and treasurer.

Editorial correspondence should be addressed to The Editors, *SCIENTIFIC AMERICAN*, 415 Madison Avenue, New York, N.Y. 10017. Manuscripts are submitted at the author's risk and will not be returned unless accompanied by postage.

Advertising correspondence should be addressed to Martin M. Davidson, Advertising Manager, *SCIENTIFIC AMERICAN*, 415 Madison Avenue, New York, N.Y. 10017.

Subscription correspondence should be addressed to Jerome L. Feldman, Circulation Manager, *SCIENTIFIC AMERICAN*, 415 Madison Avenue, New York, N.Y. 10017.

Offprint correspondence and orders should be addressed to W. H. Freeman and Company, 660 Market Street, San Francisco, Calif. 94104. For each offprint ordered please enclose 20 cents.

Microfilm correspondence and orders should be addressed to Department SA, University Microfilms, Ann Arbor, Mich. 48107.

Subscription rates: one year, \$7; two years, \$13; three years, \$18. These rates apply throughout the world. Subscribers in the United Kingdom may remit to Midland Bank Limited, 69 Pall Mall, London SW 1, England, for the account of Scientific American, Inc.: one year, two pounds 11 shillings; two years, four pounds 14 shillings; three years, six pounds 10 shillings.

Change of address: please notify us four weeks in advance of change. If available, kindly furnish an address imprint from a recent issue. Be sure to give both old and new addresses, including ZIP-code numbers, if any.

1968 July 1

Mr. George E. Trimble, Jr.
Computer Usage Development Corporation
655 Madison Avenue
New York, New York

Dear George:

I have studied your bibliography on Timesharing (we spell it as one word nowadays), and noted the papers prior to 1961. It seems that it might be useful to order these first few chronologically for interest. I found:

#14, CR, 68 MAY, 291-301

Gill	Parallel Programming	1958 Apr.
Strachey	Time-sharing in large fast computers	1959 June
McCarthy & Teager	Time-shared Program Testing	1959 Sept.
Codd, et al	Multiprogramming STRETCH	1959 Nov.
Licklider	Man-computer symbiosis	1960 Mar.
ORION	(t-s data processing system)	1960 Apr.-June
Codd	Multiprogram scheduling	1960 June
Weinberg & Buist	Real-time multiprogramming in Project-Mercury	1960
Guntch & Handler	On time-sharing problems with digital computers	1960 Aug.
Paine	Is time-sharing a real winner for users?	1960 Sept.

To make your list more complete and correct, I suggest you add:

Bemar, R.W., How to consider a Computer, Automatic Control Magazine, 66-69. 1957 Mar

Bauer, W.F., Computer Design from the Programmer's Viewpoint, 1958 Dec Proc, AFIPS 1958 FJOC, 46-50.

On page 295 of the Computing Reviews, J.W.Weil has only the one 'L'.

R. W. Bemar

cc: W. F. Bauer, Informatics
Lee Revens, Computing Reviews

/eb

LORENZI, DODDS & GUNNILL

6800 INDIAN HEAD ROAD
WASHINGTON, D.C. 20022
248-8520 AREA CODE 301

March 28, 1967

Mr. R. W. Bemer
Computer Department
General Electric Co.
Phoenix, Arizona

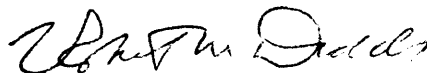
Dear Mr. Bemer:

In response to your letter in the January, 1967, issue of Scientific American, asking for early references to time-sharing on computers, I would like to quote from a letter I wrote on August 7, 1949, to IBM, listing several of my inventions and asking if they were interested:

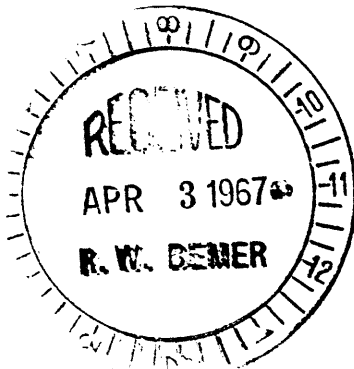
"5. A system consisting of the following: one or more input-output devices such as described in paragraph #2, each conveying its information to a common location distant from one or all of the input-output devices; one or more devices which generate the electrical impulses that convey information and that control the various operations of the several devices; one or more scanning or gating devices to segregate the information originating from the several input-output devices . . ."

No effective reply was received, and the matter was dropped at my end. I have no way of knowing if the matter was pursued further at IBM as a result of my letter.

Sincerely,



Robert M. Dodds





Date: April 6, 1959

From: R. W. Bemor

Department: Applied Programming

Location: DPDHQ

To: Mr. J. C. McPherson
Department: Research and Engineering
Location: WHQ

CC: Dr. C. R. DeCarlo
Dr. M. A. Shader

Subject and/or Reference: Market for Autotypography

Lately I have been pondering some of the aspects and possibilities of expanded character sets. I am passing along some thoughts which are perhaps not new but might be revised thinking.

7 JAN 1963

LA Times Designs Typesetter Using RCA 301 System

LOS ANGELES. — An automatic typesetting system utilizing an RCA 301 computer has been designed by the Los Angeles Times in conjunction with Radio Corp. of America's Electronic Data Processing division, Camden, N. J.

A spokesman for the Times said a standard 301 computer is used, but a special programming system had to be designed to set the type.

The editorial staff will use IBM electric typewriters with Soroband tape punch attachments, he said. The punched tape is fed into the computer and the paper tape output from the computer is fed into the Linotype machines, which have a teletypesetter attachment.

Through the specially developed program, the computer puts out a "justified" tape (words are adjusted to fit the newspaper column). Also included in the program is a hyphenation logic, which enables automatic insertion of a hyphen at the end of a line when a word has to be split.

If a sufficient set of characters is available (we are now planning for 512 in the IAL), the use of computers for typesetting and typewriting becomes more feasible. For instance, imagine the New York offices of a large newspaper with all typewriters connected to a reasonably powerful computer. A reporter or editor might conceivably type his copy on one of these and have it completely edited for publication and typesetting by the computer. This would embrace:

1. Search of a dictionary for correct spelling, complete to all principal parts of words.
2. Verification of correct grammar, possibly style.
3. Make-up composition for each line, verifying whether the overflowing word can be hyphenated; if so, giving the correct partitioning.
4. After a line is complete, respacing to effect right- as well as left-justification.
5. Selection of proper font and case.
6. Information retrieval on previous related stories or articles.
7. Typed warning of possible dangers under libel laws.

In short, all the functions of the proofreader and the linotypist would be performed automatically and without error. Since the speed of the computer would probably be much greater than that of the many typewriters connected, rather elaborate intelligence could be built into the program. Presumably other available time could be used for accounting and billing functions.

April 6, 1959

This same principle could be extended somewhat to a local area where a large central computer could be used as a ubiquitous secretary. Given the capability of understanding vocal phonetics (a la Bell telephone), the businessman could use a voice pickup the same way he uses dictating equipment today. The major difference is that the typewriter at his side is being driven (with some time lag) by the computer. Virtually coincident with the completion of dictation, the finished, error-free, pleasing, right- and left-justified letter is ready to be mailed. Presumably each possessor of such equipment would have a code number so the computer could bill him periodically for proportionate usage.

I suggest that a modest investigation could be undertaken now to experiment with programs of this class. I do not suggest that IBM do it but rather that Dr. Shader might possibly let a research contract to a university for such a study.

Secretaries are harder to get every day, partulary ones with perfect grammar and flawless typing.

rwb/ep

R. W. Bemer, Manager
Programming Systems

April 8, 1959

Memorandum to Mr. H. R. Keith

Attached is a "blue sky" proposal by R. W. Bemer of Data Processing which I hope you will pass on to the interested groups doing our really forward planning.

