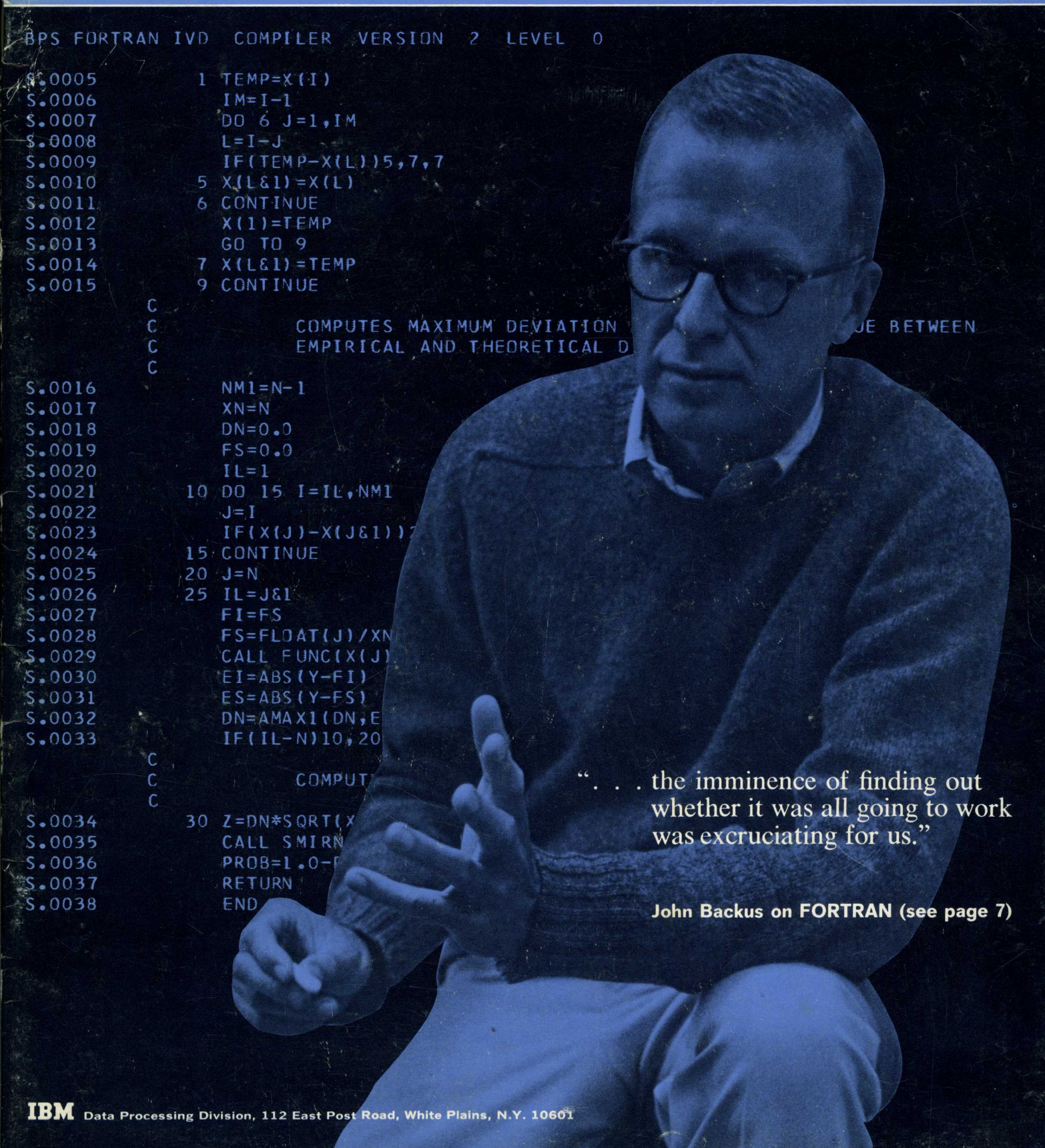
IBM

computing report

For the Scientist and Engineer

November 1966, Vol. II, No. 4



"... the imminence of finding out whether it was all going to work was excruciating for us."

John Backus on FORTRAN (see page 7)

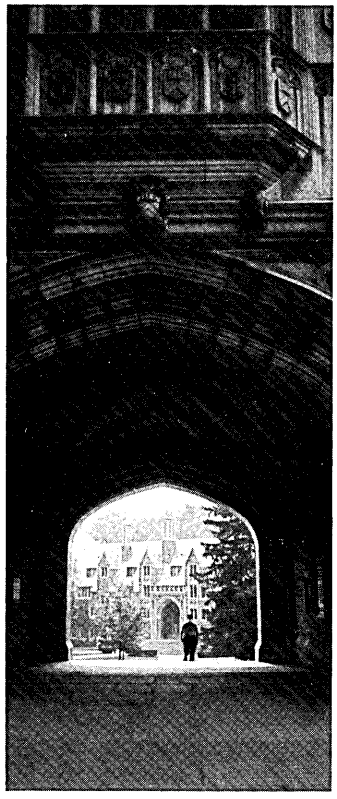
IBM Data Processing Division, 112 East Post Road, White Plains, N.Y. 10601

Contents:

Special report from Princeton

The computer has notably changed the pace and perspective of scholarly pursuits. At Princeton University the impact has been pronounced. See *Big machine on campus,* a special reprint, p. 11.

Picture credits: Cover, 9, 10, 19, P. Stackpole; 2, 4 (left), 12, 13, 16, 21, A. W. Hummers; 4 (right), Bell Telephone Laboratories, Inc.; 28, IBM World Trade Corp.



November, 1966, Vol. II, No. 4
Computing Report
for the Scientist and Engineer

Editor: H. C. Welborn
Associate Editor: D. T. Sanders

Technical Editor: R. F. Steinhart
Editorial Assistant: K. M. Joyce
Circulation: R. W. Haigh

Published by
Data Processing Division
International Business Machines Corporation
White Plains, New York 10601

FFT: shortcut the long way

New computer algorithm sharply reduces calculations, opens new avenues of scientific investigation

Over three years ago, Richard L. Garwin of the IBM Watson Laboratory at Columbia University was faced with carrying out a large Fourier transform problem in a spin calculation for solid helium 3. Even with available computer programs for Fourier analysis, he saw that he would need at least four hours on a high-speed computer to get his answer. While the Fourier transform is an indispensable mathematical tool in many scientific disciplines, it was evident that so many thousands of complex multiply-add operations were involved that even the modern computer couldn't do the job economically.

Not long after that, Dr. Garwin discussed his problem with John W. Tukey of Princeton University and Bell Telephone Laboratories during a pause in a meeting of the President's Science Advisory Committee, of which both were members.

Jotting down notes on a yellow scratch pad, Dr. Tukey outlined a computer algorithm which he thought would handle Fourier transforms far faster than the existing programs. Recognizing the potential and importance of the algorithm, Dr. Garwin took the notes to IBM's James W. Cooley at the IBM Thomas J. Watson Research Center at Yorktown Heights. A program development project was started, and in April, 1965 *Mathematics of Computation* published the Cooley-Tukey paper, "An Algorithm for the Machine Calculation of Complex Fourier Series."

Since then, the FFT computer algorithm—fast Fourier transform—has demonstrated an extraordinary range of application both in performing existing Fourier analysis tasks far more efficiently and in opening up new avenues that were once considered impracticable. In reducing computer time, the proportionate saving with the FFT method increases rapidly as the number of coefficients to be calculated increases. At the same time, the FFT is a rare shortcut: it is an exact algorithm that is actually more accurate than direct computation, which is more subject to round-off error.

In time and frequency

Fourier transforms are widely used in representing complex periodic functions in both the time and frequency domains. Thus any finite periodic function of time X(t) with period (N) can be expressed in this manner:

$$X(t) = \sum_{n=0}^{N-1} C(n) e^{2\pi j \left(\frac{nt}{N}\right)},$$

where C(n) is a function of frequency (n) given by

$$C(n) = \frac{1}{N} \sum_{t=0}^{N-1} X(t) e^{-2\pi j \left(\frac{nt}{N}\right)}.$$

The functions X(t) and C(n) are finite Fourier transforms of each other, and X(t) can be computed from C(n), and vice versa, as the user requires, as indicated above.

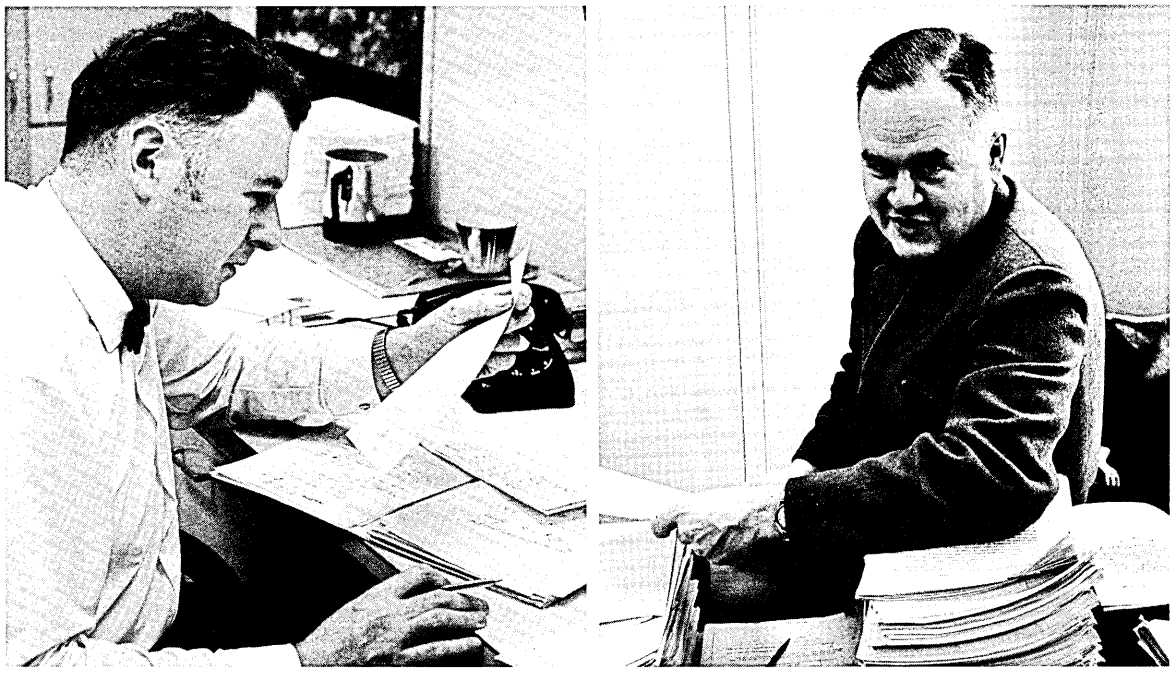
The value of the FFT algorithm in computing these transforms can be expressed in terms of the period (N) which is also the number of terms in each Fourier series. Direct computation of either X(t) or C(n) as defined above requires N^2 complex multiply-add operations. With FFT, only $N\log_2 N$ operations are needed. Thus, the ratio of computing time (FFT to direct) is given by

$$\frac{N \log_2 N}{N^2} \quad \frac{\log_2 N}{N}.$$

For a relatively simple function with N=4, therefore, FFT requires $\log_2 4/4 = \frac{1}{2}$ the number of operations to solve. For a longer period of N=1024, for example, fewer than 1/100th the number of operations are needed with FFT.

FFT and Rat Island

Because most practical applications of Fourier transforms need thousands of terms or more, the savings in computer time are great. At the American Geophysical Union meeting in Washington, D. C., last April, Lee Alsop, an IBM scientist at Lamont Geological Observatory, reported a direct



FFT ORIGINATORS Dr. James W. Cooley (left) and Dr. John W. Tukey.

comparison of the FFT algorithm and a conventional Fourier transform program in analyzing a strain seismogram of the large Rat Island, Alaska, earthquake of 1965.

The record of the Rat Island earthquake in Figure 1 consisted of 2048 numbers representing longitudinal displacement at instants equally spaced over a period of about 13.5 hours. The data were analyzed with both programs and the frequency spectrum traced out by an automatic x-y plotter. In Figure 2, the two resulting traces of the longitudinal component as a function of period are identical. The difference in computer time, however, was striking: 1567.8 seconds for the rather slow conventional program, as compared to only 2.4 seconds for FFT.

Having verified the far greater speed of FFT, Lee Alsop then ran a further check on its accuracy with a time series generated from a sum of seven sines and cosines of harmonics of a base frequency. Computer analysis times were 467.4 seconds for the conventional program and 1.2 seconds for FFT. By then computing back again from the transform, the results of each program were compared to the original series. It was found that FFT was slightly more accurate, even though the particular conventional program used was specifically developed for high accuracy (somewhat at the expense of speed).

A shortcut the long way

Convolution, an extremely useful mathematical technique in time series analysis, is performed by multiplying two time sequences $X_1(t)$ and $X_2(t)$, one lagged behind the other by the time t, as follows:

$$\frac{1}{N}\sum_{\tau=0}^{N-1} X_1(t-\tau)\, X_2(\tau).$$

Direct computation of a convolution in the time domain, as might be surmised from these equations, usually requires a massive number of arithmetic operations (N for each point in time) and so a lot of computer time.

With the speed of the FFT program, it has been found possible to reduce this effort drastically by taking the long way—in a manner somewhat like the use of logarithms in performing complex multiplications or divisions. Instead of direct multiplication of the two time series, the transforms of $X_1(t)$ and $X_2(t)$ are first computed, multiplied in the frequency domain, and then the result is transformed back into the time domain. In a convolution, for example,

$$X_1(t) \text{—(FFT)}\rightarrow C_1(n)$$
$$X_2(t) \text{—(FFT)}\rightarrow C_2(n)$$
$$\frac{1}{N}\sum_{\tau=0}^{N-1} X_1(t-\tau)\, X_2(\tau) \leftarrow \text{(FFT)—} C_1(n)\, C_2(n).$$

The procedure is similar with covariances. Prior to the availability of the FFT algorithm, this technique was never considered seriously.

