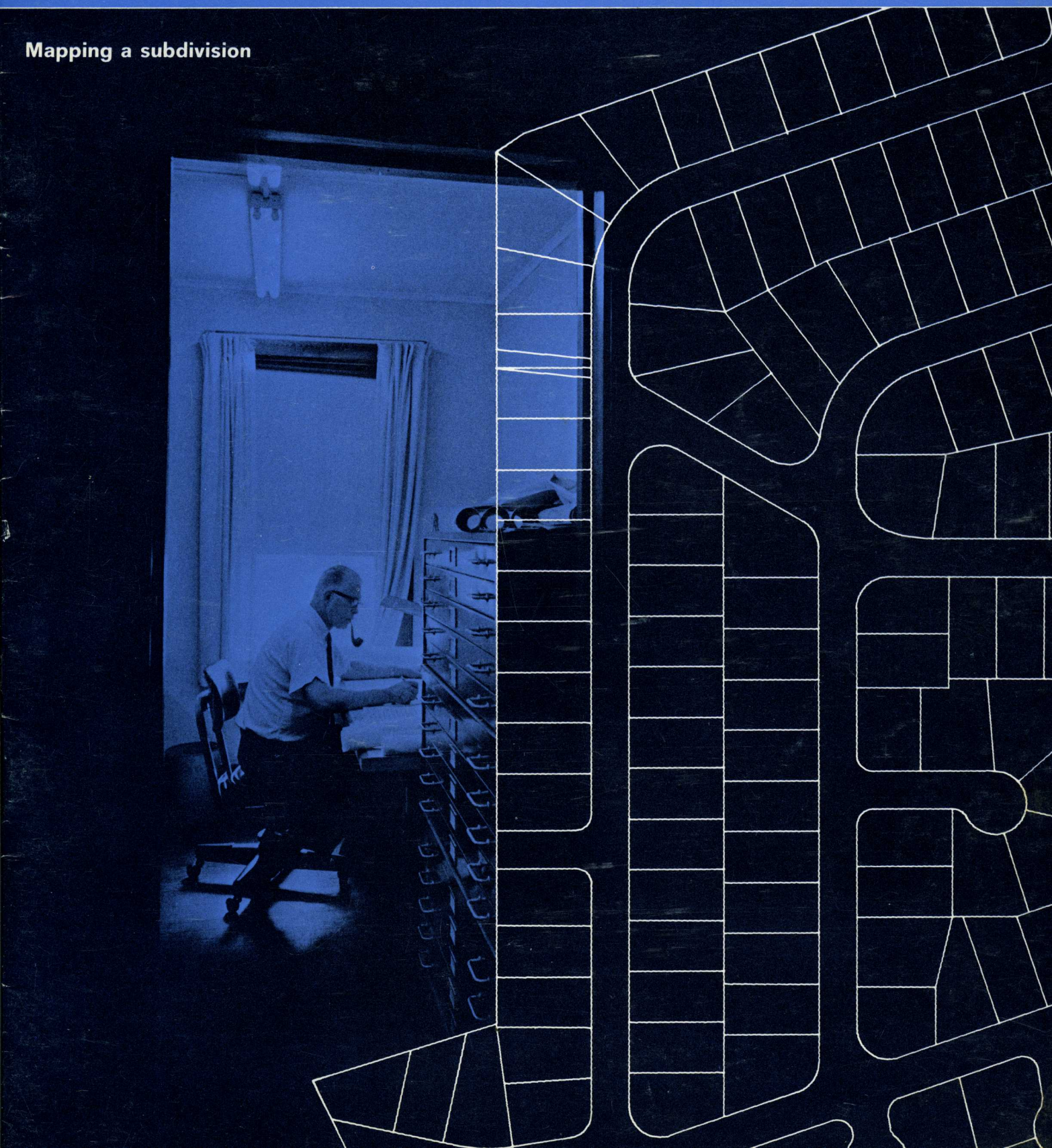
IBM

computing report

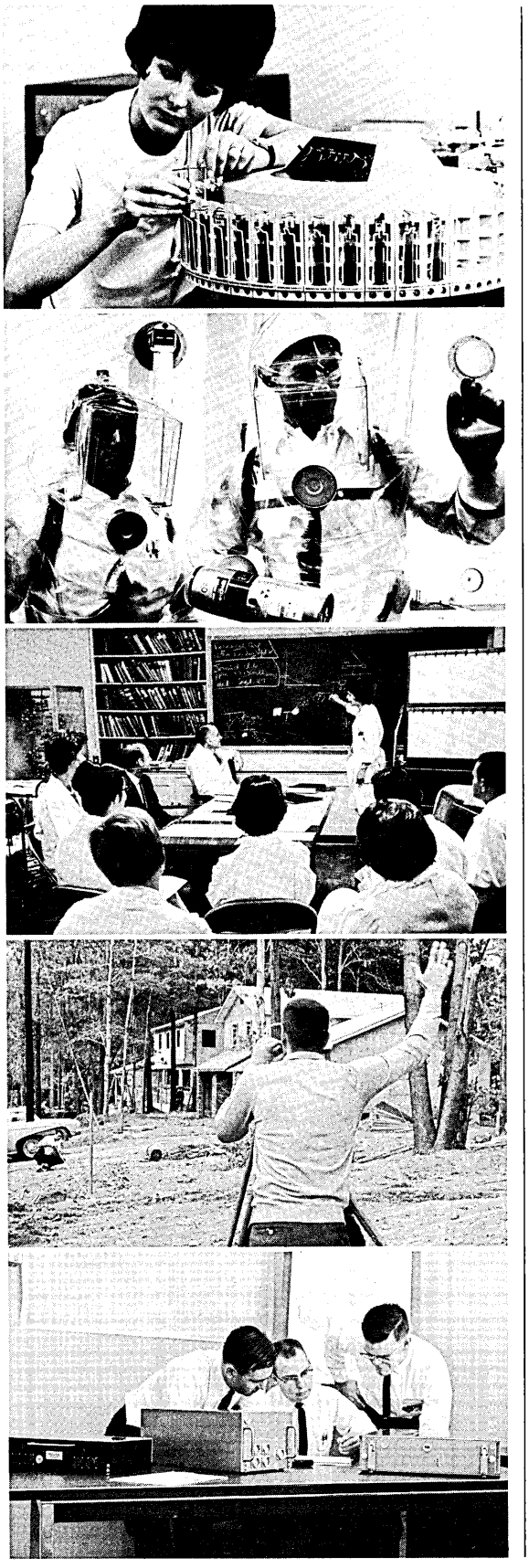
for the Scientist and Engineer

November 1967, Vol. III, No. 6

Mapping a subdivision



Contents:



Cover: Mapping a subdivision

The world's burgeoning population is creating an ever-increasing demand for new housing. One way the demand is being met is by subdividing large tracts of land for housing developments. But subdividing land into building lots of appropriate size and shape, each facing on a street, presents an intricate mathematical challenge. To meet the challenge, one engineering firm generates subdivision maps with a computer-driven plotter. See "PLAN for a housing development," page 10.