While this may sound fanciful at the moment, it is certainly no more fanciful than machine language translation or information retrieval were five years ago. I feel there is a group of applications, of which this one is a good example, where computers may do elaborate manipulation of information on a routine basis in the future.

Many of the ideas which will make this possible are being worked on today in a number of the universities interested in the programming aspects of electronic computers.

A review of these activities with a view toward trying to formulate a list of the information-analyzing-type potential applications might be useful at this time.

JCMcP:im
Attachment


John C. McPherson

b cc:
Mr. R. W. Bemer

DPDHO, White Plains
May 5, 1959

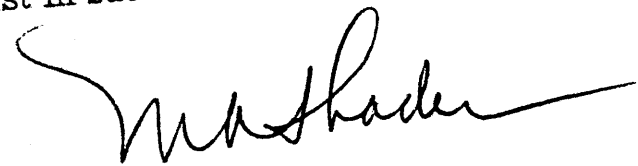
Memorandum to Mr. R. W. Bemer

Subject: Market for Autotypography

(It died here)

Reference: Your letter of April 6, 1959

While I believe that some of the ideas expressed in your memorandum will be of value in the future, I do not believe they will have any immediate impact on the IBM product line. I do agree that some of the background work that would have to be done in order to carry out this investigation would be profitable. If you have a university in mind which would be qualified to perform such an investigation, and more importantly if you have an idea of who in IBM would be able to fund it, I would be glad to assist in such an investigation.



M. A. Shader, Manager
Engineering and Sciences
Industry Marketing

MAS/bp

9 March 62

Mr. Robert W. Bemer
IBM
112 East Post Road
White Plains, New York

Dear Mr. Bemer:

Do papers appear in the Communications of the ACM in the order in which they are received? If not, what governs the order? What is the backlog of papers?

Very truly yours,

Dr. Robert M. Baer
Member, ACM

62 March 12

Dear Dr. Baer:

Your letter hints that you have sent me a paper. Is this true?

R. W. Bemer

62 March 19

Dear Mr. Bemer:

As an Editor of the Communications of the ACM, you presumably are able to consult your records to determine whether I have submitted a paper, as well as supply the information which I requested.

Your letter hints this might not be the case. Is this true?

Dr. Robert M. Baer

62 April 4

Dear Dr. Baer:

Unless you are convinced that you handed a paper to me personally, you will have to consult the records of the U.S. Post Office Department, not mine.

In answer to your questions:

1. Approximately
2. My whim and/or judgment
3. Today - 8.

R. W. Bemer

(And here the matter dropped)

Data Processor

for managers of IBM installations

Digital Code Compresses Information to Speed Data Communication

A coding scheme that may triple the data-handling capacity of communication links was recently described by IBM.

It's a kind of "digital shorthand," using digital representation or symbols for entire words or common phrases, rather than character-by-character representation. The code can compress language and, to a lesser degree, numbers. Fewer signals are needed to convey a message.

Illustrated on the left and below is an actual-size example of how digital shorthand effects message compression on five-channel teletype tape. Nearly twelve inches of tape, represented in grey, were required for the original message. Coded in digital shorthand, the message occupies about four inches of tape, represented in white in the illustration.

An IBM feasibility study indicates the

application might be used in radio, wire and cable systems, and satellite communications. Estimated savings in message lengths: about 60 percent. This would significantly reduce message transmission time and cost for some users of long-distance communication systems.

The digital shorthand can be used over existing transmission equipment. At the sending point a computer-encoder would compress the message by translating words into numbers. At the receiving end a companion unit would convert the numbers back into words. The new code is already programmed for use on the IBM 7090, and work is going forward to program it for application to other IBM equipment.

The coding scheme is an application of the branch of mathematical science called information theory.

DEVISER OF THE NEW digital shorthand — Robert W. Bemer, manager of IBM's Logical Systems Standards — points out that a Binary number is substituted for each word. And the system goes beyond conventional cryptography because numbers are assigned on the basis of usage frequency: most-used words get the smallest numbers.

Binary Numbers are the most efficient representation, not only in digital computing but also in wire and radio Transmission.

THIS IS A SAMPLE OF THE DIGITAL SHORTHAND TECHNIQUE DEVELOPED BY THE INTERNATIONAL BUSINESS MACHINES CORPORATION

BXGMAGZRLFJOBMAPQULXPWDZCALAWYBT 6.2\$.

Shorthand tape and message

Original tape and message

Here's how the sample of digital shorthand corresponds to words in the original message:

BX G M AGZ R L FJ QBBM...
THIS IS A SAMPLE OF THE NEW DIGITAL...

BUSINESS MACHINES CORPORATION

under development:

Computer-directed Map for Airborne Navigation

Pilots of supersonic craft may soon be able to determine their location over the earth by glancing at a screen on their instrument panel.

An experimental system now under refinement by IBM engineers can project a highly accurate circular map on a screen. The screen is 7½ inches in size; the earth area projected on it is 400 miles in diameter—125,000 square miles.

Calculations to position the map on the screen, and to pinpoint the pilot's location, are performed by an airborne computer.

A detailed map of half of the earth



IBM ENGINEER with hemispherical glass map, part of new air-navigation system

is reproduced photographically on the inside surface of a glass hemisphere about six inches in diameter. A section of the map, illuminated by a beam of light, is projected onto a flat, translucent screen in front of the pilot. As the plane moves, the computer automatically adjusts the map presentation.

voter registration:

Machine Assigns Precinct Numbers

In Los Angeles County, which has a voter population of more than three million, an IBM RAMAC® 305 has helped solve the problem of processing last-minute registrations or re-registrations of voters.

The machine assigns precinct numbers in an average of two seconds per registration. Previously, registration personnel had to search for the information on a master map containing 30,000 street names and 10,964 precinct numbers.

RAMAC began operating in March at the office of the registrar of voters—just in time for the April deadline for registering in California's June primary election.

Cards punched with registrants' names and addresses are fed into RAMAC; the machine either assigns the correct precinct number or rejects the card if it is incorrectly made out. The cards then go to an IBM 407 that prints sample ballot address inserts, voters lists, etc.

jury panels:

Selection Procedures Mechanized on Cards

In an effort to enlarge the New York County jury panel, the Jurors' Division of the County Clerk's Office has innovated punched-card records and mechanization with IBM unit-record equipment. Objectives are a more equitable spread of jury service among residents of the county, and longer intervals between calls upon individuals for jury service.

Punched cards replace paper jury ballots. In place of the traditional jury wheel are automated machines to scramble the punched cards, select jurors' cards at random for service, list the names of those drawn, prepare panels for the courts, and address, enclose, seal and stamp the jury summonses.

The inaugural drawing of thirteen hundred jurors for four courts recently was performed in 21 minutes. This time included scrambling of the punched-card ballots, machine selection of the jurors, and writing minutes of the drawing.

new product:

Low-cost System Handles Card Data by Telephone

The IBM 1001 Data Transmission System sends data, in machine language, on regular telephone lines—local, long-distance, or private. Another in the growing line of IBM TELE-PROCESSING* systems, it is for any size organization with one or more locations, whether many points across the country or a few within a community or building.

Fixed and variable data from one or multiple points of origin are received at a central point in the form of punched cards. These can be entered directly into an IBM data processing system to automate requisitioning, billing, inventory control, and so on.

To send information is like dialing someone on the phone. To receive, however, no one need answer; if the machine is not busy, the output station is automatically connected to the telephone line, and processing begins.

At each data-originating or sending station is a terminal, smaller than a typewriter, that contains a simple card reader and keyboard, and a modulating subset available through the local telephone company. The card reader permits automatic transmission of fixed data from prepunched cards, plus manual entry of variable numeric data. On the keyboard, in addition to ten keys for variable information, are five operating keys for remote control of the card punch at the central location.