In a paper presented at the 1966 Spring Joint Computer Conference, Thomas G. Stockham, Jr., of MIT, reported comparative convolution times for various values of N. FFT took longer than the direct method at low periods, but at N=32 it was

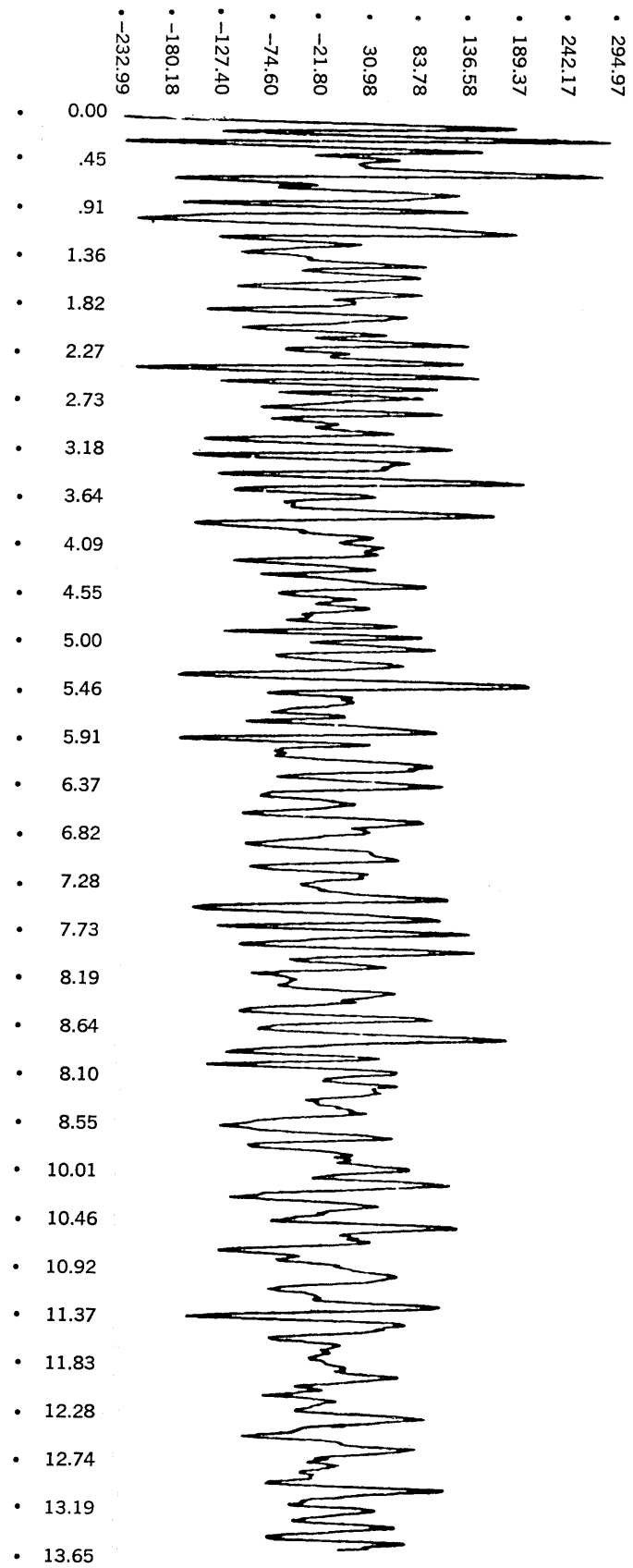

−232.99
−180.18
−127.40
−74.60
−21.80
30.98
83.78
136.58
189.37
242.17
294.97
0.00
.45
.91
1.36
1.82
2.27
2.73
3.18
3.64
4.09
4.55
5.00
5.46
5.91
6.37
6.82
7.28
7.73
8.19
8.64
8.10
8.55
10.01
10.46
10.92
11.37
11.83
12.28
12.74
13.19
13.65

Fig. 1

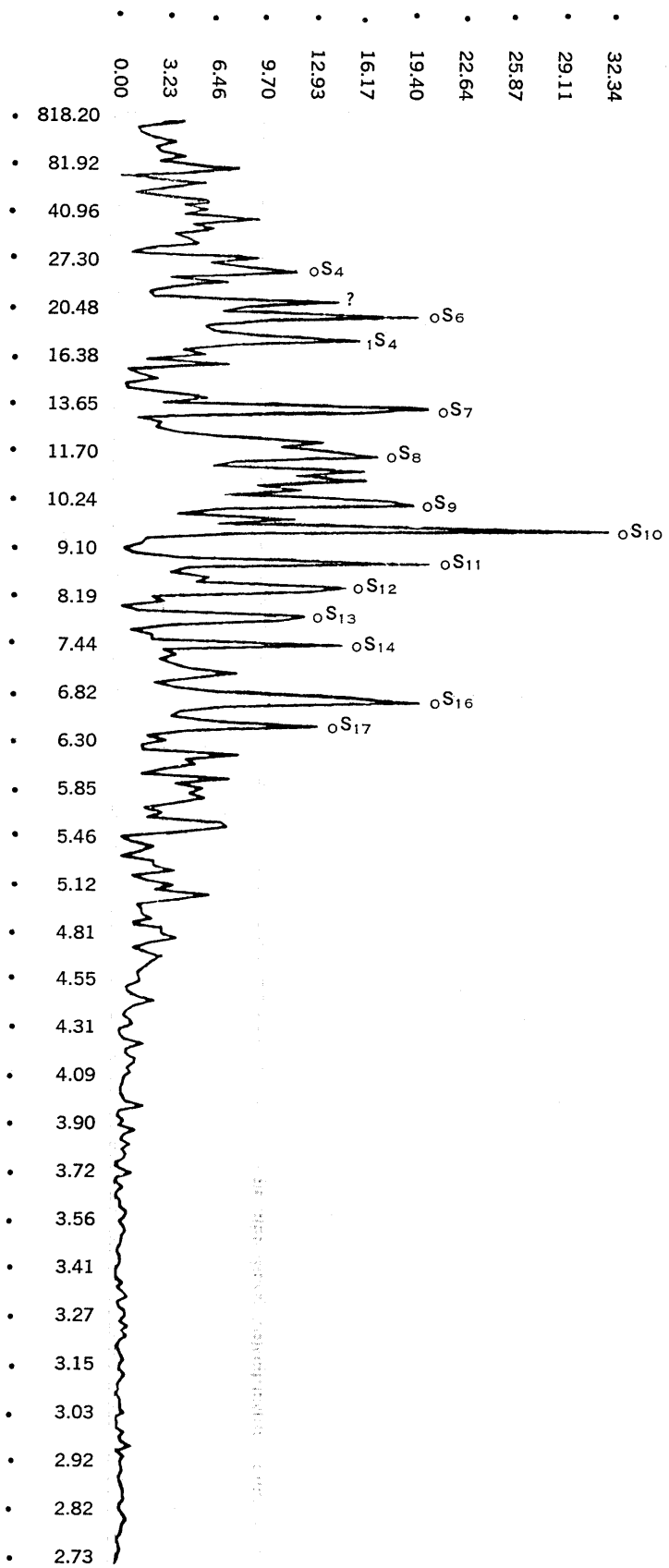

0.00
3.23
6.46
9.70
12.93
16.17
19.40
22.64
25.87
29.11
32.34
818.20
81.92
40.96
27.30
20.48
16.38
13.65
11.70
10.24
9.10
8.19
7.44
6.82
6.30
5.85
5.46
5.12
4.81
4.55
4.31
4.09
3.90
3.72
3.56
3.41
3.27
3.15
3.03
2.92
2.82
2.73
oS4
?
oS6
↑S4
oS7
oS8
oS9
oS10
oS11
oS12
oS13
oS14
oS16
oS17

Fig. 2

a third faster, at N=256 six times faster, and at N=4096 an estimated 80 times faster.

Digital signal enhancement

Convolution is a widely used method of enhancing signals through digital filtering. Signals can be far more economically "cleaned up" in this manner in the frequency domain than in the time domain. In marine exploration seismology, for example, reverberation noise—an ambient ringing that interferes with analysis—can be suppressed with the help of FFT. The time record of the reflected pressure wave is transformed to the frequency domain, where the ringing stands out as spikes. The spikes of the reverberation frequency and its harmonics are suppressed, and retransformation yields an enhanced signal.

Image enhancement in character recognition systems, now based on analog methods, can be accomplished digitally with FFT. In what is essentially a two-dimensional version of seismological applications, image noise can be effectively removed by digital filters. Spatial filtering with FFT is now being evaluated, for example, for emphasizing characteristics of fingerprints.

Other FFT applications

Other potential applications of FFT include real-time digital analysis of speech. The time-frequency transformation can be accomplished while the words are being spoken. Digital Vocoders—voice encoders, presently analog—are then practicable.

Optical applications of FFT are a natural. Using it optical systems can be efficiently simulated on a digital computer, since a lens or diffraction grating essentially performs a Fourier transform on the incoming light. In part, imperfect images can be much improved by digitizing and filtering using FFT.

The FFT algorithm is already being applied in numerical spectroscopy in physics and chemistry. In power spectrum analysis, FFT makes possible the direct estimation of the power spectra of noisy signals, whereas it was formerly necessary to determine the covariance function first. Other applications are in planetary astronomy, reentry physics, and detecting underground nuclear tests.

A final irony

In their paper, Cooley and Tukey noted that I. J. Good had described the basic mathematical procedure in 1958, although he had not considered its application to the performance of Fourier transform by computer. After publication of the FFT paper, an interested reader pointed out that Danielson and Lanczos had published the fast Fourier transform technique way back in 1942—including a discussion of the saving in number of operations for small N, at that time being performed on a desk calculator.

Certainly, it would then be fitting to conclude an article on the early life of FFT with a report on its performance on the He^3 spin problem, which was responsible for converting a 23-year-old, widely unknown mathematical idea into a generally useful computer program. Unfortunately, Dr. Garwin didn't wait for FFT—he found another way to handle his spin calculation. He is, however, fond of commenting that the factor by which FFT reduces the effort involved in Fourier transform is precisely that involved in replacing counting with arithmetic.

NEW PROGRAMS WITH FFT

The FFT algorithm has been incorporated by Dr. Cooley into a 7090 program now available from the IBM Program Information Department. Called Harmonic Analysis Subroutine, it may be requested through local IBM sales offices under File Number 7090-SDA-3425. As a contributed program, it is not product-tested, maintained, or otherwise supported by IBM.

More recently, the FFT was announced as part of the new System/360 Scientific Subroutine Package, Version 2 (see *Computing Report,* April, 1966, for a description of the original version). Version 2, which will be available during the first quarter of 1967, contains more than 70 additional FORTRAN IV subroutines—some entirely new, some with improved features of existing capabilities.

A partial list of the Version 2 subroutines follows: Solution of simultaneous, first-order, ordinary, differential equations by three methods:

Runge-Kutta method with given initial values.

Hamming Modified Predictor-Corrector method with given initial values.

Adjoint equations method for linear system with linear boundary conditions.

Integration by the following methods: Trapezoid; Simpson's Rule; Hermitian; Gaussian; Gaussian Laguerre; Gaussian Hermite; Generalized Gaussian Laguerre; and Trapezoid with Romberg's extrapolation.

Also, generalized elliptic integrals: polynomial economization; solution of linear least squares problems; roots of a polynomial by Quotient-Difference algorithm; interpolation.

The man behind FORTRAN

John Backus recalls the time when creating FORTRAN meant fun, frenzy and eventual success

Now a decade old, FORTRAN is solidly established as the mother tongue of thousands of scientists and engineers who communicate with computers. Recently, it was stamped with the official imprimatur of the American Standards Association. This action is a tribute to the power, ingenuity and universal use of the language.

It is also a tribute to John W. Backus, now in the middle of a five-year appointment as an IBM Fellow, and presently at the University of California at Berkeley. In 1954 Backus was barely three years out of college and already a "veteran" of programming research at IBM. He was also one of the very few people at that time who were convinced that a programming language could be constructed that would enable computers to produce their own machine code.

Acting on that conviction, Backus conceived the idea of FORTRAN, won company support to pursue it, and headed the project team that worked impossible hours to prove a seemingly implausible theory. Three years and 25,000 lines of machine instruction later, the proof was in. In this interview, John Backus tells the story behind that remarkable effort.