Picture credits: Cover, 2, 8, 9, 11, 12, 13, Stuart Smith; 2, 20, Ulrich Zimmermann—Euratom; 4, Fabian Bachrach.

November 1967, Vol. III, No. 6
Computing Report
for the Scientist and Engineer

Copyright 1967
International Business Machines Corporation

Published bimonthly by
Data Processing Division
International Business Machines Corporation
112 East Post Road
White Plains, New York 10601

Editor	D. T. Sanders
Feature editor	R. A. Sagar
Technical editor	R. F. Steinhart
Editorial associate	K. M. Joyce
Circulation	J. T. Brady

The status of software automation

by Dr. E. G. Fubini

Engineering approaches and programming research are needed to cope with the volume of programming that must be done

Dr. Fubini is a vice president and group executive of IBM. *These remarks are from an address delivered on August 16, 1967, at the* SHARE XXIX *meeting in Miami Beach, Florida.*

The history of software development, almost from the very first, has been vigorous and hectic. Programs have always been produced under trying circumstances. The successes, however, have surely dwarfed the failures. And if the history of science and technology is any guide, some new directions can lead to even greater achievements.

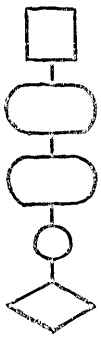
Programming today is more an art than a science, since each program must be individually crafted. We lack a clear set of rules as to how to obtain a computer solution to a problem. Moreover, when we have developed a technique to solve a problem, we don't know if it's correct. We can only keep trying the method, with empirical tests, until it fails. In other words, we can never be sure that we are right, but we will often find that we are wrong. The software design process is, today, very slow and quite inefficient because programmers cannot bring their intellect fully to bear on each problem. Too much detail is required to code programs, even in higher-level languages.

More effective programming

We must find a way to use programmers more effectively. Fortunately, I think there are several promising approaches to this problem. The approaches have, as a unifying theme, the need for a more basic understanding of programs that are written and of the nature of programming itself.

One clear sign of maturity in any developing technical discipline is that it begins to ask questions about itself with a view toward developing a formal system that can be applied to general cases. After all, one of the most important attributes of a formal discipline is the ability to codify and organize information so that knowledge can be passed on to the students of that discipline.

For example, reflect on the difficulty of teaching the *art* of programming. The mechanical details of coding are easy to transmit—but the substantive aspects of programming are beyond communication at present. The first phase of formalization will necessarily be highly empirical. A great deal of data must be gathered, then systematized and quantified so that patterns become apparent. From the patterns, a theoretical foundation may be developed.



We do have a great body of software today, but it has not been generated systematically. It is extremely difficult to analyze because it is not all in one place and the relevant documentation is largely missing. We can no longer think of software simply as a means of solving some experimentally generated problem. Somehow, the designs and the code must be examined under conditions removed from the hectic day-to-day urgencies of schedules, and in a coherent and controlled fashion, much as an experiment is observed in the laboratory.

We cannot let the gap increase between practical program production and the theoretical understanding of programs. To the contrary, we must make every effort to see that it decreases. We must do programming research. We must un-

derstand the theoretical limits of what programs can and cannot do.

In every discipline, a knowledge of theoretical limitations has always led to an enormously fruitful period in the development of that discipline. Such areas as thermodynamics, quantum mechanics, relativity, and information theory come to mind. Think of the advantages, for example, of being able to do for computer software what communication theory has done for information channels. Moreover, an understanding of the theoretical limitations and how close we are to them indicates the areas to which it would be most profitable to assign our resources.

"Every time a program is written, the programmer must essentially start *ab initio* with only his own experience to rely on."

It has been shown mathematically that we cannot prove whether any two arbitrarily chosen programs are equivalent—that is, whether they will produce the same result given the same input data. However, this does *not* mean that we *can't* prove the equivalence of two programs in specific cases. For what fraction of possible program pairs can equivalence be shown? Is it less than one per cent of all possible pairs? Greater than 99.9 per cent? If the ground rules are not too restrictive, it might prove worthwhile to write a program that would examine program pairs in an attempt to demonstrate equivalence. Such a facility would be enormously helpful in developing an understanding of the theoretical nature of programming.

Consider the situation in the physical sciences. An extremely fruitful body of theory was developed when physicists understood just enough about nature so that they could begin to structure it. The structuring had a snowballing effect, permitting more and more phenomena to be understood. The important ingredient in this understanding was the happy fact that every effect is the result of one cause or, at most, a few causes. When a scientist can say, "If I do this, then the following will happen," he is well on his way to fitting that phenomenon into the rest of the physical world. Note, however, that it becomes most important to know which things change and which don't when a deliberate change is made in the system. That's why the concept of program equivalence is so important. We need a yardstick by which we can measure whether our system is changing.

The mathematicians tell us, however, that equivalence proofs are "impossible" for nontrivial program pairs. If this is so, we are in real trouble. It means we will have to find some other way to detect changes in program function as a result of external stimuli. Perhaps we are asking the wrong question when we ask for proofs of program equivalence. Perhaps if we could change the question slightly we could get a rewarding answer. The problem is difficult, but the rewards are so great that I am sure it is worth trying.

Where do we stand today? Several decades of experience with computers and programming have led to a large body of experimental observations which as yet have no unifying thread. Every time a program is written, the programmer must essentially start *ab initio* with only his own experience to rely on. As I mentioned before, it is impossible today to write a textbook that will teach the art of programming. This situation guarantees that programmer effectiveness will be kept low and that the programs written will not be as powerful or as efficient as they could otherwise be.

Furthermore, it will be necessary to transfer a large part of the program-generating load to non-

Dr. E. G. Fubini

professional progammers who are interested in the results, but not particularly in the beauties, of programming. We will get to the point where individuals completely unfamiliar with computing systems or with programming can learn to use the machine, perhaps via a Socratic question-and-answer method or by other techniques utilizing such hardware devices as graphic displays. The distinction between programmers and non-programmers will tend to blur in the next few years, as techniques become available to allow computer users — without the benefit of program experience — to have the same control over machines that only programmers have now.

It is certainly not too much to expect that the computer could help such people write their programs, even if such a process were not economical of computer time, nor the resulting code very efficient. The development of FORTRAN, COBOL, and PL/I are, of course, the first steps in this direction. These languages represent the use of the power of the computer to simplify programming tasks.

"One thing is clear: the highly interactive nature of software and hardware means that we can no longer consider these areas separately."

I would call this an *engineering* approach to the problem of software generation. The compiler does not represent any breakthrough in the fundamental *understanding* of software itself. It is, of course, a magnificent tool. Using compilers, computer installations increase program productivity, reduce the amount of detail involved in writing programs, make computing techniques available to thousands of scientists and engineers who otherwise might not bother, and reduce debugging time.

One hears complaints that compiled codes are not as efficient in execution as some hand coded machine-language programs. That is true, but the industry can no longer afford the luxury of custom-designed programs any more than the majority of us can afford custom-made clothes or custom-built cars. This trend must be continued, or we will need too many programmers. We must move from the artisan's shop to the production line and sacrifice program craftsmanship for programming volume. We will need many more engineering solutions to programming problems if we are ever to cope with the volume of programming that must be done. It is clear that, in absolute terms, the amount of creative programming must continue to increase as well; but the gap between what we can do with present techniques and what we will be required to do must be narrowed by a more mechanical treatment of the generation process.