From any one card, the card reader transmits a maximum of 22 numeric characters. (Additional variable information, entered through the keyboard, then can follow.) Remaining card columns may contain prepunched and interpreted alphabetic description data; although this alphabetic data is not transmitted, it facilitates visual use of the cards at the originating point.

At the central receiving location (equipped with a telephone company

* Trademark

TRONIC NEWS, MONDAY, JUNE 6, 1960

TUESDAY

NEW YORK
Herald Tribune



GET THE MESSAGE—Development of a new coding system, called "digital shorthand," was announced today by International Business Machines Corp. Using digital representation, or symbols, for entire words or common phrases, rather than character-by-character representation, the new system is said to have the potential for tripling the data-handling capacity of communications systems and cutting message transmission costs by possibly two-thirds. In the photo above, I. B. M. receptionist Judith Shepard demonstrates that the same amount of information put into "digital shorthand" can be carried on a tape 60 per cent shorter than the mass of tape she holds in her right hand.

IBM Develops Numeric Data Transmission

Special to Electronic News

WHITE PLAINS, N. Y. — A binary-coded information transmission scheme that could triple the capacity of present Atlantic cables, and also promises vastly increased language and cipher translation, has been developed here by the International Business Machines Corp. Data Systems division, Electronic News learned last week.

According to Robert W. Bemer, manager, logical systems standards, this development is one of the first large-scale practical users of information theory. Such theory states, in part, that numeric representation of entire words of common phrases would be far more efficient than present character-by-character methods of transmission.

Mr. Bemer said that the savings in transmission time would be on the order of 60 per cent, no matter what the transmission medium happened to be: Cable, radio, Teletype, and so forth. For instance, if a Teletype-punched tape now requires five or more bits to transmit a character, the IBM scheme might require only two bits to transfer the same information.

The key to the plan is a pair of computers. The sending station would feed the original message in any input form into the computer, which would then reduce the information to a binary numeric representation. This would then be transmitted to a similar computer on the other end of the line, which would convert the coded message back into its original form.

The company said it had proven out its encoding plan with an IBM 7090 computer and found it thoroughly feasible, though a much more simplified computer could be designed for the purpose. Mr. Bemer stated that almost any of IBM's current computers could handle the encoding if properly programmed, and probably could handle it in addition to primary functions of data processing.

He said he is currently working on a vocabulary capable of handling 4 million words and phrases, a vocabulary that would be automatically updated by the computer. English language unabridged dictionaries contain less than 600 thousand words.

1982 March 1

Charles W. Bachman
Chairman, Turing Award Committee
Cullinane Data Base Systems Inc.
400 Blue Hill Drive
Westwood, MA 02090

This letter is a re-nomination of Bob Bemer for the ACM Turing Award. A copy of an earlier nomination by Richard M. Petersen is enclosed for your review. I concur fully with Dick's remarks, and offer this nomination to focus on particular aspects of Bob's contributions and service.

Bob Bemer has made many significant contributions in diverse areas of the computing field. I view his work related to ASCII as having the most pervasive and enduring merit; most particularly the concept of the 'escape sequence'. Bob invented the 'escape character' in 1960, and proposed the registration of escape sequences in 1962. The registration concept was adopted by the International Organization for Standardization in 1974 as ISO Standard 2375. It is also the subject of a recently adopted American National Standard, ANSI X3.83.

Attachment 1 provides for your consideration three of Bob's articles about ASCII. The last of these is primarily on escape sequences and their use for code extension and control.

Much of what is possible today is derived from this escape sequence concept. Two examples suffice, I believe, to show the need, importance, and merit of the approach first postulated by Bob Bemer.

1. Video terminals depend on control sequences initiated by the escape character in accord with the national ANSI X3.64 and international ISO 6429 standards for soft copy controls. Cursor keys generate them. Function keys generate them. They are used to shift back and forth between linear text string and graphics modes.

Attachment 2 is a listing of the internationally standardized control sequences to do these things, and more.

2. Escape sequences are assigned for the denotation of alternative character sets from among the symbols of the world -- for data communication, for database storage, for printing.

Attachment 3 lists the currently registered or proposed character sets which are declared by an internationally recognized escape sequence. Note that ASCII, the American Standard Code for Information Interchange, is registered as but one of many equals: Videotex, Cyrillic, Kata Kana, Scandinavian, ...

Attachment 4, by way of an example, is the Arabic character set.

Attachment 5 is the present ECMA (European Computer Manufacturers Association) work on a register of individual symbols of the world. Special coded character sets can be assembled from these, and then registered as invokable by a specific escape sequence, to identify them in either communication or in a data base. The work is open-ended; note that symbols are yet to be provided for an astronomical set.

The escape sequence is critical to worldwide interchange of computerized data. Just try to think of another way to do it! 16-bit characters? Very wasteful, and insufficient. The Japanese and Chinese graphic sets, invoked by escape sequences, are already each a distinctive 16-bit character set. And they acknowledge an escape sequence to return to the 8-bit environment.

I therefore have the pleasure, for this important contribution to our singular and joint ability to manage and communicate data, to nominate as a candidate for the ACM Turing Award:

Robert W. Bemer
2 Moon Mountain Trail
Phoenix, AZ 85023
602-942-1360

The full story of how this knowledge and capability has, and will, come together for our mutual benefit and interaction is not generally well known. With Bob's comprehensive records, that story would make an excellent and interesting Turing Lecture.

Among those that I know, I offer the following people as references for this nomination:

John Auwaerter
Vice President, Engg.
Teletype Corporation
5555 West Touhy
Skokie, IL 60076
312-982-2300

Herbert S. Bright
Computation Planning, Inc.
7840 Aberdeen Road
Washington, DC 20014
301-654-1800

Charles D. Card
Univac Div. of Sperry Rand
P.O. Box 500, MS B105M
Blue Bell, PA 19422
215-542-3675 or 3867

Eric H. Clamons
Honeywell Information Systems
PO Box 6000
Phoenix, AZ 85005
602-866-4117

John A. Gosden, Vice Pres.
Equitable Life Assurance Soc.
1285 Ave. of the Americas
New York, NY 10019
212-554-2323

Thomas N. Hastings
Digital Equipment Corp.
Continental Blvd.
Merrimack, NH 03054
603-884-6767

M. Dara Hekimi, Secy General
ECMA
114 rue du Rhône
CH-1204 Geneva
SWITZERLAND
+41 22 35-36-34

Vico Henriques, President
CBEMA
1828 L Street NW
Washington, DC 20036
202-466-2288

Robert S. Jones, Publisher
Interface Age Magazine
16704 Marquardt Avenue
Cerritos, CA 90701
213-926-9544

Richard M. Petersen
Honeywell Information Systems
PO Box 6000
Phoenix, AZ 85005
602-866-6000

Hugh McGregor Ross
Data Systems Consultants
5 Kingsley Place, Highgate
London N6 5EA
ENGLAND
01-340-2376

Olle Sturen, Secy General
International Standards Org.
1 rue de Varembe
1211 Geneva 20
SWITZERLAND
+41 22 34 12 40



Patrick G. Skelly
3202 W. Dailey Street
Phoenix, AZ 85023
602-866-4798

TECHNICAL COMMITTEE 97

COMPUTERS AND INFORMATION PROCESSING

Working Group E

Programming Languages

Reference: ISO/TC 97/WG E (USA-3) 22.

Subject: Programming language implications on coded character sets

Gentlemen:

A. The following points of consideration are proposed as a basis for study of character sets, as proposed by WG B and others, with regard to their suitability from a programming language viewpoint:

1. Are the sets capable of graduated size by means of regular and clearly defined rules of expansion?

2. What sizes are of particular concern to programming language needs?

4 - BIT	16
5 - BIT	32
6 - BIT	64
7 - BIT	128
8 - BIT	256
others?	