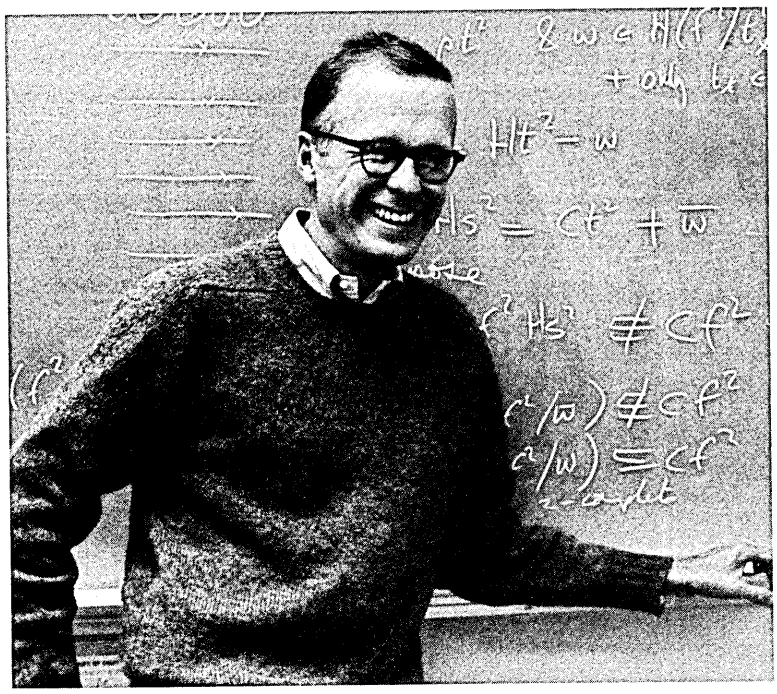
Q. *When you and your colleagues started down the road that led eventually to* FORTRAN, *did you have a pretty fair idea of how to go about it?*

A. Frankly, we didn't have the vaguest idea how the thing would work out in detail. The effort to construct FORTRAN began out of the recognition that essentially the cost of programmers was about equal to the cost of the computer, or even greater. It was clear that what engineers needed was a language they could use to code their own problems. Some work had already been done along this line. Laning and Zierler produced a little program at MIT that would take algebraic expressions and produce code from them. But that's about all it would do. And so it began with a letter to Cuthbert Hurd, my boss at the time, asking if we could embark on this thing. Irv Ziller, who is now IBM director of programming development, and I were the first two people involved, and it gradually grew to about 13 over the next three years. For the first six months we struggled to devise a language that would let you state most of the things you wanted a computer to do for scientific programming. Then we wrote a report and went out and described the project to a number of customer groups and asked them for suggestions. We got some, but essentially it was new to them and sometimes a little overwhelming.

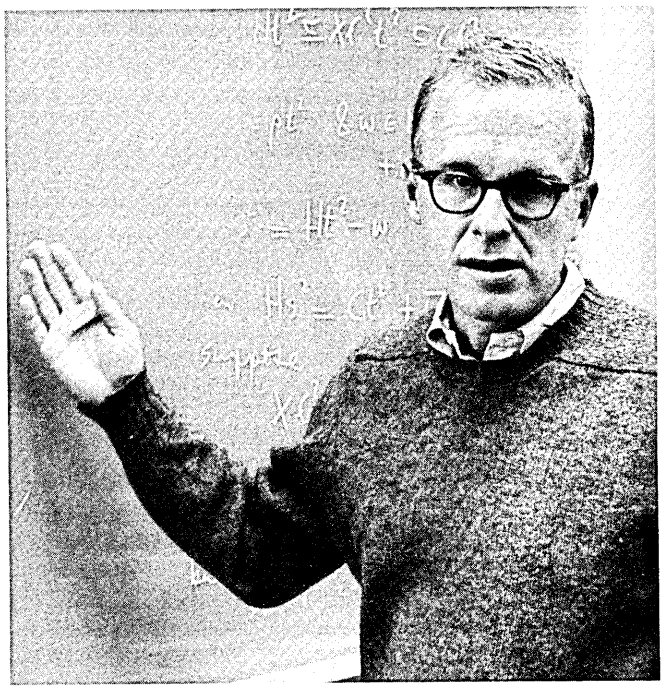
Q. *The basic role of your group in writing* FORTRAN, *it has been said, was to design optimizing techniques. To what degree do you think you succeeded? And what are the compromises that you had to make?*

A. Well, first of all, there are two kinds of optimization you can shoot for. One is to optimize the compiling time, and the other is to try to optimize the programs that are produced. And necessarily, if you try to do one you do the opposite in the other. That is, if you optimize the compiling time, of necessity you produce less than the best programs—less than you could. On the other hand, if you want to produce optimum object programs, why then you have to analyze a lot of things about the material that you're coding. And that takes a lot of compiling time. So that there's a balance that's required—which we didn't really recognize well enough—between compiling time and running time achieved. We struck out simply to optimize the object program . . . the running time . . . because most people at that time believed you really couldn't do that kind of thing. They believed that machine-coded programs would be so terribly inefficient that it would be impractical for very many applications. So, we overdid it. In our attempt to prove that the machine could write as good code in a restricted area, that is, straightforward scientific computing but covering a wide range within that area, we found we had a compiler that did produce, by and large, quite efficient object programs . . . but in some cases it spent longer than would be reasonable. To achieve a small improvement in the object program it would spend quite a bit of compiling time,

```
ALP=X+X
SUM=X*X+Y*Y
N=11-2
GO TO 203
```



"It would throw things out into memory in various places and clobber everybody . . ."



"Each part of the program was written by one or two people who were complete masters of what they did . . ."

which wasn't worth it unless you used the object program very, very many times. If you're going to use an object program for a hundred hours, then it's worthwhile spending an hour compiling it very efficiently. But if you're only going to use it for five minutes, it's pretty depressing to spend 25 minutes compiling.

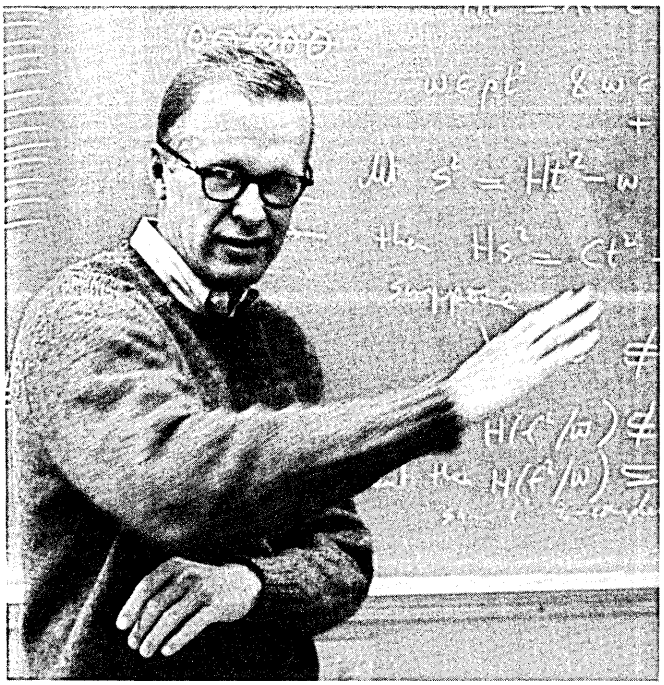
Q. *Could you describe some of the highlights of your early work on* FORTRAN?

A. One of the things that intrigues me is that the whole thing was done for reasons that don't exist anymore. That is, the original work was done to demonstrate that you could compile efficient programs by machine. In those days everybody was writing machine code, and each one felt that he was an enormously ingenious fellow and that a machine couldn't compete with him. But what turned out after three years of work was that you could analyze a lot of the things that programmers did . . . and you could get the machine to do it just as well. I'm sure in many cases that although a programmer could do as well, the limits of his time and energy were such that often he would produce a far worse program because he couldn't do the kind of backbreaking analysis that we, as a group of 13 people, planned for three years and got the machine to perform—often it would do millions of operations in analyzing and optimizing one program.

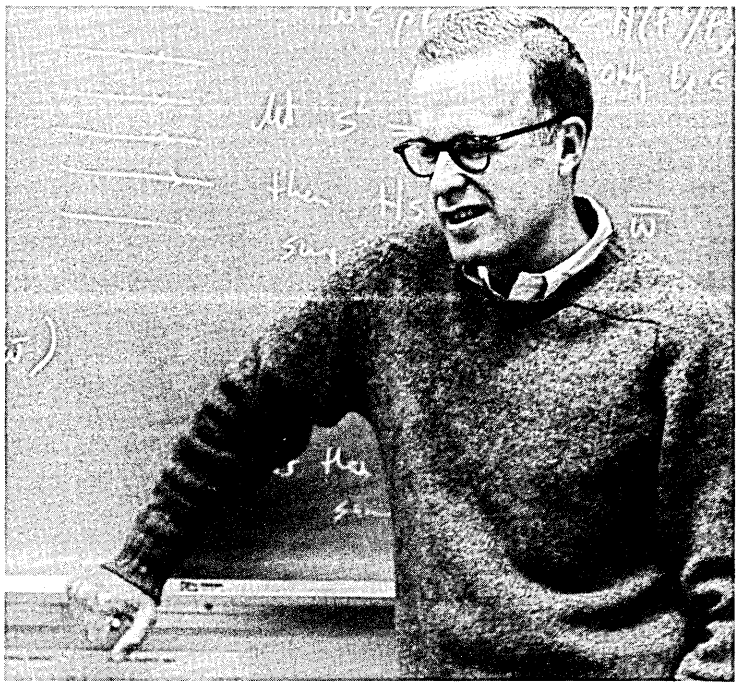
One result we didn't have in mind was this business of having a system that was designed to be utterly independent of the machine that the program was utimately to run on. This came about out of our desire, by and large, not to impose knowledge of the machine on the user. It turned out to be a very valuable capability but we sure didn't have it in mind. It was sort of a lucky fluke that we were strict enough with ourselves to rule out most machine things. There were a number of machine applications that said things like "rewind tape 10," machine dependent things or things that had to do with sense switches and other gadgets. But those have all been eliminated now in FORTRAN IV—fortunately for everybody.

Q. *In his book* The Analytical Engine, *Jeremy Bernstein reports that once* FORTRAN *was operational, the fun and the challenge were lost for you. What are some of the things you recall in the days when it was both challenging and fun?*

A. The fun of doing it was the big thing. When we started out in 1954, we said we were going to have it in 1955 . . . and then in 1955 we said we would be ready in 1956 . . . in 1956 we said that 1957 would be the year . . . and finally we did get it in 1957. We really hoped to get it when the 704 became available, but we were quite a bit



". . . if you're only going to use an object program for five minutes, it's pretty depressing to spend 25 minutes compiling."



"I think a lot more can be done to get computers to do what you want them to do . . ."

late on that. The customers made out fine without us for a while. But it was a very enjoyable experience. The customers who were interested in it were very gracious to us. Once the system was basically programmed with about 25,000 instructions, we had a session out in Los Angeles where we worked 20 hours a day, slept a little bit, and went back to work. It was very exciting because the imminence of finding out whether it was all going to work was excruciating for us. We had a lot of laughs, too. Someone wrote a little program that caused the most mysterious bugs. It would throw things out into memory in various places and clobber everybody, and we had great hilarious sessions laughing over the technical details of these momentous boo-boos.

Q. *How did you organize the work of producing this system?*

A. There was nothing organized about our activities. Each part of the program was written by one or two people who were complete masters of what they did with very minor exceptions . . . and the thing just grew like Topsy. We had basic things that we knew we had to do and each section was roughly plotted out early on, and each one became 10 times more elaborate than we had anticipated at the start. Each group simply decided with the other groups what it needed, then a preceding section group would endeavor to provide it, and so on.

I'd like to mention the people who devised the basic techniques of the system. Harlan Herrick and Pete Sheridan wrote the first section, which produces optimized coding of algebraic expressions. Bob Nelson and Irv Ziller designed the methods used to handle subscripts, arrays and loops that make up section two. The analysis performed in this section and in sections four and five, and the system's input/output facilities, comprised the primary differences between FORTRAN and many early automatic coding systems. Dick Goldberg wrote section three, which does most of the work that was either not feasible or was overlooked in the preceding two sections. It was incredible how many necessary jobs everyone was unaware of until fairly late in the game. Lois Haibt wrote section four, which analyzes how often each part of the program will be used. Sheldon Best, who was then a member of the computer lab at MIT, devised some very beautiful but complex methods for optimizing the use of index registers and incorporated them in section five. Roy Nutt was another key non-IBM participant. He was then with United Aircraft but worked full time with us designing the input-output facilities and the assembly program. Dave Sayre had the crucial job of writing the manual and coordinating things, so that the system worked as he said it would.

Q. *What happened once* FORTRAN *was distributed?*

A. That was a very interesting time. We had the problem of facing the fact that these 25,000 instructions weren't all going to be correct, and that there were going to be difficulties that would show up only after a lot of use. So for a whole year after we distributed it, the whole group worked full time just getting messages from California and elsewhere about things that wouldn't work, and then running these cases down and laboring long and hard to figure out what went wrong. We'd correct it, send it back, and sometimes the corrections were incorrect. Hal Stern was our chief coordinator of corrections. Once he had sent out a correction all over the country, if it was an erroneous "correction," as occasionally happened, Hal would be on the telephone continually for days while we worked to clear up the "bug."

It's very hard to convey all the things that pleased us all while we were doing it and shortly after it came out, but we had a good time.

Q. *Given what you know now about languages and programming, and given the fact that today computers are cheaper and programmers are more expensive, how would you approach the problem today?*

DO 30 I=1,N
KI=KI+N
HOLD=A(KI)

A. If we were doing it again, we'd do it like John Cocke at IBM's Scientific Center in Palo Alto, California. Cocke has devised methods that do a lot more optimization than we did by doing it in a far more orderly way. He can probably do a lot more in less time. It hasn't been demonstrated yet conclusively, but I'm pretty sure that would be the case.

Q. *What are your feelings about standardization? What if* FORTRAN *had been frozen one year after you did it?*

A. It would have been pretty horrible if we had been frozen at the very outset. In fact, about a year after we did it, we began putting out another system which was very much more acceptable. The big shortcoming in the first system was that it lacked the facility to define new programs . . . in the sense that it lacked the ability to accept subroutines in FORTRAN. Our second effort, FORTRAN II, produced under Libby Mitchell, had that facility built in, and it was, of course, a great improvement. Yet, the specifications of the system now aren't terribly different from that second try. There have been a number of improvements, but I don't think they are as fundamental as that one . . . there's been a lot of cleaning up of small idiosyncrasies that came from our nearsightedness about things. For example, when writing the system, we didn't have in mind its application as a device for allowing you to run the same program on many machines. We had only one machine in mind—the 704. Also, a number of additions have been made which make many things much easier to express.