Engineering approaches

What kinds of engineering approaches are anticipated? A good example is a program management system for control. This would be a self-auditing software system capable of giving program supervisors the information they need to intelligently manage a project. Obviously, a great deal of information must be on file and accessible to such a program. That information would include the program's status, its specifications, its actual machine code, test cases, and the program manuals. An information retrieval system that can provide up-to-the-minute information must be available to control costs and maintain schedules.

A terminal-oriented production system is well on the way to implementation and will provide such monitoring features. Aside from storing and processing all significant documentation, it will:

- keep track of multiple versions of systems
- organize, partition, and control design data so that the status and consistency of the design may be clearly understood
- highlight redundancy and conflict in the specification of design data so that corrective action can be taken
- reduce the possibility that detailed implementation decisions will prevent the attainment of initial high-level specifications.

Such a system will, in itself, reduce much of the clerical and routine work of our best programmers and permit them to spend more time on the truly creative parts of programming systems.

By giving all coders access to the software file, we will permit every programmer to be fully aware of the current status of his work as well as the current status of all other codes affecting his particular part of the program. This will reduce the enormous communication problems we have experienced when creating such large system programs as OS/360. However, it must be kept in mind that such a system will not solve all of our programming problems.

Programming research

No matter how ingenious and useful are such engineering aids, they will never move us toward a fundamental understanding of the nature of programs and programming. We still know less than

we need to know about the purely mechanical operation of converting a feasible method of solving a problem to a working program capable of handling real data and producing desired results. The distinction is precisely the same as the one between engineering and research. An engineering solution to a problem implies that we take a known technique and apply it to a given problem. The solution of a given problem does not help us to solve the next one, unless it is highly analogous to the first.

When we think of research, we think of investigations that proceed without highly specific problems in mind. By developing a body of fundamental knowledge and understanding about the physical world or about mathematics, we can more readily develop specific engineering techniques. The same, I am sure, will hold true for programming research. To do research, we must divorce ourselves from the day-to-day programming problems we are faced with and look at the broader aspects of what is today an art.

What, then, are the areas that we must explore in depth? Unfortunately, since we are starting from very little, we don't even know what we don't know and should know. One thing is clear: the highly interactive nature of software and hardware means that we can no longer consider these areas separately, but must, from now on, look at the joint hardware-software complex. In general,

"It will be necessary to transfer a large part of the program-generating load to nonprofessional programmers who are interested in the results, but not particularly in the beauties, of programming."

we must analyze this complex by establishing experimental methods that can be used for hypothesis generation and testing.

I am convinced that programming research will have at least these two central themes: the need to define the fundamental limitations on the performance of the processes we are interested in, and the need to remove what we have called craftsmanship in programming. The first theme is related to the need to establish a structure for the world of software. This need is analogous to the physicist's need to determine what the rules of the real world are so he can use them in exploring the microscopic and macroscopic aspects of the universe. One of the basic problems with software today is that it is too unstructured to lend itself to such analysis. Knowledge of the limitations of the processes being manipulated by the hardware-software complex can lead to such structuring.

The second theme is related to the need to reduce the artistic content of programming activity. The task of programming must become a more mechanical operation. For example, it would be highly desirable to have software take over the job of optimizing the work produced by a programmer. Although optimizing techniques do exist in compilers, the problem of optimizing over the entire hardware-software complex is as yet unsolved. The problem is so large that it is unlikely to be solved by the *ad hoc* techniques used in current compilers.

"On the horizon is a whole battery of computer-assisted techniques which, it is hoped, will make the programmer's lot a great deal easier."

In summary

Programming today is an art. With each new problem, the programmer must begin all over again to develop a software solution. He has little in the way of computer-assisted techniques to help him design the program, predict its performance, test it, code it, and debug it. Such an approach is no longer tenable with the rapidly growing demand for programmers and their much more slowly increasing supply. On the horizon is a whole battery of computer-assisted techniques which, it is hoped, will make the programmer's lot a great deal easier. This does not, however, move us any farther toward a fundamental understanding of programs, particularly of the limitations we are going to find. Such an understanding can come only with a wholehearted commitment to programming research, so that we can comprehend the theoretical aspects of software and its generation. The history of other disciplines indicates that this is where we will hit the jackpot. □

Tracing genetic diseases

By studying inherited diseases in inbred groups of people, scientists can learn what is genetically normal

A person's genetic makeup usually gets little clinical attention. Genetically, he is what he is because of the inheritance passed down to him, from his parents and their parents before them, through 46 tiny chromosomes in each of the cells of his body. The chromosomes—or more precisely the *genes* on the chromosomes—determine such characteristics as the type of blood in his veins, the color of his hair and eyes, and whether he is tall or short, fat or thin.

Genetic disorders

But an abnormality in a part of one chromosome can make a person anemic, or hemophilic, or colorblind. If some factor is missing in his genetic makeup, he may be mentally deficient. Another fault in his genes can make him a dwarf. Yet another fault may cause his muscular system to fail to develop and he will die young.

Such genetic diseases have been widely recognized in man only within the last 20 to 30 years. Hundreds have now been identified, some occurring commonly, and many of great medical significance. It has been estimated that one person in 25 may be born with a significant genetic defect.

It is a monumental task to trace such diseases in the general population. In the normal world where there is little intermarriage among relatives, few individuals with genetic diseases are related. Thus, for meaningful findings, the disease-inheritance patterns of many families must be combined —but the problem of managing such information is enormous. This is where the computer has come in—permitting the systematic organization and storage of genetic information for easier access by researchers.

One method of studying how human genes are passed from parents to offspring is to choose a group of people who have lived in an isolated community for several generations. A long-term investigation of one such group—the Old Order Amish—is under way at one of the world's leading centers of human genetics research, The Johns Hopkins University in Baltimore, Maryland. There, in a corner of the large hospital that is the heart of the School of Medicine, Dr. Victor McKusick directs the Amish studies. Dr. McKusick is investigating two major groups of Amish: one in Lancaster County in eastern Pennsylvania, the other in Holmes County in east-central Ohio. He is using IBM 7094 and 1401 computers to analyze his results.

Amish men and women marry within the group, and although members may leave the society, no outsiders enter it. Therefore the Amish are all more or less related, and the relationships can be extremely complex. It is not just a question of finding, for example, that a married couple are first cousins; they also may be second cousins through three different pathways and fourth cousins through half a dozen more. If there is a characteristic gene in such a population it tends to make its appearance—or "be expressed," as the geneticist puts it—more frequently than in the general population.