3. What is the preferential ordering of graphics for inclusion in sets of varying size?

4. Are there natural orderings of Sub Groups of graphics which should be reflected in the coded representations? Are there artificial orderings which have caused unnecessary constraints?

5. Are control characters the concern of programming languages, as well as information characters?

6. Are there defined correspondences between multiple character symbols used with smaller sets and the single characters available in larger sets? Is there a controlled pattern of replacement with increasing set size? For example, single characters for ALGOL word symbols, or multiple symbols such as — for :=, ^ for / \.

7. Are criteria specified for controlled future expansion and addition?

8. Is an "escape character" provided to allow for usage of alternate, specialized character sets? What graphic subsets should remain in common positions in alternate sets so that they may be uniquely recognizable?

9. What values following the escape character (see section B of this document) should be reserved for specialized character sets for programming languages?

10. What control considerations are necessary to utilize alternate or graduated equipments?

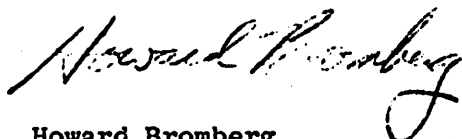
11. Are sufficient data delimiters included in sets (of sufficient size) to adequately indicate set membership in data groups?
12. Are sufficient delimiters included in sets (of sufficient size) to adequately indicate syntactic grouping?
13. What characters are required to map two-dimensional flowcharts into one-dimensional string languages? For example, a character signifying "is inside of" and characters for the shaped elements and flow lines.
14. Are the character sets equally applicable to conventional von Neumann machines and streaming (or other types of non-von Neumann) machines?

B. From the attached (proposed) IFIP definition of "escape character" it may be seen that it is possible to have many compatible alphabets within the framework of an international character set and code. The imminence of such a standard interchange code and the existing problem of interchange of algorithms suggest that the time is proper for some advanced planning in the area of characters used in programming languages.

It is proposed that a tentative group of characters (to follow the escape characters) be reserved to indicate special alphabets for programming languages. Further, work should begin at once on a specific alphabet for interchange of ALGOL and COBOL programs.

It is advisable that a generally agreed upon nucleus of graphic assignments (e.g., common to the ECMA BSI and ASA draft codes) be retained in the programming set. Also those ALGOL and COBOL graphics defined in any of these sets should be preserved. Consideration should then be given to adding the rest of the ALGOL characters, in particular reserving positions for the "complete word" characters of ALGOL. Among the further graphics which may be considered are those likely to have special usage stemming from mathematics and logics, delimiters which imply set membership hierarchy, delimiters for syntactic entities such as phrases, flow chart symbols, etc.

R. W. Bemer



Howard Bromberg
U.S. Representative
ISO/TC 97/WG E

Honeywell

1981 December 03

Olle Sturen, Secretary General
International Standards Organization
1 rue de Varembe
1211 Geneva 20
SWITZERLAND

Dear Olle:

You have received from me some informal communication on the subject of an ISO character (grapheme or mark) for the 8-bit character set. This is a more formal exposition that may be used to explore the need within your organization.

Graphic Representation

First let us dispose of the graphic shape itself, assuming that such an encoded character is desired in the 8-bit set. I think that it is inarguable that the design I propose here is the most appropriate.

- Let us take the "Oh" of "Organization", with a circular shape.
- We superimpose the "S" of "Standards" to get a yin-yang symbol (Shepherd No. 64, "Tomoye", the Japanese symbol of honor and triumph).
- We superimpose further the "I" of "International", and have our graphic shape for the ISO character.



It is legible. It obviously stands for ISO. It can be made on present typewriters by using backspace, although in the future it would be a unique character by itself.

Meaning

This symbol is in one way akin to the copyright or recording symbols. It is in another way similar to the present ESCape character and symbol. When encountered in a stream of characters (as in a database) it says:

"The number which follows is the number of a list registered by the ISO -- when that number is terminated (by length or explicitly), the symbol or token that follows is an explicit member of that registered list, and has the meaning assigned in that list."

Honeywell

When encountered free-standing it says:

"The number which follows is the number of some standard registered by the ISO, and the product upon which it is written is made in conformity to that standard."

In other words, the symbol means "the ISO list of lists". It is the means to unify all of the codifications in the world!

Examples

In the following examples, the registry number has been terminated by the explicit "-", not by constant length. Either method is acceptable, but I must show examples (wherein the registry numbers are fictitious).

- 0123-HON = Honeywell, because
ISO registry number 123 is assigned to symbols of the United States Stock Exchanges, who furnish and police their assignments. ISO only assigns the number, and identifies it in its list of lists.
- 0321-DOM = the Dominica, West Indies airport,
0321-HON = the Huron, SD, US airport, because
ISO registry number 321 is assigned to the airport codes of IATA, which furnishes and polices their assignments.
- 0241-DOM = Dominican Republic, because
ISO registry number 241 is assigned to the country codes of ISO 3166 (and to ANSI Z39.27)
- 0242-DOM = Dominican Pesos, because
ISO registry number 242 is assigned to ISO 4217, Currency.
- 066666-SV63 = Rescue vessel No. 63, because
ISO registry number 66666 is assigned to an UNESCO Universal Ship Code.
- 033333 The printer with this marking produces digits according to Standard ECMA-51, Implementation of the Numeric OCR-A Font with 9x9 Matrix Printers.
- 033334 The printer with this marking conforms to "Safety Requirements for Data Processing Equipment", Standard ECMA-57.

Honeywell

- 04337 Magnetic disk drives and magnetic disk packs with this marking conform to ISO Standard 4337 (here the number is the same as the number of the standard; it may be desirable, but not mandatory).
- 01000 Products so marked are of metric manufacture.
- 02955,II The document carrying this mark, along with copyright notice, ISBN, etc., was printed with Form II representations of SI Units, according to ISO 2955.

Registry Number Assignment

If the ISO should decide to create such a registry with this symbol, it would probably fall within the scope of TC97/SC14 to determine the precise usage. I certainly have no suggestions in such an expert area. TC97/SC2 should be informed as soon as possible, so it can assign the ISO character a code in the 8-bit set. Two positions in column 10, and eight positions in column 13, are now still open for future standardization.

I can offer (at random) some preliminary considerations and suggestions for usage of the ISO character/mark/symbol:

- In fixed data fields in records, it may be more economical to have the encoding known by descriptors for such fields. However, the ISO symbol is designed to be free-standing in defining an entity. It is non-contextual.
- The free-standing property is not possible by spelling out "ISO" in letters (e.g., "PROVISO 355" conflicts).
- In the case of revisions of registered lists (which is frequent with airport codes, for example), the registry directory would have to indicate the latest revision in force. This directory would be published by ISO, and sold to support operational costs.
- It is likely that availability of the ISO symbol in the world's common character set, and its usage (in product marking and identification, in databases, in Videotex and Prestel), would enhance the visibility, prestige, and influence of the ISO.
- As shown for the country codes, a single ISO registry number can be a pointer to multiple standards that may be identical. As usage matures, the ISO registry number can eliminate the need for this duplication.

Honeywell

- In the case of product markings, there may well be many such on a single product, indicating various conformities. They may be augmented verbally, at option, e.g.

Ø1000 (metric thread)

- Theoretically there is not a standard, nor any list of token-to-entity correspondence in the world that cannot be accommodated with this method of identification.
- Structure of the registry number (if indeed there is to be any, other than a randomly assigned list) is in the province of the data element experts themselves. It is to be hoped that none of the universality will be compromised by its design. Possibly letters could be components as well as just digits.

Finally,

I trust that the ISO will be convinced that this ISO symbol will standardize the identification of standards worldwide. This concept has been churning in my mind for a year and a half (published tentatively in the Danish magazine "Data"). I think it is time to GO!