Q. *In your view, should development be toward a multitude of languages for many specific disciplines, or should the path lead toward a single, general language allowing only minor variations for different applications?*

A. I don't know that I would like to go along either of those lines. I guess the former more than the latter. I think what is needed is an arrangement whereby people can describe a language which they want to use to say what they want to say. In other words, to use their own notation where that's convenient. As I envisage it, it would have various little pieces of language that would be useful for a number of people . . . and you could draw on those and say, "I want this statement and that statement, this kind of notation, or that" . . . and then for particular applications in your own work, you could define your own notation and add it to those pieces that you had selected, that were already previously described, and with which you were familiar. But basically, I think it's getting to the point where a language like FORTRAN, as well as all the other languages being proposed, is becoming too much for a user to learn . . . too much to provide all the richness that is needed for him to say something very concisely and conveniently, and that the best route is to let people describe their own languages.

Q. *What are the prospects of building a machine whose programming language is English?*

A. Dismal. If it's really English, we just don't know how to cope with it now. And I think it will be a long time before we can cope with it quickly enough and efficiently enough to be economically feasible. And if you have some kind of hybrid that looks like English, it becomes even more confusing and more difficult to learn than an artificial language, where you at least know that it's not English, that English rules do not apply, and you know that you have to learn a set of rules. I think

(Please turn to page 19)

Princeton University and the computer

Big Machine on Campus

THE computer has been in residence on university campuses a relatively short time, but already it has had a profound effect on scholarship in almost every field—even in classics and music. At the same time its use in science and engineering, whose needs brought it into being, has continued to spiral upward.

Though Princeton's growing computer capabilities are impressive for an institution of her size, the swelling demand has put them under strain. The University will soon have one of the very best of a new generation of machines and, with it, greatly increased capabilities. But as the demand for computer time continues to climb no one is quite sure whether a machine with many times the capacity of Princeton's biggest machine at present will be adequate in only a few years.

Princeton's key computer today is an International Business Machines 7094, one of the largest and most powerful members of what is now the older generation of computers. As installed here, it has a capacity to store more than 56 million characters, or units of data, enough to contain some 22 novels the size of *Gone With the Wind*. It can whip through 250,000 additions or subtractions, 100,000 multiplications or 62,500 divisions per second. So big and fast is it that its time is too valuable to be spent on processing input data into a form it can use, and printing out the results. These tasks are performed for it by an IBM 1410 computer, a medium-sized machine that itself is big enough for many computer installations. The 1410, in effect, is a research assistant, taking care of the humdrum jobs so the professor can concentrate on the important ones.

The 7094 assembly is situated at the main quarters of Princeton's Computer Center on the second floor of the Engineering Quadrangle. On the floor above is another computer, the IBM 7044, installed in January. This machine, which also has a smaller assistant, has a capacity rated at 75 per cent of the 7094.

Three medium-sized computers at the James Forrestal Campus come under the Computer Center—an IBM 1410 at the Plasma Physics Laboratory, an IBM 1620 at the Daniel and Florence Guggenheim Laboratories for the Aerospace Propulsion Sciences and an IBM System/360 Model 40 at the Princeton-Pennsylvania Accelerator. The Model 40 is a medium-sized version of the widely heralded new generation of computers. This machine is due for later conversion to a larger version, the Model 50.

The University also has access, for 20 hours a week, to a Control Data Corporation 1604 at the Institute for Defense Analyses, a nonprofit corporation devoted to scientific studies for the Department of Defense, conveniently located in John von Neumann Hall next door to the Engineering Quadrangle. Finally, there is in the Controller's office a small computer, an IBM 1401-G—not under the Computer Center—which, unlike the other machines, cannot handle magnetic tapes for data processing but is limited to use of punched cards.

PRINCETON'S big new machine will be an IBM System/360 Model 67. It will have a computing capacity six times that of the present 7094 installation, according to Prof. Edward J. McCluskey, who directed the University's Computer Center until July 1966. (Prof. McCluskey has returned to teaching and research in the Department of Electrical Engineering and has been succeeded as Director of the Computer Center by Roald Buhler.) "The 67 should be able to take care of all the University's general computing needs for several years," Prof. McCluskey said. "After our people have had a few months to make the change we should be able to give up most of our other major computers." The Model 67, he explained, provides capacity for growth because various extra sections can be added as needed.

To house the new computer, the University will build a new $2-million building on a four-acre plot near Palmer Memorial Stadium, convenient to William Charles Peyton Hall (new home of the Department of Astrophysical Sciences) and to the other buildings now rising nearby for the Departments of Mathematics and Physics. It will also be near the Engineering Quadrangle and the campus headquarters of the humanities and social science departments.

The building will provide 50,000 square feet of floor space, substantially more than is available in the Computer Center's present quarters, and will be designed so that it can easily be enlarged if required. The new building will be a boon not only to the computer staff, which at its present strength of about 30 is quite cramped, but to users, who will gain desk space for such things as doctoring programs and scanning punched cards for errors.

As an interim measure, the Center plans to install by mid-1967 a simpler version of the Model 67 that will still multiply the University's processing capacity. The Center hopes to work out the programs and pro-

cedures required for all System/360 machines on this one so that the changeover will be as painless as possible when the bigger version is installed in the new building in 1968.

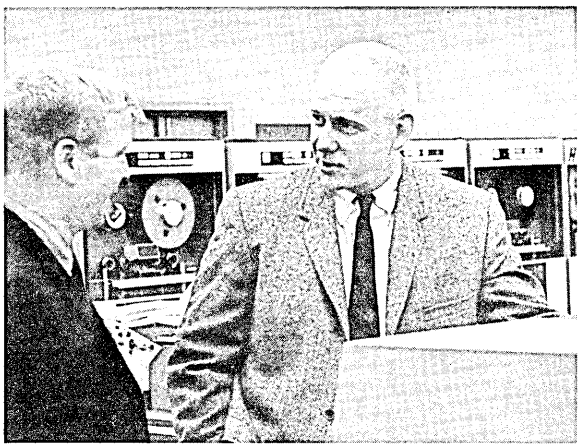
Princeton conceives of the Computer Center as a University-wide scholarly resource, like the library, which members of the University community may use without charge. Computer time does not appear as a direct charge in the research grants Princeton receives from the government, corporations and foundations. In effect such use is paid for through a charge for indirect costs, or overhead. (There is some pressure on the University to change this policy.) The Center also has direct support from the National Science Foundation.

The main Computer Center machines now run virtually around the clock, seven days a week. The only times they aren't working on users' problems are a couple of hours every morning, when IBM personnel stationed at the University take them over for preventive maintenance, and at odd times when Center personnel use them in their own work in improving the languages and programs that set up the circuits and operations needed to solve different problems.

The Center runs on a "job-shop" system—the user brings in his job coded on punched cards or, occasionally, magnetic tape and later picks up the machine's output, often an accordion-fold stack of continuous green-striped paper covered with columns of numbers. At present the average waiting or "turnaround" time for a small job is two hours or so. Before the 7044 was installed the demand led to such queues that the delays were much longer.

ONE feature of the Model 67 that, it is hoped, will ease such bottlenecks and, in fact, make the computer almost as accessible as a dictionary will be a mode of operation called time-sharing. Many Princeton users, irked by the delays and their remoteness from the machine under the job-shop system, are eagerly looking forward to this innovation.

Time-sharing means that a user will no longer be restricted to dropping a prepunched deck of cards on the counter and waiting for the output to come back, though this may still be the best procedure for jobs requiring a long sequence of computations. Instead the user will have access to the machine through a typewriter-like console in his own building. He will be able to type out instructions on the



PRINCETON'S COMPUTER CENTER, under the direction of Roald Buhler (right), is in operation twenty four hours a day.

keyboard and have the machine answer him. So far as he can tell he will have exclusive control of the machine, even though someone else—perhaps dozens of others—may be using it at the same time. This will be possible because the machine can slice its operating time into slivers measured in millionths of a second and deal them out to the various users.

A new time-sharing feature will be provided first on the System/360 Model 67 to be installed in 1967, but most of the time this computer will be fed data in the normal way. About six time-sharing consoles will be used on the site a few hours a day mainly to test the different types of console so the Center can judge which kind will be most suitable for remote use.

Time-sharing will make possible a host of uses that are impractical now. An engineer will be able to use the machine to help develop a problem a step at a time, checking for errors as he goes, as well as to produce the final solution. This will save him time and effort, for in designing a program for the computer he will not need to worry about having all the complicated steps worked out perfectly in advance. At present he may work for days designing his program and wait hours for the solution, only to find that some oversight in his program's logic prevented the machine from reaching the answer.

An architect will be able to engage in a sort of dialogue with the machine, asking it questions, getting answers, and asking new questions on the basis of the earlier answers. At the same time a chemical engineer will be able to develop a program on a

"What is difficult is getting a modern student to think at the 'simple-minded' level of the modern computer when he is programming."

MANY PRINCETON STUDENTS use the IBM 7094 for homework. In Computer Center's "ready room" (above) students prepare their punched-card decks (stored in wall rack) for processing, and they review results for programming errors.

"One interesting thing about the computer is that many classical methods of analysis that had been neglected because they required tedious hand computation are coming back into vogue."

complicated problem involving, say, the depletion of an oil reservoir, and an anthropologist can feed in data on an obscure Bantu dialect in order to study vowel shifts.

The Computer Center expects the initial installation of the new machine to include a number of remote consoles. If the reality of time-sharing comes up to its promise, 100 or more consoles will eventually be placed about the campus.

If demand continues to mount as expected, the Computer Center will have several possible ways to cope with it. As installed, the System/360 Model 67 will have two processing units—in effect, it will be a double computer. Two more processing units can be added, which would nearly double its capacity. Eventually the Center plans to connect a System/360 Model 91 with the Model 67, providing many times the capacity of the present 7094. This should be adequate for several years, but with the present rate of growth in demand few care to predict exactly how long this will hold true.

THE first computer of the University's own was acquired in 1952 and installed at the Forrestal Campus. This was a punched-card-programmed calculator, a rudimentary machine by today's standards, that was used on a job-lot basis. Six years later an IBM 650 was set up in the Gauss House on Nassau Street for the Statistical Techniques Research Group and the School of Engineering. In a few months Project Matterhorn, now the Plasma Physics Laboratory, acquired a 650 that later came under the control of a University-wide committee, the forerunner of the present Computer Center. A computer was installed at the Institute for Defense Analyses in 1960, and the Guggenheim Labs received their first computer in 1961.

With the University's computer requirements burgeoning, the Computer Center was created late in 1961 and Prof. McCluskey, a professor of electrical engineering, was named its director in 1962. Its principal machine was an IBM 7090, operating by December, 1962, which later was converted to the present 7094 by replacing some of its components with faster ones.

An example of how a user goes about designing a computer program to solve a problem that is not obviously numerical has been given by Dr. Carl Helm, 35-year-old lecturer in Princeton's Psychology Department and an active worker with computers. Dr. Helm takes as his goal the solution of a puzzle called "The Towers of Hanoi." This puzzle consists of three rods or pins and a set of pierced discs. The discs are of different sizes, so that, stacked on one pin in order of size, they form a cone, presumably reminiscent of an Asian temple. The object is to transfer all the discs from one pin to another by moving them one at a time without ever placing a larger disc on a smaller one. Thus if there are three discs numbered 1, 2 and 3, and the pins are designated A, B and C, the discs can be moved from pin A to pin C with these moves: 1:A-C (that is, disc 1 from pin A to pin C), 2:A-B, 1:C-B, 3:A-C, 1:B-A, 2:B-C and 1:A-C.

The puzzle gets enormously more difficult as more discs are added. If there were 10 or 15 discs it would be so complex that a person lacking a correct strategy might blunder along forever without finding a solution. Even with a correct strategy it would take nearly forever for a 50-disc puzzle.

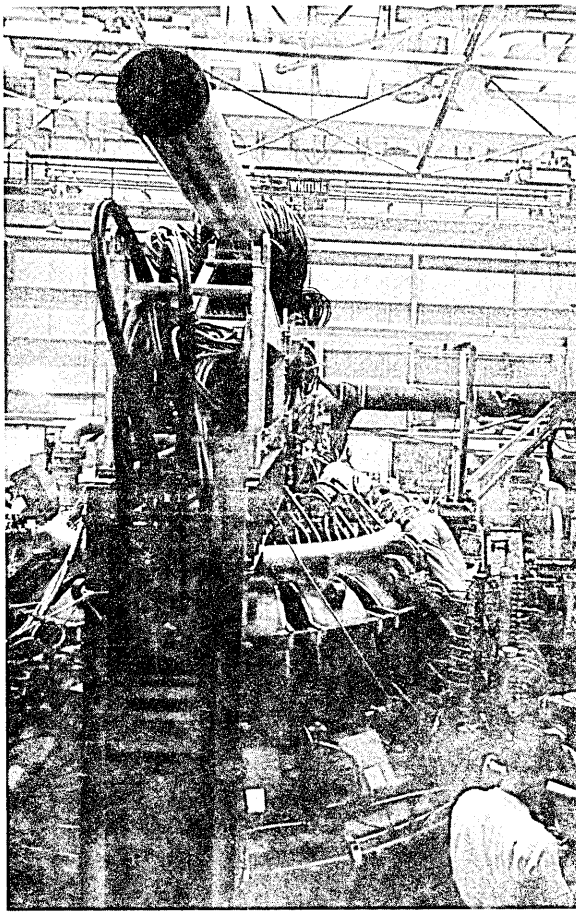
How could the problem be given to a computer? Dr. Helm devised a strategy based on the psychological principle that people often try first to make the move that will get them as close as possible to a solution as quickly as possible. With this in mind he ranks the possible pin-to-pin moves in this order of desirability: first, A-C; second, A-B; third, B-C; fourth, C-B; fifth, B-A; and last, C-A. Since it is not clear that the move from pin A to pin B is preferable to the move from pin B to pin C in terms of this strategy, he ranks these moves arbitrarily.