Such a gene for a particular variety of six-fingered dwarf exists among the Amish of Lancaster County. At least 52 known cases have appeared in the ten generations since the first

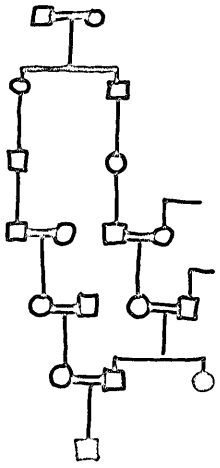
A catalog of diseases

Dr. McKusick has used the IBM 1401 computer at Johns Hopkins Hospital to compile an unusual book—a catalog of hereditary diseases, complete with pertinent genetic information and references to the literature on each disease. The book, published in 1966 by The Johns Hopkins Press, is entitled *Mendelian Inheritance in Man*. It is divided into three main sections dealing, respectively, with genetically dominant disorders, genetically recessive disorders, and disorders linked to the X chromosome (the female sex chromosome). In addition, all entries are cross indexed by subject and author.

The entire content of the book (exclusive of the foreword and other front matter) is stored on magnetic tape, and the 344 catalog and index pages consist of photo-offset copies of 1401 printout. "The information contained in the book will be kept updated on the tape," says Dr. McKusick, "and new editions will be printed whenever the number of changes warrants it."

members of the group came to America in 1720. Dr. McKusick and his staff have traced the ancestry of every member of this group back to the original immigrants, and they have stored the entire genealogy on magnetic tape for use in the 7094 Data Processing System.

To code the information for the computer, Dr. McKusick and his staff assigned a number to each individual—from those living today back to their earliest known ancestors in America. Each person's number can be entered into storage along with the numbers representing his parents. For example, the entry for the person denoted by number 12 might be 123456, meaning that number 12's father is designated by number 34, and his mother by number 56. Similarly, the entries for the father and mother include the numbers for *their* parents, and so on back through the generations. Thus in seconds the computer, going through a process somewhat resembling a chain reaction, can search sequentially from generation to generation and print out the entire known genealogy of any person in the Amish group.



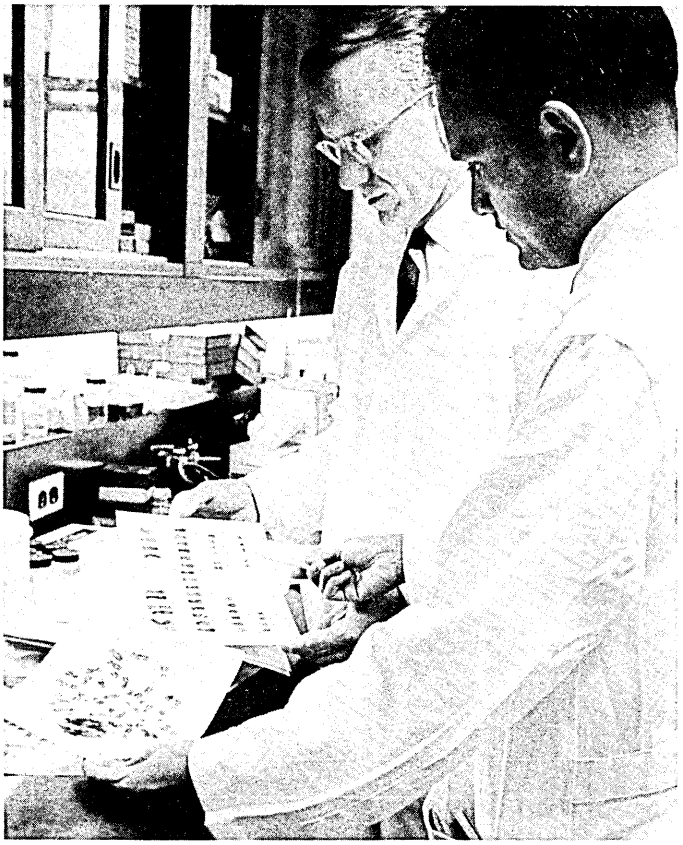
Genealogy

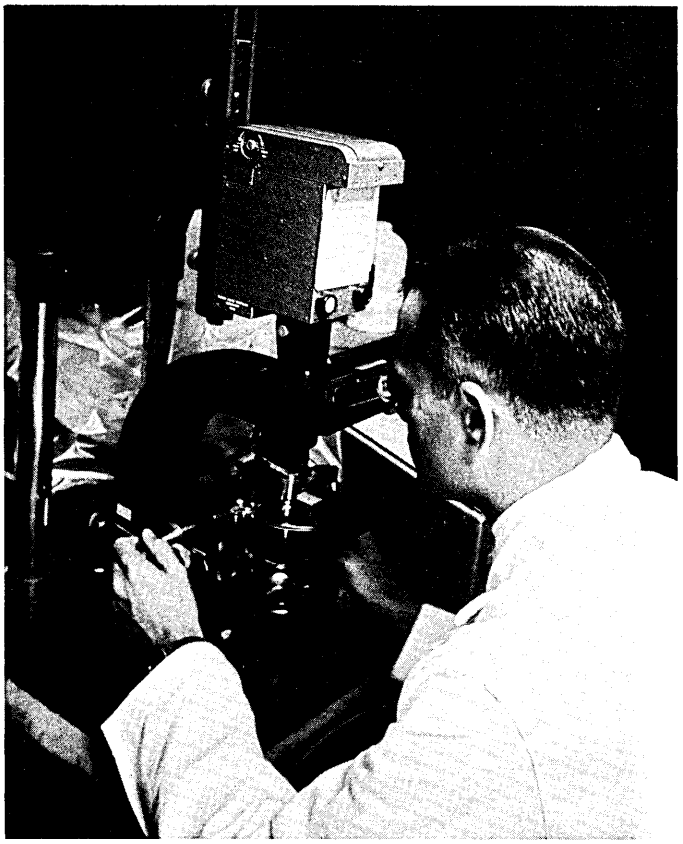
"The computer makes the data easy to keep up to date and to use for analysis," Dr. McKusick says. By reconstructing the genealogy of affected persons, he has traced the dwarfism gene back to one Samuel Koenig, or King, who came to Amish country in 1774. Either he or his wife brought the gene to America.

Another type of dwarfism appears in an unrelated branch of the Amish living in Holmes County, Ohio. This defect is entirely different genetically from the one that affects the Pennsylvania community.

Other such disorders may be discovered and their genealogy plotted, but simply hunting up "new" genetic diseases is not the point of Dr. McKusick's work. It is by learning about abnormal conditions that geneticists find out what is genetically normal. For example, the fact that hemophilia exists demonstrates that the ability to clot must be a normal factor in blood.

Explains Dr. McKusick, "The simply inherited genetic diseases in man are like photographic negatives from which positive pictures of his normal genetic constitution can be constructed." ▫





CHROMOSOME PHOTOMICROGRAPHS are studied by Dr. Victor A. McKusick and Dr. D. S. Borgaonkar (picture at top of page). The photomicrographs, used in genetics research, are obtained with the device shown at right.

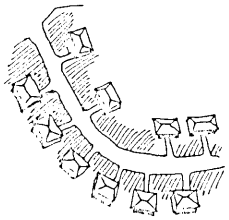
STAFF MEETING (opposite) is called by Dr. McKusick, at head of table, to discuss X rays showing genetic disorder.