Cordially,



R. W. Bemer

cc: Secretariat, ISO TC97
Secretariat, ISO TC154
Secretariat, ISO TC97/SC2
Secretariat, ISO TC97/SC14
Secretariat, ECMA
Secretariat, ANSI X3L2
Secretariat, ANSI X3L8
Secretariat, AFNOR/TC97/SC14
Secretariat, ANSI/SPARC
Secretariat, BSI/O15/14
Secretariat, UNIPREA/TC97/SC14

Erik Bruhn, Data Magazine, Copenhagen
Dr. Thomas N. Pyke, US National Bureau of Standards
FIPS/TG17

Honeywell

1974 January 23

Professor Donald E. Knuth
Computer Science Department
Stanford University
Stanford, CA 94305

Dear Don:

Glad you asked. I have always been a little irked about having this paper left out of the timesharing bibliography in Computing Reviews.

The timesharing aspects are given on the last two pages. Evan Herbert was the editor at the time, and persuaded me to write the article.

Cordially,

Bob
Bob Bemer

n

Encl.

DATA Control

What the Engineer Should Know About Programming

How to Consider A Computer

Engineering is taking on a "new look." Computers are the logical and more powerful successors to the desk calculator and the slide rule, the previous working tools of the engineer. There is really only one major difference: because of their necessary size and cost to be so powerful, computers must be shared by a great many users. This means a new concept of shared system operation must be accepted by the engineer.

To help you get oriented, here are some vital considerations affecting present and future computer use in your work and some helpful sources of further highly specialized information.

BY ROBERT W. BEMER

*Programming Research Department
International Business Machines*

■ A computer should not be rented or purchased unless an automatic programming or coding system is furnished for its operation. The computer and the operational system constitute a matched pair, and one without the other is highly unsatisfactory from the point of view of getting work done at minimum cost.

For engineering work, any automatic system should contain provision for indexing and floating point operation, if these are not built in as hardware, for they are the two most vital features for easy usage. Indexing allows for algebraic array nota-

tion, which in turn makes for easy understanding of how a problem should be programmed. Floating point, although it may sometimes introduce either spurious accuracy or loss of it to the uninitiated, prevents a Gordian tangle of scaling difficulties from cluttering up the problem.

HOW CODING SYSTEMS HELP

Automatic coding systems have by no means reached their ultimate efficiency or sophistication, yet remarkable savings in programming costs have already been achieved, sometimes by an order of 50! For the best of the present systems it is a reasonable estimate to say that they can, in general, reduce the programming

costs and time to a tenth of that required to code in stubborn machine language.

There have been many attempts to relieve the burden of programming through special coding systems of all types. The data sheet on computer coding systems is not only an interesting history of growth, but is also presented for the edification of those now entering the field with incomplete knowledge of what code to use for their machine. The time may come soon when you will be using a common language exclusive of the characteristics of any particular computer. Thus, with an automatic translator for each different computer, a running program may be produced for any desired machine from the single original problem and procedure statement in the common language. Credit is due to Dr. Saul Gorn of the Moore School of Electric Engineering for first championing these principles.

GOOD COMPUTER OPERATION IS STATE OF USER'S MIND

It is axiomatic that a computer should never stop, run useless problems or be subjected to manual oper-



ation and dial-twiddling. To do so deprives your fellow engineers of its benefit. Here are some detailed considerations pertinent to good computer operation:

▶ It can do only what it is explicitly ordered to do, and this ordering must be done eventually in its own machine language, which is all it can understand.

▶ The reliability of most present-day computers is so high that answers are not right or reasonable, the chances are at least 99 to 1 that it is somehow the user's fault. Wrong answers usually stem from wrong equations or misuse.

▶ Allow for growth when doing the original planning. Build in flexibility for changes, or else costs will soar if the entire problem must be re-programmed. A stored-program may always be corrected or augmented to give exactly what the engineer desires, including special report format for jobs where repetition justifies the effort.

▶ For design studies, plan parameter variation carefully and allow flexibility in changing individual parameters. The computer may surprise you by showing that some parameter values for optimum conditions may be outside of the range expected or allowed for. To make certain that the

program gives answers consistent with hand-computations, first test the program on the machine for a specific combination. Time this run. Then multiply this time by the total number of parameter combinations to see if the study is feasible in time and cost. If M parameters are combined for N values each, the total number of combinations is N^M .

For example, with 6 Parameters: 6 values for each will produce 46,656 combinations; 5 will produce 15,625; 4 will produce 4,096.

The moral: *Don't triple the cost of your problem if you are engineer to draw a curve through one less point.*

FUTURE COMPUTER LANGUAGES

New synthetic languages are in the process which will affect your use of computers. As problem-solving languages they will be much superior to present systems in these ways:

1. Even though the binary type of computer will probably be universal for both engineering and commercial work, the need for the user to know binary representation will effectively disappear. Logical decision will be the only remaining function which will not appear in decimal form, and even here facilities will be provided so that the programmer need not concern himself with the precise method of operation within the machine. Conversion from fixed point decimal to floating point binary for operation, and the converse for output, will all be done automatically by the synthetic language translator.

2. The elements of the synthetic language will be essentially algebraic, both arithmetic and logical, and linguistic so that procedures may consist of real sentences in a living language. Idiom will be such that the program will be operative in any spoken language, with minor changes