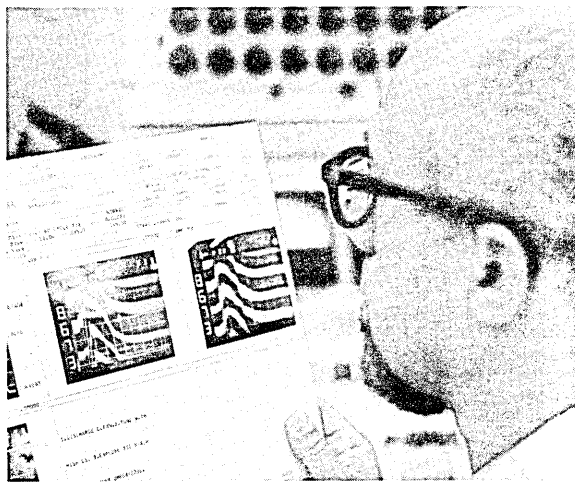
Dr. Helm compares the computer's performance to that of a clerk who can perform a few simple operations absolutely flawlessly. This clerk has a large blackboard on which he writes the instructions and data given to him, as well as the results of various intermediate calculations. The clerk represents the puzzle's pins by columns labeled A, B and C on the blackboard and writes the numbers 1 to 5 in order from top to bottom in column A to represent the discs in the starting position. On another part of the blackboard he writes a "move list," composed of the six possible moves in their order of priority. Since the clerk has no capacity to make independent judgments, he must be instructed in great detail. We can present him with this operating method, or program, incorporating the rules of the game:

Instruction 1. Get the first move from the move list.

Instruction 2. Would this move put a larger disc on

PROFESSOR LELAND ALLEN of the Princeton Department of Chemistry is surrounded by some of the computer printout generated in his quantum-mechanical investigations of the electron structure of molecules.





DR. MILTON ROTHMAN reviews printout of results obtained in thermonuclear-fusion experiment conducted with ionized matter, or plasma, contained in ring-shaped Stellarator (top).

"The digital computer, perhaps more than any other innovation in history, has suffered from the prejudices and misunderstanding attendant to any major change."

a smaller one? If so, go to instruction 5. If not, go to instruction 3.

Instruction 3. Move the disc as indicated.

Instruction 4. Are all the discs on pin C (that is, is the job done)? If so, stop. If not, go to instruction 1.

Instruction 5. Find the next move on the move list, then go to instruction 2.

Told to begin, the clerk (1) gets the first move from the move list, (2) finds pin C empty, (3) moves 1:A-C and (4) finds the job unfinished and returns to instruction 1. (The numbers in parentheses refer to the number of the instruction being executed.)

Now he (1) finds the first move (2:A-C), (2) finds it violates the rule and goes to instruction 5, (5) finds the second move, (2) finds it legal, (3) moves 2:A-B and (4) finds the job still unfinished. Shuttling between instructions 1 and 5, the clerk now finds he cannot make the first, second or third moves (3:A-C, 3:A-B, 2:B-C) because they would place discs on smaller ones. The fourth move (2:B-A) is legal and is therefore executed. However, this results in moving disc 2 back to the pin it just came from. Since the next move will put the same disc on pin B again a circular pattern is established that will get him nowhere. In computer parlance, the clerk is "hung up in a loop."

How can this be fixed? Dr. Helm suggests a simple amendment to instruction 2 so that it reads, "Will this move put a larger disc on a smaller one *or move the disc that was moved on the preceding trial?* If so, go to instruction 5. If not, go to instruction 3." With this change the program can be executed by the clerk to complete the task in 81 moves.

This example illustrates two important characteristics of computer work—first, the power of the conditional response (note the "if" clauses in instructions 2 and 4), and second, the fact that all programmers sometimes fall into traps of illogic that don't become apparent until their programs are tested in an actual run. Finding and correcting such lapses and the more mundane key-punching errors make up the "debugging" process, a standard programmer's practice.

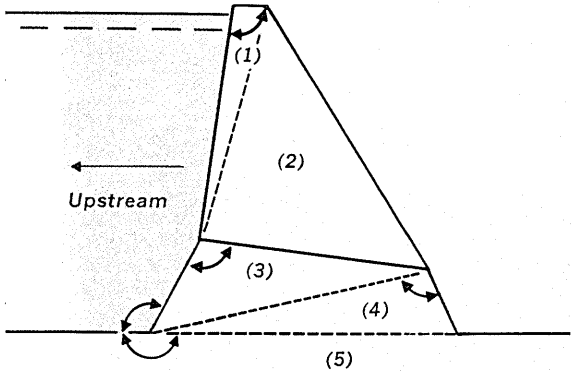
The procedure described here is not the most efficient one possible. If the priorities of the second and third moves were reversed the program would find a solution in 31 moves, the mathematical minimum. But Dr. Helm's purpose was to construct a kind of working model of the way a person might actually solve the puzzle. The question of efficiency was secondary to the question of whether the procedure would successfully mimic human performance, even making the same wrong moves. The power of the computer makes it possible to explore such problems even though the efficiency of the procedures may be very low. Dr. Helm has observed at least three other strategies for attacking this puzzle that lead to quite different programs.

THE uses of computers at Princeton are so many and so varied it would be impossible in this space to describe them all, but a few examples may suggest the extent to which the computer has made itself indispensable.

In an applied science, Prof. Rowland Richards Jr. noted that the computer is now widely used in all phases of civil engineering. It has caused a revision of many existing techniques of analysis, design and construction, he said, and has also bred new ones, such as the "systems" approach to decision-making —used at Princeton primarily for transportation planning. Many courses in the department therefore stress familiarity with the computer.

"One interesting thing about the computer," he said, "is that many classical methods of analysis that had been neglected because they required tedious hand computation are coming back into vogue, since conceptually they are relatively easy. This trend is good for the student; it points up historical continuity and the importance of fundamentals not only in formulating problems but also in solving them."

As an example Prof. Richards cited the problem of designing a concrete gravity dam—one that stays in place through its own weight. A variety of alternate configurations can be solved through visualizing the dam as a combination of wedges (see diagram) and applying the standard solution for a single wedge with different angles and surface loadings. The first wedge is solved, but this gives incorrect stresses on the exposed face of the second wedge. The second and remaining wedges are adjusted in turn by applying correction loads to satisfy the known external conditions. However, although a rough approximation of the stresses throughout is reached at this point, the exposed faces of the fifth wedge now do not satisfy known conditions and the correction sequence must continue in reverse order back to the starting point. After one cycle the first wedge is again out of balance and the whole process starts over with



slightly more accurate initial values. Eventually, through repetition, the stress field is determined that will match the water and air pressures on the dam's boundary. The precision depends primarily on the number of cycles.

This procedure, known as "iteration," is one of the principal techniques of problem-solving by computer. "It might take 20 hours for a student to understand the actual computer program to do this," Prof. Richards commented, "but the basic strategy is simple. In fact, getting a modern student to think at the 'simple-minded' level of the modern computer when he is programming is what is difficult."

Prof. Leland Allen, of the Department of Chemistry, is Princeton's largest consumer of computer time. He is also one of the University's most articulate and forceful apostles of the computer, seeing a limitless role for it in all academic disciplines.

Prof. Allen's work involves theoretical studies of the electron structure of molecules, which in even simple types can be bewilderingly complex. With a group of graduate students and post-doctoral research associates, he has developed and applied several new quantum-mechanical techniques to a wide variety of chemical problems. One of these is the explanation of the bonding mechanism in noble-gas compounds. Until recently it was a fundamental hypothesis of chemistry that atoms of certain rare gases were incapable of entering into chemical combinations. All this has changed with the discovery of such compounds by Prof. Neil Bartlett (who joined the Princeton faculty last summer). Prof. Allen's group has also been using computer-generated electron structures to predict the types of noble-gas compounds that are *unlikely* to exist, which is very difficult to do by any other method.

WORK at the Department of Astrophysical Sciences focuses on two problems that are central concerns of the field today. Prof. Martin Schwarzschild uses the computer to study one of these, the life cycle of a star. "The physics of the problem is quite varied," he said. "It involves atomic processes, nuclear processes, thermodynamic equations, the laws of physics applied to the state of gases in the interior of a star. The computer gives us a chance to follow a star through all these processes step by step, to see how its size, brightness, appearance, and so on, change as the star ages.

"Some other very exciting work is being done in this department by Prof. Dimitri Mihalas. He is using the computer for the analysis of the stellar atmosphere, the outer fringe of the star that we can see directly. In this way he hopes to determine the chemical composition of the star, which is practically synonymous with the composition of the universe. The mathematics of this problem is very tricky. The loss of radiation as it interacts with the individual atoms in the stellar atmosphere must be taken into account in detail. Only a computer can do it.

"I might also add that the computers may come in for very heavy use from our Orbiting Astronomical Observatory, which we hope will be circling the earth with its 32-inch telescope in 1968. This satellite will be studying interstellar matter and will be sending us large quantities of data in digital form that will be impossible to handle without a computer."

In a social science, Prof. Michael D. Godfrey of the Department of Economics has used the computer's cathode-ray tube to give undergraduate classes an idea of how the economy works. "We have taken models of the economy from the economic literature and written programs for them," he said. "Then we display the results on the tube and see what happens when certain factors are changed over a period of time—the level of investment, for example. This use has been quite successful."

Prof. Godfrey has also used the computer for research. One effort has been to analyze how the government's adjustment of economic data to remove seasonal influences affects the quality of the data. His conclusion is that this procedure has a serious effect

"The digital computer fundamentally changes the way we ask questions about the physical world and about human nature."

that should be of concern to economists.

In the Department of Classics, Prof. Samuel D. Atkins has developed a computer program that can be used for compiling concordances for works in ancient languages. A concordance is an alphabetical catalogue of a work's words, with the context in which they appear. A familiar example is the concordance provided with many editions of the Bible.

The computer program has been used to compile a concordance of Vergil's *Eclogues* and other works in Latin and Hittite. The work is transferred to punched cards and fed to the computer, and the output is the finished catalogue. "Concordances have been compiled for many years, but by hand," Atkins explained. "They are very useful to scholars studying a number of problems, such as stylistics, authorship, meter, linguistics, and so on."

THE computer is gaining in use in Princeton's administration, too, and even here gives some scope for program research and development. The small card-operated machine is at present used principally by the Office of the Controller in preparing the payroll and distributing budget accounts, according to Joseph G. Bradshaw, General Manager of University Services. Eventually, when the Computer Center's System/360 Model 67 is installed, the administration will give up its machine and use a small System/360 Model 20 as a remote terminal of the big computer. Mr. Bradshaw expects other offices of the administration, such as the Purchasing and Food Services Departments, to develop effective uses of the machine.

An adjunct of the Registrar's Office, the Office of College Operations, has devoted considerable study and work to computers. "The requirements of a university are becoming so complex that a machine is almost necessary," said the office's director, Dr. William A. Stuart. "At most universities, computers are used to process data for a large number of students. At Princeton, however, we can develop computer applications and systems to produce data in depth on individual students. These data can then be brought to bear on the decisions which *people* must make—admission, majors, courses, and so on."

The assistant director of the Office of College Operations and its computer expert is Dr. Walter B. Studdiford, who doubles as a lecturer in the Department of Psychology. One of his recent projects was to adapt for Princeton's use a program developed at MIT for scheduling classes, assigning classes to classrooms, apportioning the students who sign up for them, and so on. He is also working on various programs that will make student data available when and where needed. The office undertakes special analyses, such as studies of how well an incoming student's Princeton grades can be predicted by his scores on pre-admission tests. (The answer: very well indeed.)

Chemistry Professor Allen views the invention of the computer as one of the most important events in human history and sees its use as transforming all fields of scholarship, not just chemistry and the other natural sciences.

"The digital computer, perhaps more than any other innovation in history, has suffered from the prejudices and misunderstanding attendant to any major change," he has written. "This is not unexpected since the digital computer fundamentally changes the way we ask questions about the physical world and about human nature. Many sophisticated scholars have viewed the digital computer mechanistically as a device contributing a mechanical and impersonal element to mental activity and in some way diminishing the human spirit. In reality an exactly opposite manifestation is the greatest promise offered by its existence.

". . . Application of nonnumerical methods will contribute a larger fraction of scholarly computer use in the future and this shift will parallel the diffusion into other disciplines. . . . It is clear that the influence of digital computers on the whole spectrum of scholarship is so great that all major universities must make this area a central feature of their planning."

The value of the Computer Center as an educational resource was stressed by Mr. Buhler, its incoming Director. "Computer experience is becoming more and more vital to students entering many different careers," Mr. Buhler said. "The only way they can get this is by actually using a computer. Princeton has an excellent facility that is already used by students to a large degree. We hope to be able to make this use still larger."

—Charles E. Pepper

(Continued from page 10)
a lot of people have found that it's just deceiving to call it English, because it's really artificial and its only merit is that it can be read by people without too much training. Yet, it can't be written by people without too much training, and to learn it is difficult because of the confusion between its artificiality and its apparent similarity to English. Beyond that, even the reading of it is a sort of spurious thing, because most programs that are of any significance are really sufficiently complex, so that an untrained person isn't going to be able to follow it anyway. And for the trained person, all the verbiage and extra words only detract from the conciseness of it and, therefore, the ability to understand it.