PLAN for a housing development

1130 COGO and PLAN produce computer-plotted maps to speed civil-engineering work in land subdivision

To cope with staggering workloads created by the growing demand for new housing, the civil-engineering firm of C. T. Male Associates has turned to the computer to speed the basic work of site development and land subdivision. Headquarters of this small, family-owned company is an old farmhouse in the town of Niskayuna, near Albany, New York.

In a corner of what once was the kitchen of the old house, an IBM 1130 Computing System busily divides land parcels into building lots, lays out roads and sewer lines, and calculates acreage and angles, curvatures and coordinates. In short, the computer makes all the basic computations that are required before the builder can turn his first shovelful of earth.



Computer applications of this kind are not uncommon, but Charles T. Male Jr., one of the partners in the firm, and chief programmer James Ballentine have added significant innovations. One innovation is the form of the computer's output. In addition to reeling off tabular printout of survey data, which in most installations would be laboriously converted to hand-drafted maps, the 1130 is linked to an IBM 1627 Plotter that turns out subdivision maps automatically.

PLAN with COGO

Equally remarkable is the input. The computer gets raw data and instructions in the firm's own computer lingo. "I wouldn't call it our own *language,*" says Mr. Ballentine, "just our local dialect." Basically, the "dialect" is a form of COGO (the MIT-developed COordinate GeOmetry language that is couched largely in common civil-engineering terms) modified by PLAN (IBM's Problem Language ANalyzer). With graphic output from the plotter, the COGO-PLAN combination provides notable gains in operating efficiency.

The most obvious advantage lies in the visual output, which makes it unnecessary for draftsmen to produce maps by painstaking, point-by-point plotting from tabular printout. The computer-plotted maps can be used directly by stake-out crews, since the computer can be instructed (by a separate subroutine) to insert all necessary alphanumeric information. To satisfy legal requirements for heavy, linen-backed maps for use as land registry records, however, final maps are drafted by tracing the computer output.

At a single command, the computer can change the scale of a plotter-produced map to suit any need. For example, the plotter may be drawing a small-scale reference map showing the overall pattern of a subdivision. A moment later, without extensive reprogramming, it can be made to produce large-scale maps of single building lots within the subdivision, or detailed road sections. Moreover, the computer can mathematically rotate an entire plot in increments of 5 degrees, permitting odd shapes to be fitted conveniently onto the 29-inch-wide plotter strip.

Many advantages

Pulling a 15-foot-long map from the plotter, Mr. Male casts a practiced eye on an intricate pattern of subdivision streets with circular turnarounds. "Now we can make immediate visual checks of a layout," he observes. "If some lots come out with awkward shapes or odd angles, we can see it right away. Before we had the plotter, we wouldn't notice those things until after our draftsmen had converted the tabular printout to hand-drawn maps."

Prompt layout verification is but one of many advantages inherent in the computer-plotter complex. Among others:

- Area determination for individual lots. "Before," says Mr. Male, "we couldn't take the time to figure individual lot areas, especially for odd-shaped lots with curved boundaries. Now the area is a routine part of the printout."
- Better traverse closure and balancing. "We let the computer analyze our field traverse data. It automatically adjusts angles and distances according to accepted methods, compensating for the inherent instrument and human errors that occur in field work. The result is increased accuracy, which is reflected in subsequent surveying applications." Precise traverse closure and balancing is not usually attempted in routine survey work because it is too laborious. "Now," Mr. Male points out,

SURVEYORS check property line in computer-planned subdivision.

What is PLAN?

PLAN (Problem Language ANalyzer) is a set of programs used by IBM to facilitate the incorporation of problem-oriented languages into application programs. It has been used in the development of three available 1130 programs: the Data Presentation System, Mechanism Design System—Gears and Springs, and Work Measurement Aids. Although PLAN is not distributed separately, it is part of these three programs. Therefore its capabilities can be used as desired. Its open-ended design makes feasible the kind of application developed at C. T. Male Associates.

PLAN will be the subject of a panel discussion at the 6th meeting of CEPA, the Civil Engineering Program Applications group, to be held at the Warwick Hotel, New York City, on November 20 and 21. Moderator of the discussion will be Mr. J. G. Sams of IBM, who was primarily responsible for developing PLAN. For further information on the meeting, please contact Mr. M. H. Eligator in care of Weiskopf & Pickworth, 230 Park Avenue, New York, N. Y. 10017.

PRINTOUT OF SURVEY DATA is reviewed by Charles T. Male Jr. (left) and chief programmer James Ballentine.

"the computer enables us to add this service at no extra cost."

- Faster reduction of survey data. The firm's surveyors sometimes measure distances with a geodometer, an electronic device that is used instead of a surveyor's tape when greater accuracy is required. "With a desk calculator it used to take us about twenty minutes to interpret a single geodometer reading," says Mr. Male. "With the computer we can do ten readings a minute."
- Compact data storage. All the data pertaining to each job (including coordinate points) is stored on a magnetic disk as a permanent record for future reference. Any item—the size of a lot, the width of a street, an overall view— can be extracted quickly from the disk.

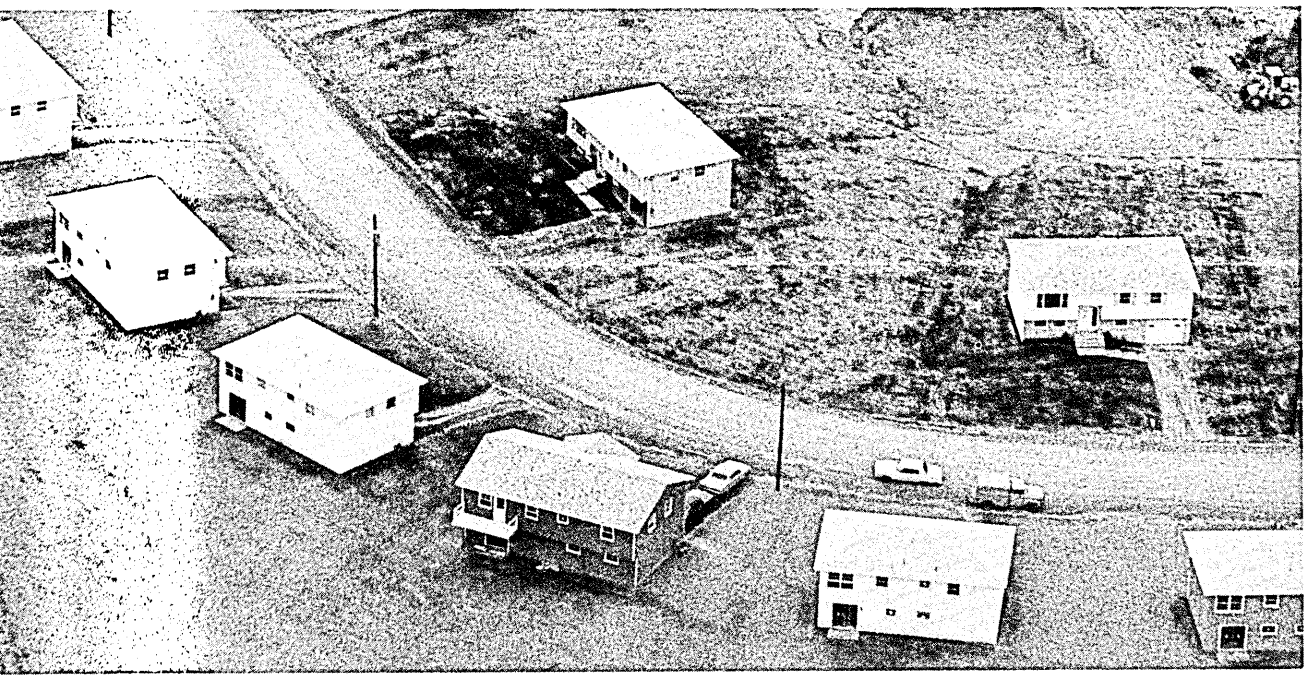
Economically, the most important advantage is the time saved in determining the boundaries of individual building lots. Mr. Male recalls that it usually took about 15 minutes to compute one lot with a desk calculator. The 1130 does the job in less than half a minute, including the previously omitted area determination. Hence subdivision computational costs have dropped significantly.