DATA Control

COMPUTER	SYSTEM NAME OR ACRONYM	DEVELOPED BY	CODE	SYSTEM TYPE				OPER. DATE	INDEXING	FL PT	SYMB.	ALGEB.	COMMENTS
				M.L.	ASSEM.	INTER.	COMP.						
I.B.M. 704	R-S Cage Fortran NYAP Pact 1A SAP	Los Alamos General Electric I.B.M. I.B.M. (See Pact Group) United Aircraft	1		X		X	Nov. 55 Nov. 55 Jan. 57 Jan. 56 Jan. 57 Apr. 56	M2 M2 M2 M2 M2 M2	M M M M M M	1 2 2 2 1 2	X	Modified Pact I for 704 Official Share Assembly
			1		X		X						
Sperry-Rand 1103A	Compiler I FAP Mishap Trans-Use Use	Boeing, Seattle Lockheed M.S.D. Lockheed M.S.D. Holloman A.F.B. Ramo-Wooldridge	1	X	X	X	X	Mar. 57 Oct. 56 Oct. 56 Nov. 56 Feb. 57	S1	S S S S M/S	1 0 1 2 2	X	Magn. Tape Assembly + Correction Official for Use Organization
			1		X		X						
1103	Chip Flip/Spur Rawoop Snap	Wright Field Convair San Diego Ramo-Wooldridge Ramo-Wooldridge	1	X		X		Jun. 55 Mar. 55 Aug. 55	S1 S1 S1	S — S	0 1 1		Similar to Flip Spur Unpacked, Twice as Fast One-Pass Assembly Used with Rawoop
			1		X		X						
I.B.M. 705	Autocoder Fair Print I Symb. Assem. SOHIO	I.B.M. Eastman Kodak I.B.M. I.B.M. Std. Oil of Ohio	1	X	X		X	Dec. 56 Jun. 56 Oct. 56 Jan. 56 May 56	— — S2 — S1	S S S — S	2 0 2 1 1		May be Assembled on Acctng. Equip.
			1	X	X		X						
Sperry-Rand Univac I, II	A0 A1 A2 A3 AT3 BO BIOR GP NYU Relcode Short Code X-1	Remington Rand Remington Rand Remington Rand Remington Rand Remington Rand Remington Rand Remington Rand Remington Rand New York University Remington Rand Remington Rand Remington Rand	2 1,2	X X X X X X X X X X X	X X X X X X X X X X X		X X X X X X X X X X X	May 52 Jan. 53 Aug. 53 Apr. 56 Jun. 56 Dec. 56 Apr. 55 Jan. 55 Feb. 54 Apr. 56 Feb. 51 Jan. 56	S1 S1 S1 S1 S1 S2 — S2 — — — —	S S S S S S — S — — — —	1 1 1 1 2 2 1 1 1 1 1 1	X	Fortran-Like, Output To A3, I + II Runs on Univac I + II Primarily Business Data-Processing For Expert Programmers Runs on Univac I + II
			1	X	X		X						
I.B.M. 702	Autocoder Assembly Omnicode Script	I.B.M. I.B.M. G.E. Hanford G.E. Hanford	2 1	X X X X	X X X X		X X X X	Apr. 55 Jun. 54 Propos Jul. 55	— — S2 S1	S — S S	1 1 2 1		May be Assembled on Acctg. Equip. Super-Script
			1	X	X		X						
I.B.M. 701	Acom Bacalc Douglas Dual 607 Flop Jcs 13 Kompiler 2 Naa Assembly Pact I Queasy Quick Seesaw Shaco So 2 Speedcoding	Allison G.M. Boeing, Seattle Douglas Sm Los Alamos Los Alamos Lockheed Calif. Rand Corp. Ucl Livermore N. Amer. Aviation (See Pact Group) Nots Inyokern Douglas Es Los Alamos I.B.M. I.B.M.	1,2 1	X X X X X X X X X X X X X X X	X X X X X X X X X X X X X X X		X X X X X X X X X X X X X X X	Jul. 55 May 53 Mar. 53 Sep. 53 Mar. 53 Dec. 53 Jun. 55 Jan. 55 Jun. 53 Apr. 53 Apr. 54 Apr. 53	— — — — — — S2 — S — S S S S S1	S S S — S S — S — S S S S S	1 1 1 1 1 1 1 — 0 1 1 1	X	Modification of 607 Also Assembles 704 Programs Most Programs run on Pact IA Double Quick for Dbl Prec
			1	X	X		X						
Datatron	Cic Dot I Uglic	Purdue Univ. Electro Data United Gas Corp.	2				X X	Incomp	S2	S	1	X	
I.B.M. 650	Ades II Bacalc Baltac Bell Cic Druco I Flair Miltac Omnicode Sir Soap I Soap II Speed coding Spur	Naval Ordnance Lab Boeing, Seattle M.I.T. Bell Tel. Labs Carnegie Tech I.B.M. Lockheed Msd, Ga. M.I.T. G.E. Hanford I.B.M. I.B.M. I.B.M. Redstone Arsenal Boeing, Wichita	2 1,2 2 1	X X X X X X X X X X X X X X X	X X X X X X X X X X X X X X X		X X X X X X X X X X X X X X X	Feb. 56 Aug. 56 Jan. 56 Aug. 55 Dec. 56 Sep. 54 Feb. 55 Jul. 55 Dec. 56 May 56 Nov. 55 Nov. 56 Sep. 55 Aug. 56	S2 — S1 S1 S2 S S S1 S1 — — (M) S1 (M)	S S S S S S S S S S S S S S S	1 1 2 0 1 0 2 2 2 2 2 0 1	X X X	Must Process on 701 For all 650 models Output Processed by soap For all 650 models Must Process on drum 702 Operates with soap I, II For all 650 models + variations Resembles 701 speedcoding For all 650 models
			1	X	X		X						
Whirlwind	Algebraic Comprehensive Summer ses.	M.I.T. M.I.T. M.I.T.		X X X	X X X		X X X	Nov. 52 Jun. 53	S2 S1 S1	S S S	1 1 1	X	
Midac	Easiac Magic	Univ. of Michigan Univ. of Michigan		X X	X X		X X	Aug. 54	S1 S1	S S	1 1		
Burroughs Ferranti Illiac Johnniac Norc Seac	Transcode Dec order input Easy fox Base 00	Burroughs Lab Univ. of Toronto Univ. of Illinois Rand Corp. Naval Ordnance Lab Natl. Bur. Stds.	1 1	X X X X X	X X X X X		X X X X X	Aug. 54 Sep. 52 Oct. 53 Feb. 56	M1 S1 S1 M2	S S S M	1 1 1 1		

Pact Group Contains Douglas Sm, Es, Lb, Lockheed Cal, Nots, N. Amer., Rand

in the dictionary, much as the FORTRAN coding system has already been translated for use by the French.

● Much more intelligence will be built into the translators. The program may make a reasonable guess about the intent of the programmer when some omission or violation of language rules occurs. Learning procedures will be incorporated so that the translator may take advantage of statistics of previous operation to improve the program which it creates. This may extend as far as a form of creativity where the data processor may originate programs merely from the statement of the problem to be solved, rather than from a given procedure for this solution.

4. Flow-chart construction for procedures will be automatic, having been determined in their linkages by the before-and-after conditions for the components of procedure. Today, Monte Carlo techniques are already being used to determine frequencies of occurrence for various logical branchings in the procedure, with resultant optimization of the program by taking these factors into consideration. The efficient usage of the various and graded components of the computer system will be automatic; the programmer considers an infinite machine, which the processor arranges as it knows its own limitations and capabilities.

5. Not only will programs be created which write programs to do actual computation, but other levels

EXPLANATION OF SYMBOLS

CODE 1) Recommended for this particular computer, sometimes for volume of usage or interchange, or for lack of better.

2) Common language to more than one computer.

SYSTEM TYPE (See "Terms Used In Automatic Coding")

INDEXING M) Actual index registers or B-boxes exist in machine hardware.

S) Simulated in the synthetic language.

1) Limited form, either stepped unidirectionally or by 1 word only, having certain registers applicable to only certain address positions, or not compound (by combination).

2) General form, where any address may be indexed by any 1 or a combination of registers, which may be freely incremented or decremented by any amount. Have associated loop termination instructions.

FLOATING PT. M) Inherent in machine hardware.

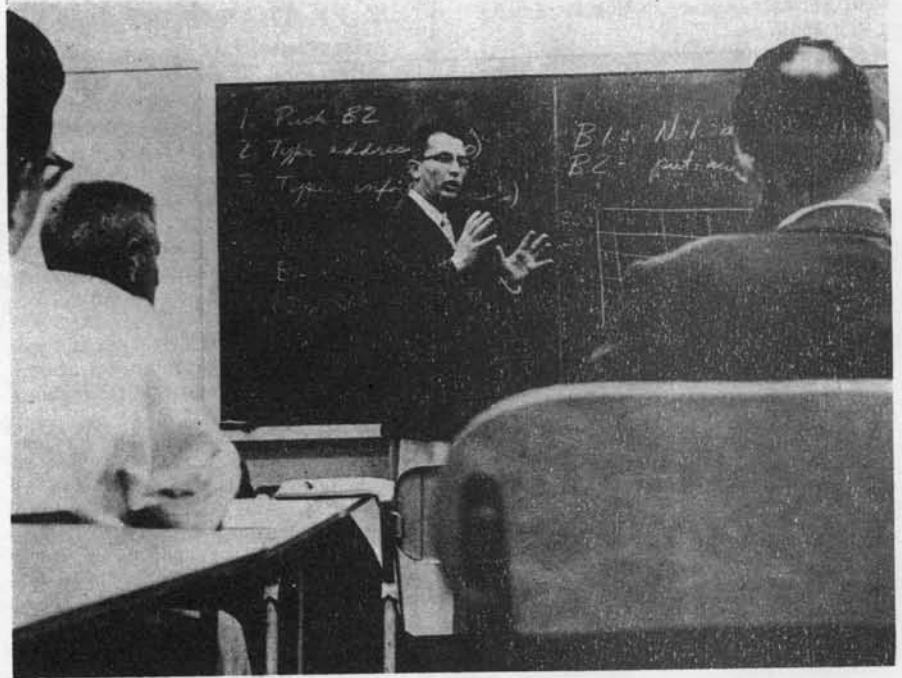
S) Simulated in the synthetic language.