Q. *Is the fact that English itself is a very complicated language part of the problem?*

A. Yes. You must have simple formation rules. And when you're trying to make a programming language look like English, you invariably get into rather complex formations because there's nothing simple about English. There's a fascinating little study by Casimir Borkowski in IBM *Research* that analyzes the problems in merely picking out a piece of newspaper text and trying to identify personal names in it. The task was selected because of its apparent simplicity . . . names are usually preceded by a "Mr." and first letters are always capitalized. So you think it might be easy to do, but when you get into the details of actually doing it, it is enormously difficult. The number of rules and exceptions that you get in to is simply hair-raising.

Q. *There seems to be of late a diminishing interest in so-called artificial intelligence programs, such as game playing and pattern recognition. What's your view of this whole area of heuristic programming? Did the first promising steps in the 50's ever live up to their potential?*

A. No. First of all it's a terribly ill-defined expression in that it seems to imply that you're trying to do something that you really don't know how to do. Heuristic programs that have been written on that basis have turned out to be sort of abysmal. There's a classic example where an elaborate program was written to prove some of the theorems in *"Principia Mathematica."* The program proved something like 20 percent of them, but many of the ones it did prove took 15 or 20 minutes or more. Then, in one summer Hao Wang, a well-known logician, learned how to program and then wrote a program that proved all of them in two minutes.

Q. *In the* FORTRAN *that you developed, expressions are permitted only when all of their elements are of the same type (either all floating point or all integer). Similarly, only a fairly rigid kind of expression can be used as a subscript. With Operating System/360* FORTRAN *you can have mixed expressions and generalized subscripts. In your view is this a proper development path?*

A. Sure.

Q. *Why?*

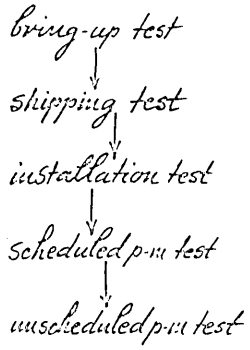
A. It's a good path because it makes it easier for the user. Less restrictions, and so on. The difficulty is that it makes it harder for the compiler. Some of the reasons that some of the idiosyncrasies in FORTRAN exist is the effort to try to make it clear to the user when the system will be able to efficiently code his problem and when it won't. And if you allow all kinds of generalized subscript expressions, why then the compiled code will be more efficient in some cases, but not so efficient in others. And it will be more difficult for the user to know where the boundary between the two is. In that sense, there is a difficulty, but people aren't so worried about efficiency anymore anyway. Nobody writes code by hand anymore, so they don't know what efficiency really is. They get it all through some system and unless it's simply awful, they don't know whether it's efficient or inefficient.

Q. *Looking to the future, do you foresee any important modifications of* FORTRAN*?*

A. I think that a lot more can be done to get computers to do what you want them to do . . . make them more accessible to more people . . . and make it still cheaper to program. I think some of the difficulties with programming systems that are being developed is that there is too much to learn. And that's one of the shortcomings of FORTRAN itself. There's too much to learn just to begin programming. With console type operations, where a person can really communicate with a computer, the system should be able to teach the user, be able to answer his questions, be able to tell him how to say what he wants to say. But the future of FORTRAN I guess is tied to the fact that a lot of people already know it. Many thousands of engineers and other people have taken the trouble to learn it and want to use it.

Finding faults in computers

Diagnostic engineers program computers to reveal how well they are working, and even whether they were properly designed



Diagnostic engineering is to computers as the practice of medicine is to people. Like medicine, diagnostic engineering is both an art and a science. It involves the development of methods for testing computers, finding errors in equipment design, isolating component malfunctions, and generally improving the overall operation of data processing systems.

John J. Dent is the Diagnostic Engineering Manager at IBM's laboratory in Kingston, New York. Mr. Dent's staff of 60 engineers is responsible for designing programs to test System/360 Models 65, 67, and 75, and 2250 and 2260 display systems, all manufactured at Kingston, and System/360 Model 91, manufactured across the Hudson River at Poughkeepsie.

"Good diagnostic programs and testing," says Mr. Dent, "are intended to increase the performance of our products by catching any design errors before the systems leave the plant. The same programs are used by customer engineers in the field as system maintenance tools."

The diagnostic engineer is at work early in the designing of a new system. He reviews such preliminary specifications as processor computing speed, type and capacity of storage elements, type and number of input/output devices, instruction operations, and special equipment configurations and design criteria. From these specifications, and his intimate knowledge of system components and how they interact with each other, the engineer determines whether the proposed system can be efficiently programmed and maintained. If he sees room for improvement, he recommends appropriate design changes. He also reviews the preliminary designs with an eye to recommending maintenance features and planning the programs to be used in diagnostic testing.

Five stages of testing

Those programs, placed on magnetic tape, are used at a number of stages during the life of a computer. First is a *manufacturing bring-up test,* which seeks fabricating errors—for example wiring faults and bad components. Next is a *shipping test,* at which time, explains Mr. Dent, "we run the toughest program available to make sure the computer is solid—that all units are working together as a total system."

After a computer has been shipped to the customer and installed, it is tested once again. "The *installation test* is similar to the manufacturing bring-up test," he states, "except that we are looking not for fabricating errors but for faults that may have appeared during transit."

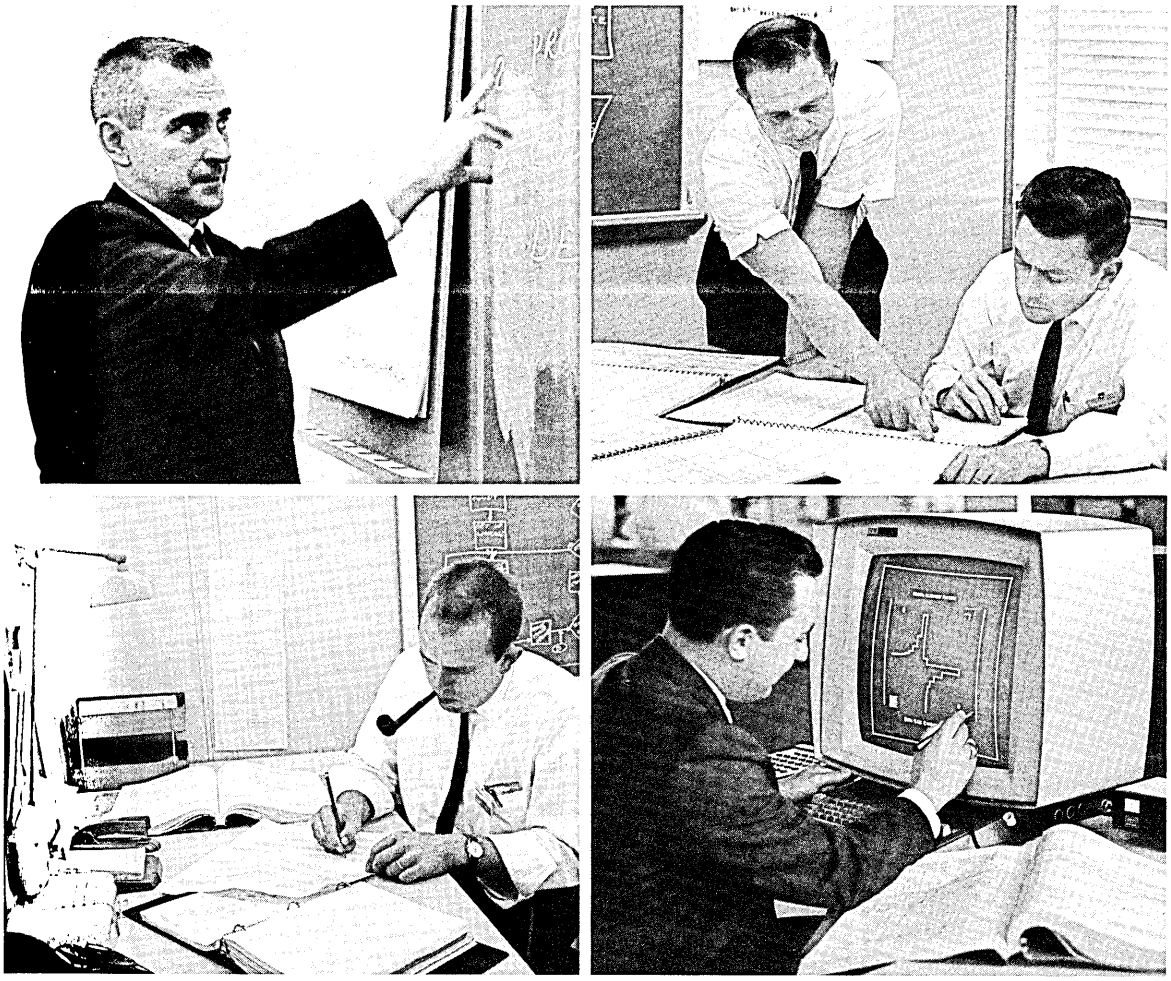
Then the customer engineer uses the diagnostic-program tape for regular, scheduled *preventive-maintenance tests,* as well as for any *unscheduled tests* required because of an unexplained machine breakdown. A goal of diagnostic engineering is to come up with one system of test programs that will satisfy all of these requirements.

Mr. Dent points out that there are two basic concepts of diagnostic testing. One is functional testing, which determines that a given machine performs the functions it was designed to perform. The other is logic testing, which determines whether all the logic circuits are operating correctly; that is, it tests the hardware. For example, the data-processing compatibility of System/360 is tested by running the same set of functional tests on all models. But a unique logic test is needed for the hardware of each model. As John Dent puts it, "Usually we end up with one program that contains a little of both concepts."

Fault detection and isolation

The diagnostic engineer depends primarily on logical, arithmetic, and condition-sensing instructions to detect faults in a computer system. He analyzes the logic of the equipment and considers the various failures that might occur. Then he develops test routines to detect those failures, analyze the results, and present data on any failures that are found. In designing a diagnostic program, the engineer faces three basic problems:

1. He must determine what the sequence of failure indications will be for each fault defined.
2. He must devise a set of instructions that will



JOHN DENT (upper left) discusses main steps in diagnostic engineering. Those steps are (upper right) reviewing computer logic and designing a diagnostic-program flow chart; (lower left) coding the program; (lower right) debugging. Here analog circuits in a 2250 display console are being tested. Computer-generated alignment pattern can be corrected with a light pen, if necessary, to produce the shape prescribed by the diagnostic program.

detect all the faults defined in a relatively short time.

3. He must devise a fault-isolation program that will determine the specific cause of each fault.

Faults are detected by comparing output data with predetermined expected results or simulated results. Faults are isolated by cross-correlating the information in a set of failure indications. In general, the procedure is to relate each failure indication to a set of possible causes, then eliminate the causes not associated with all failure indications generated during the test.

The diagnostic tape for a typical System/360 Model 65 configuration might contain 220 different programs with a total of perhaps 264,000 machine instructions. The tape tests the system automatically, each program testing one aspect, such as the floating-point circuitry or the magnetic-tape unit. A control program, or monitor, is placed on the diagnostic tape to automatically sequence the test programs. "The monitor permits a number of individual test programs to be run from one reel of tape," explains Mr. Dent. "It also provides the field engineer with a customized test tape that can be used for field maintenance of each system configuration."

Good diagnostic engineering—particularly in the design stage—can mean the difference between a system's early release to manufacturing and wasteful hours of investigation in the laboratory. "The diagnostic engineer is the first programmer to get on a machine," says Mr. Dent. "If anything is wrong, he'll find it." □

Graphics speeds cost estimates

An experimental computer terminal at IBM's San Jose plant enables engineers to cut cost-estimating time by 75 percent

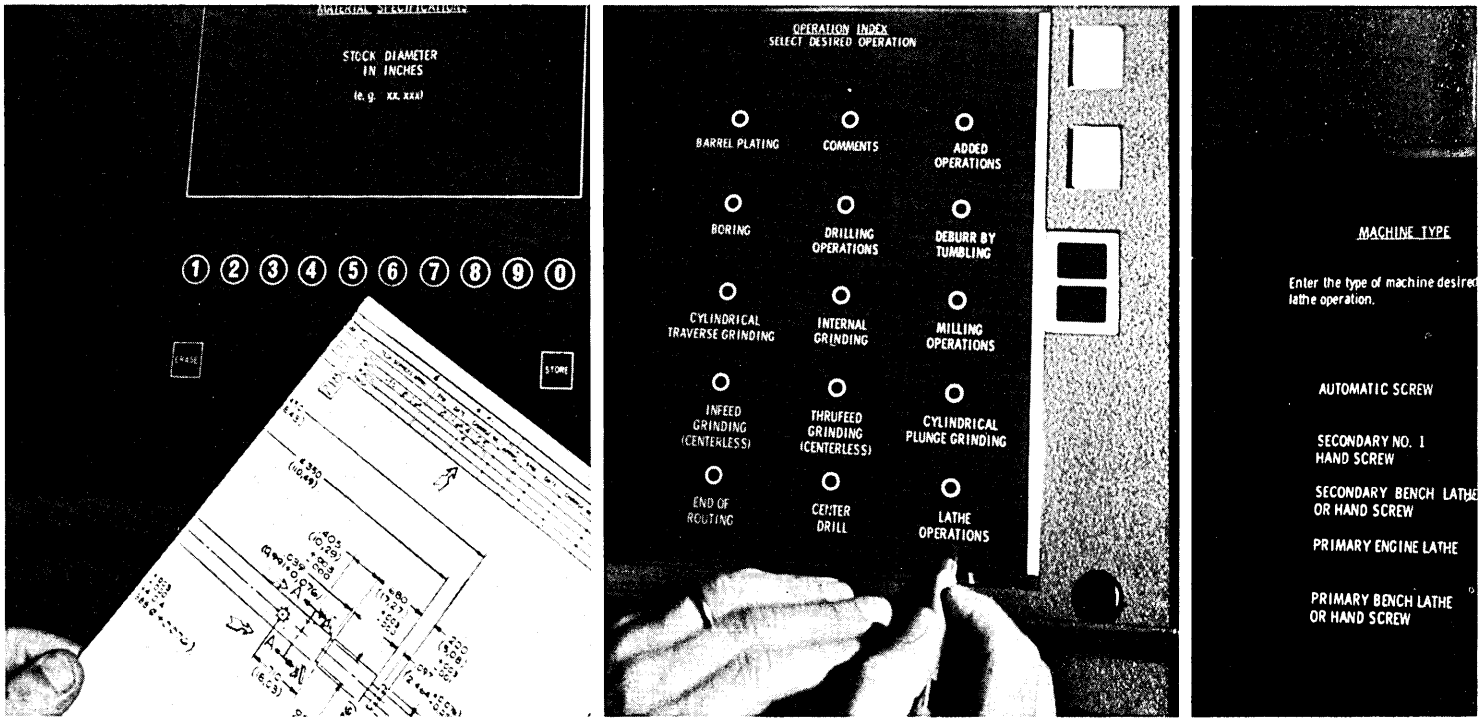
How long does it take an engineer to estimate the cost of manufacturing a new product? For most engineers, who must spend a good deal of time wading through bulky manufacturing-cost manuals, the answer is 'too long.' At IBM's San Jose plant, where this kind of search is conducted at electronic speeds, the answer is different and eye-opening.