A larger repertoire

The chief aim of the firm's software approach has been to simplify and speed up programming functions by creating a more inclusive command repertoire. With COGO, they already had a language geared to civil-engineering terms and problems. COGO proved so readily comprehensible, in fact, that Mr. Male, without previous computer experience, was able to program all of his routine operations. Subsequently, he and Mr. Ballentine developed additional macro instructions within the COGO framework. For example, at a single command an engineer can instruct the program to automatically adjust a lot boundary to obtain a specified area.

Another modification provides tabular readout in ordinary surveyor's terms, rather than in the "quadrants" of the original COGO. Mr. Male and Mr. Ballentine also implemented the COGO concept under a special version of FORTRAN to gain speed and to increase the number of survey points they could store in the 1620 computer that preceded the firm's 1130. The arrival of the 1130, with its disk file, again increased the number of storable points. Through modified COGO programming, up to 5,000 coordinate points can now be stored for a given job.

After the modified COGO program had been perfected, Mr. Ballentine rewrote it so that it could operate under PLAN, which might be considered an intermediate step between a monitor and a program. One of its functions is to let the user write his own commands in terms of more in-



NEW HOUSING DEVELOPMENT near Schenectady, New York, designed by C. T. Male Associates. IBM 1130 computer calculated locations of streets and property lines, and attached plotter created maps drawn to any required scale.

clusive macro instructions. Another function is to save execution time by the faster loading of subroutines into core storage. The principle behind this function is "core image loading," in which various subroutines are preassembled on a disk in exactly the format they will have in central-processor storage. The subroutines then can be transferred quickly from disk to core, eliminating the time-consuming building of subroutines and searching in different locations on the disk.

The COGO-PLAN combination yields remarkable efficiency. System overhead for core-loading a given subroutine into the 1130 has been reduced from one minute to two or three seconds. System throughput has been increased 50 per cent. These improvements, in turn, have resulted in other operating economies. Engineers and draftsmen, for example, no longer have to wait for data to be processed by slower routines, and plotting time has also been decreased.

Engineering efficiencies

Mr. Ballentine points out that important civil-engineering efficiencies are inherent in the design of the 1130. "Thanks to its disk file," he explains, "alternate plot patterns for a given land parcel can be produced quickly. Before we had the 1130, every time we wanted to change a layout pattern we had to feed in the whole card stack that represented the raw coordinate points. Now this information is permanently stored on a disk with direct access, allowing us to run pattern changes without feeding the primary data into the machine for each change. As a result, our engineers can use the 1130 almost in a conversational mode."

Encouraged by their success with the computer, C. T. Male Associates are planning to incorporate a new package into their operating routine. By adding IBM's Numerical Surface Techniques and Contour Map Plotting program, they hope to have the plotter overlay topographic information on subdivision maps, drawing in contour lines descriptive of the terrain. With input data derived largely from field information, the output, at a glance, would reveal the suitability of a given area for its intended use.

In addition to its engineering functions, the plotter-equipped 1130 is programmed under IBM's Data Presentation System to draw bar graphs for management information. Basically, this is a cost-accounting function in which the graphs represent man-hours allocated to various aspects of the total business operation. Further, the firm looks forward to extending the use of its 1130 into bookkeeping and, eventually, into performing consulting services for building contractors, advising them on optimum use of money, manpower, and equipment. ▫

Around the globe with 4 Pi

System/4 Pi, IBM's new family of computers, is designed especially for aerospace applications

It has not been so many years since the auxiliary equipment in a military airplane consisted only of flight instruments and weaponry. But times have changed, and the traditional fighter-plane image has gone the way of the pilot's white scarf and goggles. Today's supersonic combat plane is crammed with more instruments, indicators, and controls than the pilot can manage by himself.

For example, the U.S. Air Force F-111, the new swing-wing fighter-bomber currently being tested, is equipped with highly sophisticated navigation facilities and radar-guided attack aids. These devices, along with the electronic cockpit displays tied in with them, make up an integrated avionics system which, because of its complexity, is based on a pair of specially designed digital computers. The computers are part of the recently announced IBM System/4 Pi family (see "For further reference," page 19). They are designed as multi-mission devices, making possible a high degree of standardization. One of the computers in the F-111 is used for weapons delivery, the other for navigation.

What is System/4 Pi?

The name "4 Pi" bears a relationship to IBM's System/360. Just as "360" refers to the number of degrees in a circle, "4 Pi" refers to the number of steradians in a sphere—suggesting a full sphere of performance characteristics. Read-only storage can be used in either system to control logic operations, making it possible for these general-purpose computers to be adapted to specialized applications without rebuilding their logic circuitry.

Aerospace applications are extremely specialized. Often they require custom-built computers, or at least radically different configurations of standard computers, with the higher costs that inevitably are associated with non-standard equipment. Therefore IBM set out to produce a computer that could perform many tasks with little or no modification. The result was System/4 Pi, which consists of three basic models ranging from a low-cost tactical computer to a highly sophisticated, yet compact, general-purpose processor for handling large quantities of data at extremely high speeds.

The smallest model, the TC (Tactical Computer), is a lightweight computer designed for such jobs as weapons delivery and navigation. It can be used to guide tactical missiles, to control satellites, to perform tasks involving portable field equipment, and in the guidance systems of undersea search craft.

The intermediate model, the CP (Customized Processor), is designed for the guidance and control of aircraft and space vehicles and for battlefield information control systems. Some versions can be tailored to specific jobs by making use of a plug-in read-only storage element.

The most powerful System/4 Pi computer, the EP (Extended Performance) model, has an extremely fast processing capability for such applications as the surveillance and control of high-speed vehicles in an air-defense system. Yet it is compact enough for airborne and space missions. Read-only storage is an integral feature.

Packaging and input/output flexibility were given high priority in the development of System/4 Pi for an obvious reason: there is precious little room to spare in a space vehicle or a modern fighter plane. Small enough to fit into a suitcase, the new computers bear little resemblance to the ordinary data processing machine with its banks of switches, console lights, and tape units.