SYMBOLISM 0) None.

● Limited—either regional, relative or exactly computable.

2) Fully descriptive, where a word or symbol combination may be descriptive of the contents of the assigned storage.

ALGEBRAIC A single continuous algebraic formula statement may be made. The processor contains mechanisms for applying the associative and commutative laws of algebra to form the operative program.



will be superimposed on these. Bootstrap methods are being considered which will allow even the person who develops the processor to have a good portion of his work done automatically by reference to and through previous work.

6. The actual operation of the computer will be under control of an integrated portion of the processor known as a supervisory routine. In some instances the program will not have been created by the processor prior to execution time, but will be created during a break in execution time under orders of this supervisory routine, which detects that no method is in existence in the program for a particular contingency. Although these supervisors will be on magnetic tape for a while, it is envisioned that they will be buried in the machine hardware eventually, to be improved by replacement like any other component.

FUTURE COMPUTER SYSTEMS

Future computer operation, which strongly influences the design of the programming languages, has some vitally interesting possibilities. In this glimpse, the picture presented here is dependent upon three axioms:

▶ Faster computers always lower the dollar cost per problem solved, but not all companies will be able to afford the high prices of the next gen-

eration of super-computers. They simply may not have enough problems to load one.

▶ Producing a spectrum of machines is a tremendous waste of effort and money on the part of both the manufacturers and the users.

▶ Availability of a huge central computer can eliminate the discrete acquisition of multiple smaller computers, homogenize the entire structure of usage, and allow a smaller and more numerous class of user into the act, thus tapping a market many times the size of presently projected with current practice in computer access.

Assuming the availability of practical micro-wave communication systems, it is conceivable that one or several computers, much larger than anything presently contemplated, could service a multitude of users. They would no longer rent a computer as such; instead they would rent input-output equipment, although as far as the operation will be concerned they would not be able to tell the difference. This peripheral equipment would perhaps be rented at a base price plus a variable usage charge on a non-linear basis. The top-most level of supervisory routine would compute these charges on an actual usage basis and bill the customer in an integrated operation. These program features are, of course, recognizable to operations research

people as the Scheduling and Queuing Problems.

Using commutative methods, just as motion pictures produce an image every so often for apparent continuity, entire plant operations might be controlled by such super-speed computers.

These future hardware capabilities (and few competent computer manufacturers will deny the feasibility, even today, of super-speed and inter-

leaved programs) demonstrate a pressing need for an advanced common language system so all users concerned can integrate their particular operations into the complex of control demanded by an automated future.

Just one last prediction—the engineer who is going to be at the top of his profession in the years to come had better become a computer expert, too.

A WORKING GLOSSARY OF SOME AUTOMATIC CODING TERMS

AUTOMATIC CODING — Systems which allow programs to be written in a synthetic language especially designed for problem statement, which the processor translates to presumably the most efficient final machine language code for any given computer. Usually such a system will examine one entry at a time and produce some amount of coding which is determined by that entry alone.

AUTOMATIC PROGRAMMING — Systems further up the scale of complexity, where the computer program helps to plan the solution of the problem as well as supply detailed coding. Such systems usually examine many entries in parallel and produce optimized coding where the result of any single entry depends upon its interactions with other entries.

ASSEMBLER — An original generic name for a processor which converts, on a one-for-one basis, the synthetic language entries to machine instructions. This process occurs prior to the actual execution of the working program. It is a one-level processor which can combine several sections or different programs into an integrated whole, meanwhile assigning actual operation codes and addresses to the instructions.

COMPILER — Generally a more powerful processor than the assembler, although there is a great deal of confusion and overlapping of usage between the two terms. The compiler is capable of replacing single entries with pre-fabricated series of instructions or sub-routines, incorporating them in the program either in-line or in predetermined memory positions with standard mechanisms for entry from and exit to the main routine. Such compound entries are sometimes called "macro-instructions." The basic principle of a compiler is to translate and apply as much intelligence as possible ONCE before the running of the program, to avoid time-consuming repetition during execution. It produces an expanded and translated version of the original, or source program. According to the ACM, a compiler may also produce a secondary synthetic program for interpretation while running.

FLOATING POINT — Number notation whereby a number X is represented by a pair of numbers Y and Z in the form: $X = Y \cdot B^Z$ where B is the number base used. For floating decimal notation the base B is 10; for floating binary the base is 2. The quantity Y is called the fraction or mantissa, and in the best notation $O = Y - 1$. Z is an integer called the exponent or power.

GENERATOR — A generator is a program which writes other programs, usually on a selective basis from given parameters and skeletal coding. It may be either a character-controlled generator, so that it selects among several options according to a preset character matrix, or a pure generator, which writes a program on the basis of calculations which it makes from the input data. Almost all assemblers and compilers have generating elements in some form.

INDEX REGISTER — A register whose contents are used to automatically modify addresses incorporated in instructions just prior to their execution, the original instruction remaining intact and unmodified in memory. It may either be built in the hardware and circuitry of the computer or be simulated by the program. The original unmodified addresses are termed presumptive, the modified addresses are termed effective.

INTERPRETER — In contrast to an assembler, compiler or generator, a source program designed for interpretation is converted to an object program which is not in machine language when run. The interpreter itself is an executive program which must always be used in conjunction with the object program, and always resides in high-speed memory during execution of the problem. The object program is always subservient to the interpreter, which fabricates from the incompletely converted instruction the necessary parts of machine language instructions just before they are executed. Each entry in the interpretive language usually calls for the use of a complete subroutine, which is used again and again by filling in the missing parts of certain of its instructions from the object instruction.

MACHINE LANGUAGE — The wired-in circuitry language at a low logical level which is intelligible to the computer. It should seldom be used to code problems because of the difficulties of usage at this level and the tendency to error.

OBJECT PROGRAM — The output of the processor when it has translated the source program to either machine language or a second level synthetic language.

PROCESSOR — Also called a translator, this is a computer program which produces other programs, in contrast to programs which are working and produce answers.

SOURCE PROGRAM — The original program written to solve problems and produce answers, phrased in the synthetic language.

1 April 1957

Mr. Robert W. Bemer
Programming Research Department
International Business Machines Corp.
590 Madison Avenue
New York 22, New York

Dear Bob,

I am utterly delighted by your article in "Automatic Control". I'm trying to buy reprints (lots) so I can spread it all over. Of course, I liked the first paragraph, but best of all is your definition of "machine language". This does my heart good. So many many thanks from all of us for such a grand job.

I still don't have any B-Zero (now called Flow-matic by the Sales Department) manuals. But I'm sending

1. Flow chart and pseudo-code for an inventory run.
2. A complete report on a three-way merge run including file designs, flow charts, and final program.
3. The "hand-out" material used in the first course last week.

It will at least give you an overall view. We are busily engaged in adding eleven more generators and improving the overall operating time. In the meantime, we are using it steadily. Public release, with manuals, is now scheduled for June and/or July. This will be a first version--- we are improving B-1.

Again many thanks for the terrific article.

Sincerely,

Grace

Dr. Grace Murray Hopper

P.S. Could you send me Roy Goldfinger's address?

OXFORD UNIVERSITY COMPUTING LABORATORY
PROGRAMMING RESEARCH GROUP

45 BANBURY ROAD
OXFORD
OX2 6PE

Telephone OXFORD 58086

1st May, 1974.

Professor D. E. Knuth,
Stanford University,
Computer Science Department,
Stanford,
California 94305,
U.S.A.

Dear Don,

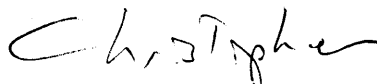
The paper I wrote called 'Time Sharing in Large Fast Computers' was read at the first (pre IFIP) conference at Paris in 1960. It was mainly about multi-programming (to avoid waiting for peripherals) although it did envisage this going on at the same time as a programmer was debugging his program at a console. I did not envisage the sort of console system which is now so confusingly called time sharing. I still think my use of the term is the more natural.