At San Jose, the time that engineers devote to making cost estimates has been reduced 75 per cent by an experimental computer-assisted technique. An important key to the new method is a desk-top, image-display terminal linked by telephone lines to an IBM 1620 data processing system. The terminal—developed at the company's Advanced Systems Development Laboratory in nearby Los Gatos—allows rapid interaction, or "conversation," between engineer and computer.

Cartridges of 16-mm film strips containing descriptions of various manufacturing alternatives are used with the terminals. An engineer selects a cartridge, places it in the terminal, and views a series of alternatives on its 9- by 7-inch screen. Simply by pointing an electronic light pen at one of the alternatives, the engineer informs the computer of the one he wants it to work with.

Complete data for each alternative are stored in the computer's memory. As the engineer chooses successive manufacturing steps, the computer keeps a running account of the costs involved. At the conclusion of the problem, the computer prints out a complete cost estimate and manufacturing routing.

There is no need for the engineer to enter in-



WORKING WITH LIGHT PEN on CRT, a cost estimator (1) refers to engineering drawing; (2) selects 'lathe operations' from machine index;

formation at a keyboard or wait while a printer taps out the computer's response to individual questions. He can proceed through a cost problem about as quickly as he can react to the alternatives appearing on the screen. And by having to react only to visual images, he can proceed without interrupting his train of thought.

The images displayed on the screen make up a series of logical steps that lead the engineer through an evaluation of all variables associated with the cost of the part to be manufactured. Usually, about 100 images are enough to cover an entire cost-estimating problem.

How much does a drive shaft cost?

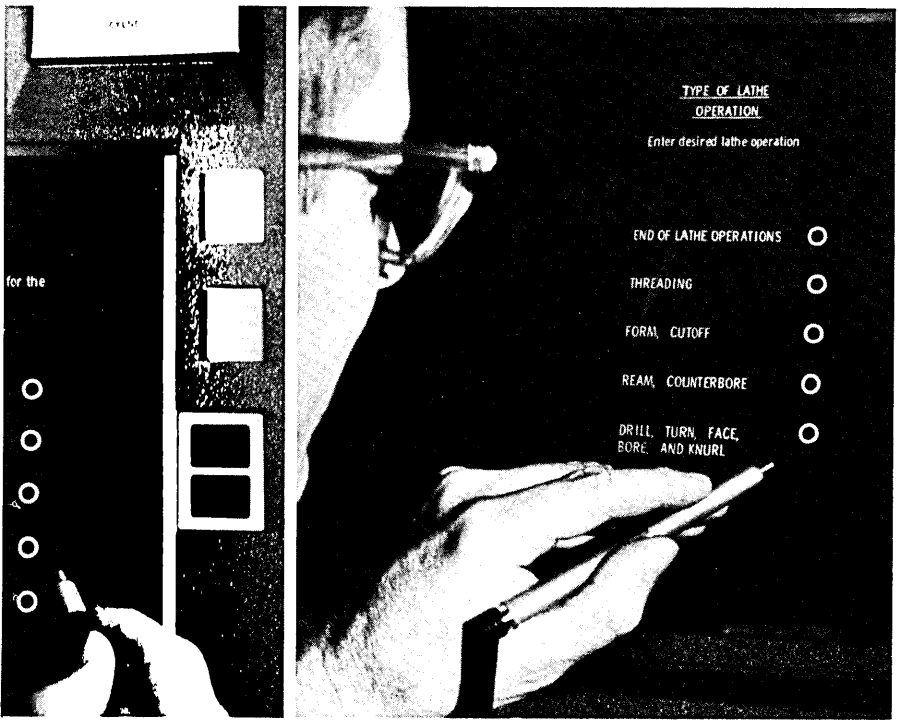
Engineers can use the pilot technique for such commonly produced items as cylindrical parts, stampings, sheet-metal parts, wound cabling, plastic injection moldings, frames, and gates. If an engineer wished to estimate the cost of producing a drive shaft, for example, he would select the film cartridge that corresponded to and triggered a "cylindrical parts" program stored in the computer's memory. This program would be applicable to almost any cylindrical part and would call up from pre-stored data all variables associated with such parts.

After the engineer supplied identification data in response to the first few screened images, subsequent images would request information on the material and its stock shape, diameter, and length. When the material specification was complete, a selection of the various operations that might be performed on a cylindrical part would appear on the screen. Associated with each operation would be several images requesting more details about what the engineer would like to do to the part.

For example, if a lathe operation were indicated, the engineer would be asked whether he wanted to use an engine lathe, turret lathe, or automatic screw machine. Proceeding through the steps, the engineer would be asked whether he desired a form cut, groove cut, or threading operation. He could select such operations at random and repeat them as often as necessary.

After he had specified the desired manufacturing methods, the engineer would call for a computation and print-out of the entire routing. A document complete with step-by-step information and total-cost figures would be printed out after the computer had taken into account tolerance limits, labor rates, pre-programmed information on machine set-up and operating times for each operation, and a host of other pertinent variables.

Comparative cost estimates can be obtained with the experimental terminal by altering only those variables or manufacturing methods the engineer wishes to re-examine. The computer can recall the unaltered variables from its memory and adjust its computations to take account of the new specifications, so there is no need to start from scratch and re-run the entire program. ::



CYLINDRICAL PARTS ROUTING

PART NO. 1234567 DESCRIPTION SAMPLE E.C. 7654321 ANALYSER - KELLY

MACHINE TYPE - 360 SPECIAL TOOL COST $ 0.00

ORDER QTY. 500 LABOR AND BURDEN RATE $ 10.00/HR.

OP. NO.	OPERATION DESCRIPTION	HOURS/100	SETUP
	RAW MATERIAL IS 06-060 CARBON STEEL STOCK SHAPE IS ROUND .750 DIAMETER 3.500 LENGTH, COST IN SUMMARY		
5	A PRIMARY BENCH LATHE OR HAND SCREW, OPERATION AS FOLLOWS FORM OR CUTOFF .750 DIAMETER .380 INS SUMMARY OF HANDLING, MACHINE TIME AND SETUP,	4.456	1.8
10	A SECONDARY BENCH LATHE OR HND SCW, OPERATION AS FOLLOWS DRILL, TURN, FACE, BORE OR KNURL .750 DIAMETER 1.250 INS SUMMARY OF HANDLING, MACHINE TIME AND SETUP,	2.989	.9
15	PLAIN MILLING CUT LENGTH OF .250 INCHES	1.192	1.4
20	A SPECIAL OPERATION WITH $ 0.00 IN SUMMARY COMMENT (25) DEBURR WITH HAND TOOL	.700	.2
30	A DRILL PRESS OPERATION AND PART HELD IN A FIXTURE DRILL (1), TAP (1), REAM (0)	1.320	.6
35	PLUNGE GRIND AND REMOVE .007 STOCK	1.713	.6
	TOTALS	12.372	5.5
	TOTAL INCLUDING PRORATED SETUP	13.472	
	TOTAL LABOR $	1.347	
	MATERIAL COST $	.122	
	A 6 PERCENT ADD FOR PACKAGING $	.088	
	TOTAL ESTIMATE $	1.557	

(3) specifies lathe type; (4) indicates kind of lathe operation; (5) receives computer-generated output four times faster.

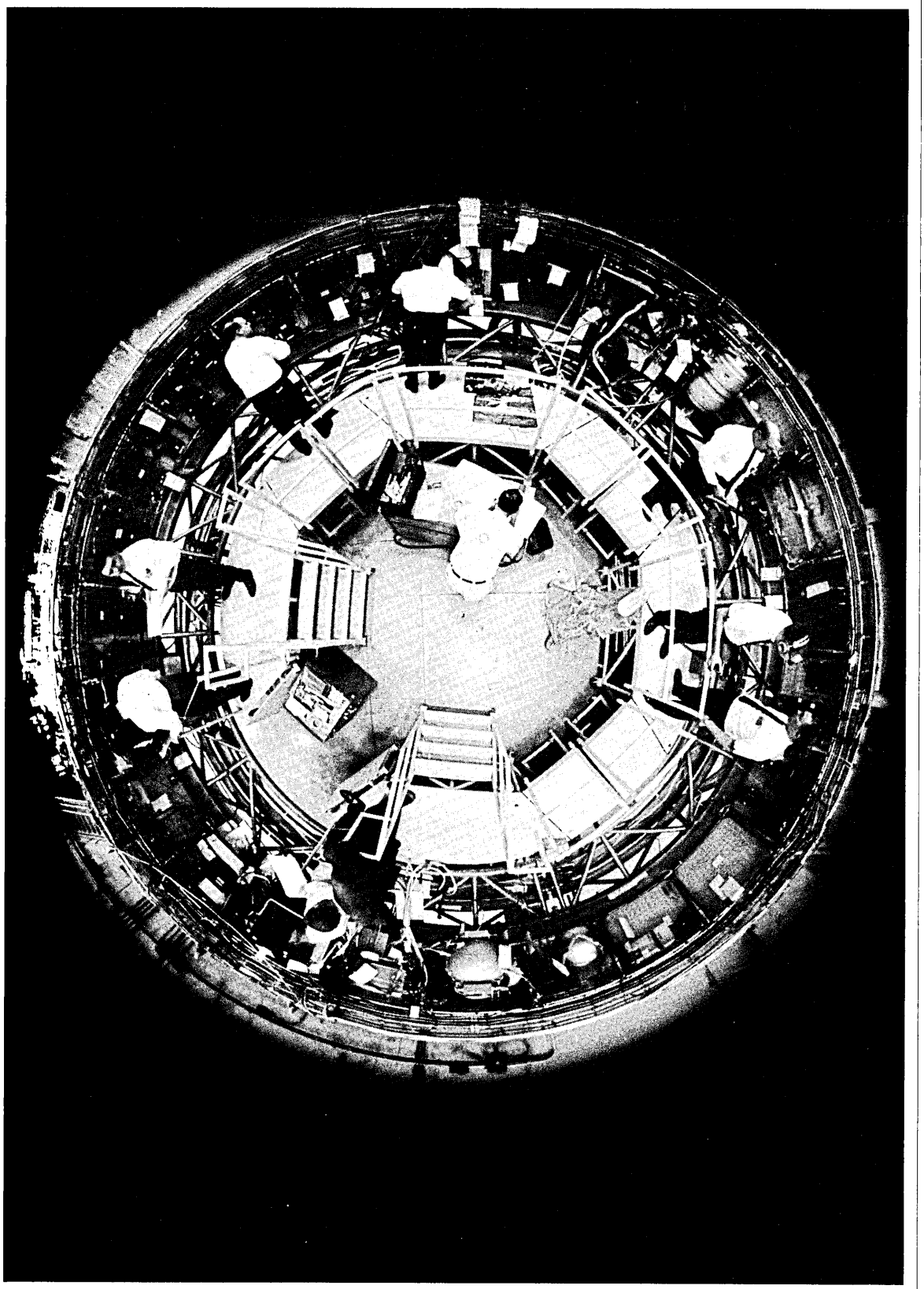
Newsfronts

A NEW MICROFILM CONVERSION SERVICE, now available to IBM customers, is cutting the document maintenance problem down to size. Designed to reduce the cost, space and time involved in the storing of documents, the service provides full microfilm capabilities ranging from camera work to distribution and film maintenance for both aperture cards and microfiche. An aperture card, such as the one shown coming out of the console-type Microcopier (above), is a standard punched card containing a single frame of microfilm. Microfiche are sheets of microfilm on which 90 pages of documents can be reproduced. A frame of a microfiche sheet is projected on the desk-top document viewer at left. The service is available at two centers located in White Plains, New York, and Campbell, California.

Much of the routine work involved in the editing and typing of documents can be eliminated with a new time-sharing system called DATATEXT. The system, which can be used simultaneously by up to 80 customers at different locations, was successfully field tested in a wide variety of applications including the preparation of engineering documents, technical manuals, detailed bibliographies, price lists and telephone directories. Using DATATEXT, a customer need type any document only once for entry into the computer. The computer then accepts revisions to any part of the material at any time and holds the latest version available for immediate print-out at the customer's terminal—at a speed of 150 words a minute. DATATEXT is now in operation in the San Francisco Bay area. Additional systems are planned for installation in Chicago, Cleveland, Los Angeles, New York and Philadelphia.