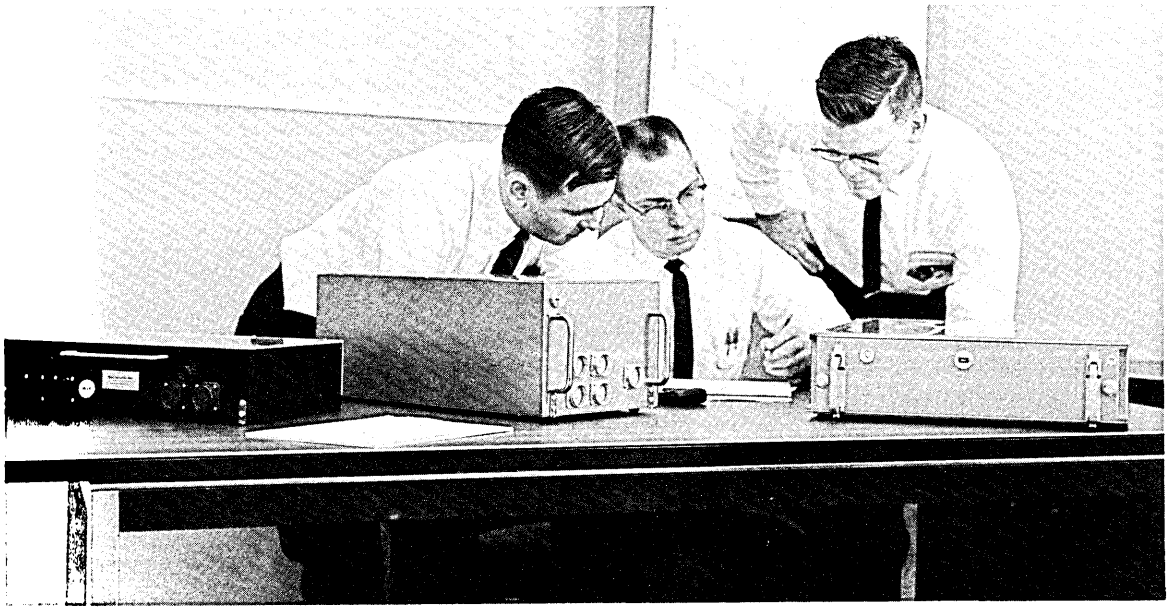
Monolithic circuits are used in all logic elements of System/4 Pi, permitting high-speed operation and providing compactness and reliability. Off-the-shelf TTL (transistor-transistor-logic) circuits are used, and the circuitry is packaged in sub-assemblies that can be plugged into one another.

All models of the System/4 Pi family are packaged in aluminum structures that can withstand temperature extremes ranging from −55 degrees to +100 degrees centigrade, as well as the stresses associated with the G forces encountered in satellites and airplanes and the bumping of a jeep traveling rapidly over rough terrain. In short, System/4 Pi is designed to operate under rugged field conditions.

System/4 Pi Characteristics

	Small Model TC	Intermediate Model CP	Large Model EP
Organization:	general-purpose, parallel	general-purpose parallel	general-purpose parallel
Storage:	8,192 8-bit words, expandable to 65,536 words	8,192 32-bit words, expandable in 8,192-word increments	16,384 32-bit words, expandable in 8,192-word increments
Storage cycle:	2.5 Usec.	2.5 Usec.	2.5 Usec.
Arithmetic:	2's complement, fixed-point	2's complement fixed-point	2's complement, fixed-point; floating-point option
Data word length:	16 or 32 bits	16 or 32 bits	16, 32 or 64 bits
Instruction word length:	8 or 16 bits	16 or 32 bits	16, 32 or 48 bits
Logic control:	conventional	conventional or read-only storage	read-only storage
Execution times:	add: 9-18 Usec. multiply: 48-54 Usec.	add: 5-10 Usec. multiply: 29.58—34.58 Usec.*	add: 2.1—5 Usec. multiply: 9.2—10.4 Usec.
Weight:	17.5 lbs.	50 lbs.	75 lbs.
Power	56W	250W	365W
Volume:	0.38 cu. ft.	0.86 cu. ft.	1.88 cu. ft.

*Assumes read-only storage



THREE BASIC MODELS of System/4 Pi are (left to right) the small TC, the intermediate CP, and the powerful EP.

Index to Volume III

Back issues of *Computing Report* can be obtained through local IBM sales offices.

Newsfronts

On-line scientific applications using the System/360 Model 44 computer will receive additional support with the new IBM Data Acquisition Multiprogramming System (DAMPS), scheduled to become available late in 1968. DAMPS is particularly useful for real-time applications requiring quick response to multiple external processes. In addition to the devices supported by the Model 44 Programming System, DAMPS supports 32 levels of the Priority Interrupt Special Feature and the IBM 1827 Data Control Unit for interfacing directly with such applications as wind tunnels, cyclotrons, and analog computers. The new system will support the assignment of program tasks to various priority levels.

Maps of congested urban areas, for use in urban planning, have been prepared experimentally using a Geo Space DP-203 Digital Plotter connected to an IBM System/360 Model 50 computer. Results of the project were reported to the annual meeting of the Urban and Regional Information Systems Association, September 7 through 9, in Garden City, New York, by R. G. Loomis and J. J. Lorenzo of IBM's New York Scientific Center.

Data used in the experiment was provided by the New Haven Census Use Study, which included the streets, blocks, and census tracts of West Haven, Connecticut, as well as a calculated block center and the number of housing units for each block. The paper, entitled "Experiments in mapping with a Geo Space Plotter," will be published in the proceedings of the meeting.

Transition from a 1620 computer (Model 1 or 2) to a System/360 Model 44 is aided by the System/360 Model 44 1620 Simulator program, which is scheduled to become available in the fourth quarter of 1967. Without additional hardware, the new program enables programs that have been operating on the 1620 to be executed on any suitable System/360 Model 44 configuration. By providing program compatibility, the simulator relieves reprogramming schedules, and it can eliminate the need to convert infrequently used programs.

Three-dimensional pictures are being created by an IBM System/360 Model 50 computer at Brown University. Charles Strauss (above), a graduate student in the Division of Applied Mathematics, has programmed the computer to produce, on the screen of an IBM 2250 Display Unit, a pair of images that are similar but that differ slightly in perspective. As viewed through a stereoscope, the images merge into a single three-dimensional figure. The images can be enlarged, reduced, or rotated on the screen, and they can be altered by using a light pen.

A laser aiming simulation study has been undertaken by IBM for the National Aeronautics and Space Administration (NASA). The study involves formulating a mathematical model by means of which a computer can analyze and interface with a laser communications experiment proposed for NASA's Apollo Applications Program. The model, a mathematical representation of the equipment and environment, will simulate the conditions expected in orbit.

The purpose of the experiment is to evaluate the feasibility of optical communication in space. Compared with radio equipment, laser-beam equipment is lighter in weight and, because of the laser's higher frequency, it can transmit data

faster. The study will be conducted at IBM's Space Systems Center in Huntsville, Alabama.