I am afraid I am so rushed at the moment, being virtually alone in the PRG and having just moved house, that I have no time to look up any old notes I may have. I hope to be able to do so while settling in and if I find anything of interest I will let you know.

Don't place too much reliance on Halsbury's accuracy. He tends to rely on memory and get the details wrong. But he was certainly right to say that in 1960 'time sharing' as a phrase was much in the air. It was, however, generally used in my sense rather than in John McCarthy's sense of a CTSS-like object.

Best wishes,

Yours sincerely,



C. Strachey
Professor of Computation
University of Oxford

Time sharing

January 1, 1959

TO: Professor P. M. Morse
 FROM: John McCarthy
 SUBJECT: A Time Sharing Operator Program for our Projected IBM 709

1. INTRODUCTION

This memorandum is based on the assumption that MIT will be given a transistorized IBM 709 about July 1960. I want to propose an operating system for it that will substantially reduce the time required to get a problem solved on the machine. Any guess as to how much of a reduction would be achieved is just a guess but a factor of five seems conservative. A smaller factor of improvement in the amount of machine time used would also be achieved.

The proposal requires a complete revision in the way the machine is used, will require a long period of preparation, the development of some new equipment, and a great deal of co-operation and even collaboration from IBM. Therefore, if the proposal is to be considered seriously, it should be considered immediately. I think the proposal points to the way all computers will be operated in the future and we have a chance to pioneer a big step forward in the way computers are used. The ideas expressed in the following sections are not especially new, but they have formerly been considered impractical with the computers previously available. They are not easy for computer designers to develop independently since they involve programming system design much more than machine design.

2. A QUICK SERVICE COMPUTER

Computers were originally developed with the idea that programs would be written to solve general classes of problems and that after an initial period most of the computer time would be spent in running these standard programs with new sets of data. This view completely underestimated the variety of uses to which computers would be put. The actual situation is much closer to the opposite extreme, wherein each user of the machine has to write his own program and that once this program is debugged, one solves the problem. This means that the time required to solve the problem consists mainly of time required to debug the program. This time is substantially reduced by the use of better programming languages such as Fortran, LISP (the language the Artificial Intelligence Group is developing for symbolic manipulations) and

COMIT (Yngve's language). However, a further large reduction can be achieved by reducing the response time of the computation center.

The response time of the MIT Computation Center to a performance request presently varies from 3 hours to 36 hours depending on the state of the machine, the efficiency of the operator, and the backlog of work. We propose, by time sharing, to reduce this response time to the order of 1 second for certain purposes. Let us first consider how the proposed system looks to the user before we consider how it is to be achieved.

Suppose the average program to be debugged consists of 500 instructions plus standard subroutines and that the time required under the present system for an average debugging run is 3 minutes. This is time enough to execute 7,000,000 704 instructions or to execute each instruction in the program 14,000 times

Most of the errors in programs could be found by single-stepping or multiple stepping the program as used to be done. If the program is debugged in this way the program will usually execute each instruction not more than 10 times, $1/1400$ as many executions as at present. Of course, because of slow human reactions the old system was even more wasteful of computer time than the present one. Where, however, does all the computer time go now?

At present most of the computer time is spent in conversion, (SAP-binary, decimal-binary, binary-decimal, binary-octal) and in writing tape and reading tape and cards.

Why is so much time spent in conversion and input output.

1. Every trial run requires a fresh set of conversions.
2. Because of the slow response time of the system it is necessary to take large dumps for fear of not being able to find the error. The large dumps are mainly unread, but nevertheless, they are necessary. To see why this is so, consider the behavior of a programmer reading his dump. He looks at where the program stopped. Then he looks at the registers containing the partial results so far computed. This suggests looking at a certain point in the program. The programmer may find his mistake after looking at not more than 20 registers out of say 1000 dumped, but to have predicted which 20 would have been impossible in advance and to have reduced the 1000 substantially would have required cleverness as

subject to error as his program. The programmer could have taken a run to get the first register looked at, then another run for the second, etc., but this would have required 60 hours at least of elapsed time to find the bug according to our assumptions and a large amount of computer time for repeated loading and re-runnings. The response time of the sheet paper containing the dump for any register is only a few seconds which is ok except that one dump does not usually contain information enough to get the entire program correct.

Suppose that the programmer has a keyboard at the computer and is equipped with a substantial improvement on the TXO interrogation and intervention program (UT3) (The improvements are in the direction of expressing input and output in a good programming language). Then he can try his program, interrogate individual pieces of data or program to find an error, make a change in the source language and try again.

If he can write program in source language directly into the computer and have it checked as he writes it he can save additional time. The ability to check out a program immediately after writing it saves still more time by using the fresh memory of the programmer. I think a factor of 5 can be gained in the speed of getting programs written and working over present practice if the above-mentioned facilities are provided. There is another way of using these facilities which was discussed by S. Ulam a couple of years ago. This is to use the computer for trial and error procedures where the error correction is performed by a human adjusting parameter.

The only way quick response can be provided at a bearable cost is by time-sharing. That is the computer must attend to other customers while one customer is reacting to some output.

3. THE PROBLEM OF A TIME-SHARING OPERATOR SYSTEM

I have not seen any comprehensive written treatment of the time-sharing problem and have not discussed the problem with anyone who had a complete idea of the problem. This treatment is certainly incomplete and is somewhat off the cuff. The equipment required for time-sharing is the following

1. Interrogation and display devices (flexowriters are possible but there may be better and cheaper)
2. An interrupt feature on the computer -- We'll have it.
3. An exchange to meditate between the computer and the external devices. This is the most substantial engineering problem, but IBM may have solved it.

In general the equipment required for time-sharing is well understood, is being developed for various advanced computers e.g. Stretch TX2, Metrovich 1010, Edsac 3. I would not be surprised if almost all of it is available with the transistorized 709. However, the time-sharing has been worked out mainly in connection with real-time devices. The programs sharing the computer during any run are assumed to occupy prescribed areas of storage, to be debugged already, and to have been written together as a system. We shall have to deal with a continuously changing population of programs, most of which are erroneous.

The major problems connected with time-sharing during program development seem to be as follows:

1. Allocating memory automatically between the programs. This requires that programs be assembled in a relocatable form and have a preface that enables the operator program to organize the program, its data, and its use of common subroutines.
2. Recovery from stops and loops. The best solutions to these problems requires
 1. Changing the stop instructions to trap instructions. This is a minor modification to the machine. (At least it would be minor for the 704.)
 2. Providing a real time alarm clock as an external device.
 3. Preventing a bad program from destroying other programs. This could be solved fairly readily with a memory range trap which might not be a feasible modification. Without it, there are programming solutions which are less satisfactory but should be good enough. These include:
 1. Translations can be written so that the programs they produce cannot get outside their assigned storage, areas. A very minor modification would do this to Fortran.
 2. Checksums can be used for machine language programs.
 3. Programming techniques can be encouraged which

make destruction of other programs unlikely.

4. There is an excessive tendency to worry about this point. The risk can be brought down to the present risk of having a program ruined by operator or machine error.

4. SUMMARY

1. We may be able to make a major advance in the art of using a computer by adopting a time-sharing operator program for our hoped-for 709.

2. Such a system will require a lot of advances preparation starting right away.

3. Experiments with using the flexo connection to the real-time package on the 704 will help but we cannot wait for the results if we want a time sharing operator program in July 1960.

4. The co-operation of IBM is very important but it should be to their advantage to develop this new way of using a computer.

5. I think other people at MIT than the Computation Center staff can be interested in the systems and other engineering problems involved.