Scientists and engineers now can hold problem-solving exchanges with System/360 Models 30, 40 and 50 without leaving their work areas or awaiting their turns. A new computer program called RACS (Remote Access Computing System), functions as a high-speed switchboard, controlling the flow of information between remote terminals and a central computer. A prototype of RACS has been in operation at the Lockheed-Georgia plant (Marietta), where hundreds of engineers have access to a System/360 Model 50 through communications terminals scattered throughout the Lockheed complex. (See Lockheed Pioneers Remote Computing, *Computing Report,* April, 1966.) The programming language used to communicate with a RACS-equipped computer is FORTRAN. The system is scheduled to be available in the second quarter of 1967.

A computer that simulates the behavior of pulsating stars is helping astronomers measure cosmic distances with greater precision. Dr. Robert F. Christy, professor of theoretical physics at California Institute of Technology (Pasadena), is using an IBM 7094 computer to imitate RR Lyrae stars located in the Milky Way. By simulating observed stellar spectra of these oscillating stars, Dr. Christy can determine the chemical composition of the actual stars and their distance from each other. His study has shown that RR Lyrae stars are all about 50 times brighter than our sun, about half as massive, and that 30 percent of their surface consists of helium—a much higher percentage than previously believed.

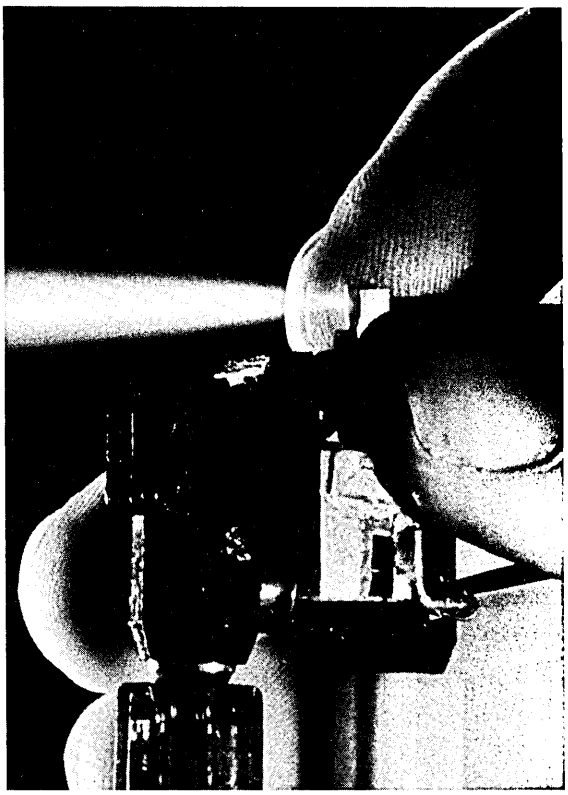


THE "NERVE CENTER" of Saturn rockets during orbital test flights is a three foot high instrument unit assembled by IBM. The unit, which contains 60 different component parts, guides the vehicle, controls it in orbit, and navigates its flight throughout the mission. It has guided all of the Saturn launches to date. These missions included tests of the Apollo spacecraft's heat shield and the behavior of liquid hydrogen in zero gravity.

Communication capabilities can now be added to the IBM 1130 computer with the attachment of a new device—the 1130 communications adapter. Equipped with the new unit, the 1130 can operate either as a stand-alone computer or as a communications terminal with access to the power and memory capacity of the System/360. The ability to switch from one role to the other makes the 1130 well-suited for branches of large firms that need both a local computer facility and access to a more powerful central computer. Other potential users include universities with widely separated campus facilities, government installations, and aerospace companies with remote testing sites. The adapter, which permits the computers to be connected by regular leased telephone lines, will be available to customers in the first quarter of next year.

Computers are taking the conflict out of class scheduling for many of the nation's schools. A new set of programs called Student Scheduling System/360 is being used to assign students to class sections more rapidly and efficiently. The system, which requires no knowledge of programming, enables educators to test alternative scheduling strategies before choosing the one that best meets their educational and fiscal objectives. Two supporting programs within the package known as Tally and Conflict Matrix routines are run prior to actual scheduling. The Tally routine lists the total number of students requesting each course by grade level and sex. The Conflict Matrix program points out potential scheduling conflicts, such as a student registered to take two different courses at the same hour.

Laboratory technicians can eliminate the excess paperwork "syndrome" with a new information gathering system that automatically records and identifies for computer processing the results of hundreds of different kinds of clinical tests. Designed for hospital, pharmaceutical and commercial laboratories, the IBM 1080 data acquisition system cuts time-consuming paperwork, virtually eliminates errors, and speeds reports on test results to doctors or laboratory managers. In addition, the computer can be instructed to perform a "reasonability test" to determine if a patient's test results are in keeping with his medical history. The system will be available in the second quarter of 1967.



PHOTOS TRANSMITTED TO EARTH from future interplanetary space vehicles, such as the Mariner, may be carried on the beam of light generated by a semiconductor laser. A patent covering such injection lasers of a type that includes the gallium arsenide laser (the most widely used of the semiconductor lasers), has been granted to IBM. This type of laser system, now being studied by IBM's Federal Systems Division, may make possible the transmission of photos in a few seconds as compared with the eight hours required to transmit each of the Mariner pictures by radio. The Division, under contract to NASA, is also exploring the feasibility of using the lasers to replace some of the cables in the "umbilicals" of space rockets on the launching platform.

For further reference

Publications available on request from local IBM sales offices

Readers desiring more detailed information on subjects covered in the articles in *Computing Report* may address inquiries to local IBM offices. Some specific publications pertinent to topics in this issue follow. They may be requested through any IBM sales office.

FORTRAN

Operating System/360 FORTRAN IV *Language*	*(C28-6515-4)*
IBM *1130* FORTRAN *Language*	*(C26-5933-3)*
FORTRAN	*(F28-8074-3)*
IBM *1800* FORTRAN *Language*	*(C26-5905-3)*
IBM *7090/94* IBSYS *V13* FORTRAN IV *Language*	*(C28-6390-3)*
IBM *7040/44* FORTRAN IV *Specifications*	*(C28-6330-1)*

Bibliographies

Literature on Information Retrieval and Machine Translation	*(320-1710)*
Literature on Information Retrieval and Machine Translation (Second Edition)	*(953-0300)*
List of Books on Computers	*(520-1588)*

Miscellaneous New Publications

The Role of Computers in Civil Engineering	*(520-1114)*
The Role of Computers in Electrical Engineering	*(520-1288)*
IBM *1800 for Wind Tunnel Data Acquisition*	*(E20-0271)*
Operating System/360 Job Control Language Charts	*(C28-6632)*
IBM *1080 Data Acquisition System for the Clinical Laboratory*	*(E20-0176)*
IBM *1800 Time-Sharing Executive System Operating Procedures*	*(C26-3754)*

IBM System/360 Operating System PL/I

Language Specifications (revision)	*(C28-6571-3)*
Subroutine Library Computational Subroutines	*(C28-6590-0)*
Programmers Guide	*(C28-6594-0)*

Catalogs of Programs — Supplements

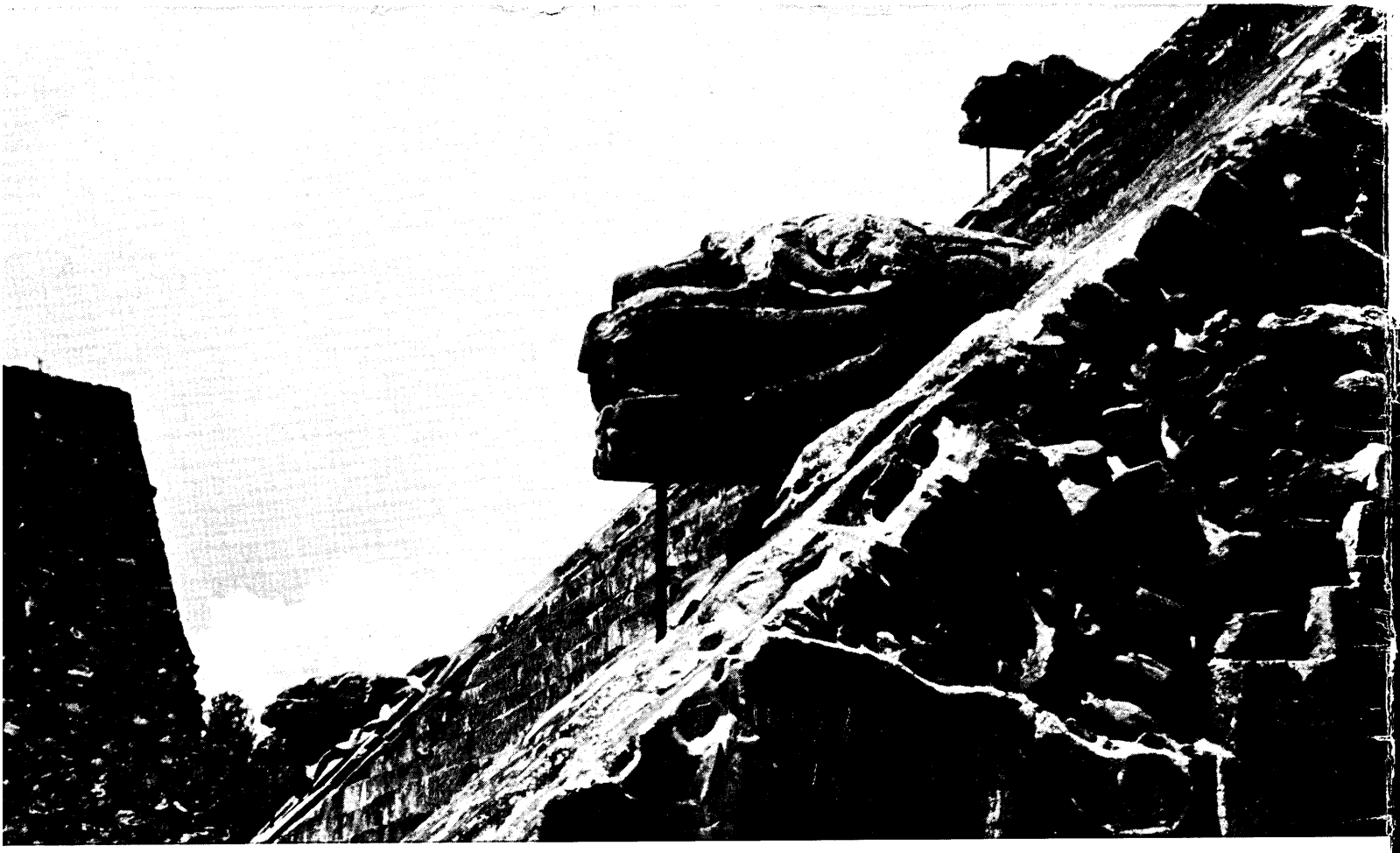
IBM *1240, 1401, 1420, 1440, 1460*	*(N20-0013-7)*
IBM *705, 1410, 7010, 7070, 7072, 7074, 7080, 7740, 7750*	*(N20-0014-7)*
IBM *1620, 1710*	*(N20-0015-7)*
IBM *704, 709, 7040, 7044, 7090, 7094*	*(N20-0016-7)*
IBM *System/360*	*(N20-0030-5)*
IBM *1130, 1800*	*(N20-0031-1)*

Revised Publications of General Interest

IBM *1130 Assembler Language*	*(C26-5927-2)*
System/360 Scientific Subroutine Package Application Description	*(H20-0166-2)*
Medical Information System Programs	*(H20-0182-1)*
System/360 Data Communication and Acquisition Configurator	*(A22-6824-3)*

IBM Journal of Research and Development

The IBM *Journal of Research and Development*, published bimonthly, is available to readers of *Computing Report* on a single issue or subscription basis. Articles contained in the current issue, Volume Ten, Number Four, include: Diagnosis of Automata Failures: A Calculus and a Method; A Numerical Analysis of the Transient Behavior of a Transistor Circuit; Properties of a Free, Steadily Travelling Electrical Domain in GaAs; Effect of Domain and Circuit Properties on Oscillations in GaAs; Resonant Excitation of Magnetostrictive Driven Print Wires for High-Speed Printing; A High-Speed Read Only Store Using Thick Magnetic Films; Dynamic Laser Wave-length Selection; Localized-Field Permanent Magnet Array for the Thick-Film Read Only Store; and Domain Wall Velocities in Thin Magnetic Films. Information about subscription rates may be obtained from the IBM *Journal of Research and Development* Circulation Department, IBM Corporation, Old Orchard Road, Armonk, New York 10504.



Pyramid probe in Mexico

Two thousand years ago, when Augustus was carving the profile of the Roman empire, the architects of the Teotihuacán civilization were sculpting two of the most awesome monuments yet discovered in Mexico—the Pyramids of the Sun and the Moon.

Set on a base of 10 acres, the Pyramid of the Sun soars to a height of 216 feet. But for all of its size, it reveals little of the Teotihuacáns, whose civilization flourished until 900 A.D. when it mysteriously ceased to exist.

Archeologists who have unearthed the remnants of this ancient city—including exquisite carvings, wall paintings and buildings—are now mapping a research path back into the past with the aid of computers.

Under a grant from the National Science Foundation and the Wenner-Gren Foundation for Anthropological Research, Dr. George L. Cowgill of Brandeis University will use an IBM 1620 and a 7094 to scrutinize data sifted from the ruins.

The major aim of the computer study is to see what insights computers can provide which traditional methods would overlook. "We are especially interested," says Dr. Cowgill, "that these procedures will add importantly to our reconstructions of the culture and social organization of Teotihuacán."

International Business Machines Corporation
Data Processing Division
112 East Post Road
White Plains, N. Y. 10601

Return Requested