LEEP, which stands for Local Government Engineering Exchange Program, is the newest of a growing number of computer user groups—organizations formed to provide a free exchange of information, ideas, and programs among IBM customers. Members of LEEP specialize in public-works applications such as roadway design, air pollution data reduction and analysis, filtration plant control, and project cost estimating. A LEEP newsletter is mailed to members every four months to keep them informed of meetings and other activities. Readers desiring further information should write to Mr. R. J. Hansen, IBM Corporation, 1120 Connecticut Avenue, N.W., Washington, D.C. 20036.

A new graphics display terminal, designed for engineering design and analysis as well as for complex information displays, will be available in the third quarter of 1968 for use with the IBM 1130 Computing System. The terminal, called the IBM 2250 Model 4 Display Unit, enables scientists and engineers to exchange information with the computer in the form of charts, diagrams, drawings, or printed letters. With an electronic light pen, users can revise images, add or delete lines, modify curves, or change dimensions directly on the face of a cathode-ray tube. The 1130/2250 configuration can be a stand-alone installation, or it can be linked by communication lines to a System/360 computer when greater computing power is needed.

Time standards in industrial engineering can be developed quickly and easily with the aid of the IBM 1130 Work Measurement Aids package, which recently became available to users of the 1130 Computing System. The package consists of two programs: (1) a Machineability program uses local machine-tool and material data to determine optimum machine-tool parameters such as speed, horsepower, tool life, and process time for various machining operations, and it can produce routine output that is compatible with the System/360 Bill of Material Processor; and (2) a Work Measurement Sampling program determines job standards for long-cycle operations (over 15 minutes) from work measurement sampling observations, and it can also be used for conventional work sampling—that is, for distributing time to job activities.

The number of blood tests performed each day at the Youngstown (Ohio) Hospital Association has been increased with the help of a new system that couples computers with automatic testing instruments. Blood specimens and identifying punched cards are inserted into a carousel-like sampling device (above) which is a component of IBM's 1080 Data Acquisition System. The 1080 monitors both automated and manually operated laboratory instruments that perform many different tests on each sample, and it plots the results on a graph for visual inspection. The results, in punched-card form, are also entered into a System/360 Model 30 computer, which prepares a laboratory report for each patient's records. A test can be run and the results recorded in the computer within minutes after a blood specimen is placed in the sampling unit, and up to 60 specimens can be analyzed in an hour.

For further reference

Publications available on request from local IBM sales offices

Listed below, with their form numbers, are publications pertaining to topics covered in this issue of *Computing Report,* along with some new or revised publications of general interest. Unless otherwise noted, all the publications listed are available from local IBM sales offices. Readers desiring further information on subjects covered in *Computing Report* should address inquiries to local IBM offices.

Programming

Catalog of programs: IBM *1130 and 1800* *C20-1630-2*

Catalog of programs: IBM *System/360* *C20-1619-3*

Programming: Words that move machines *520-1498-0*

Bibliography of data processing techniques *F20-8172-4*

A sense of customers' needs: Programs and program information *320-0945*

PL/I . . . One programming language linking computer and user *520-1926-0*

Control

Nine papers on various problems in control theory and its applications are included in Volume 11, Number 4, of the IBM *Journal of research and development* (July 1967). *322-0035*

The titles and authors are:

"A generalized Legendre-Clebsch condition for the singular cases of optimal control," *H. M. Robbins*

"Rapid computation of optimal trajectories," *K. R. Brown, Jr., and G. W. Johnson*

"Some results in two-point boundary value problems," *S. M. Roberts and J. S. Shipman*

"Computer control of a paper machine—an application of linear stochastic control theory," *K. J. Aström*

"On-line identification of process dynamics," *E. B. Dahlin*

"Sensitivity constrained optimal control synthesis," *W. J. Budurka*

"Synchronization of traffic signals in grid networks," *A. Chang*

"Design principles for sampled-data systems with application to attitude control of a large, flexible booster," *L. A. Gilson, E. F. Harrold, and F. G. Kilmer*

"A computer-operated manufacturing and test system," *J. E. Stuehler and R. V. Watkins*

Environmental sciences

Proceedings of the IBM *Scientific Computing Symposium on Environmental Sciences* *320-1951*

Contents of the *Proceedings* are as follows:

Keynote address, *Sir Edward C. Bullard*

Session I: Modern Concepts in Data Collection

"Satellite data collection," *C. L. Bristor*

"Oceanographic data collection," *P. M. Wolff*

"Meteorological data collection" (not included), *H. A. Bedient*

"Seismic data collection," *P. Green*

Session II: Data Processing and Presentation

"Data processing for environmental research," *W. H. Haggard*

"Graphic displays generated from digital computers," *K. J. Berg and A. J. Schmitt*

"Space sciences data processing," *G. H. Ludwig*

Session III: Statistical Methods

"Spectrum analysis of geophysical data," *R. A. Haubrich and J. W. Tukey*

"Time series and earthquake prediction," *C. Lomnitz*

"Tide prediction," *W. H. Munk*

"Spectra and other statistics in meteorology," *H. A. Panofsky*

Session IV: Analytical Methods

"Programming needs in the water resources field and the role of a problem-oriented language," *G. Bugliarello*

"Satellite navigation in oceanography," *M. Talwani*

"The representation of small-scale turbulence in numerical simulation experiments," *D. K. Lilly*

"Optical scattering in the atmosphere," *B. M. Herman*

Session V: Mathematical Models

"Global circulation" (summary), *S. Manabe*

"Models of wind set-up and oscillations of Lake Erie" (summary), *G. W. Platzman*

"Effects of an ensemble of convective elements on the large-scale motions of the atmosphere," *A. Kasahara*

"Stream modeling for pollution control," *D. J. O'Connor and R. V. Thomann*

Special guest speakers

"Speculation on the future of environmental sciences," *G. J. F. MacDonald*

"Responding to environmental challenge," *R. W. White*

System/4 Pi

System/4 Pi is marketed by IBM's Federal Systems Division. Readers who desire additional information about this family of computers should see their Federal Systems Division representative or write to the Director of Marketing, Federal Systems Division, IBM Corporation, 18100 Frederick Pike, Gaithersburg, Maryland 20760.

Data for Euratom

Euratom, the European Atomic Energy Community, was founded in March 1957 by West Germany, Italy, Belgium, France, Luxembourg, and The Netherlands. Today some 4,000 scientists and technicians, like those in the nuclear research center at Ispra, Italy (photograph at left), are investigating the peaceful uses of atomic energy.

To serve these researchers, Euratom has established a Center of Information and Documentation (CID) in Brussels. The center's information retrieval services are based on the use, for only an hour a day, of an IBM System/360 Model 40 computer at the European Community Computing Center.

On magnetic tape, CID personnel have coded references to some 500,000 publications on such topics as reactor technology, the chemistry and metallurgy of nuclear materials, and nuclear physics, biology, and medicine. The publications include scientific papers, governmental proceedings, theses, and monographs from throughout the world.

Abstracts of the referenced publications are kept in card files at the center. When a request comes in for a bibliography on some topic, the request is coded on punched cards and read into the Model 40. The computer compares the coded request with the publication codes on the tape and, in minutes, prints out both the request and the names of all related abstracts on file. Copies of the abstracts are then sent to the person who requested the information. That person, after reading the abstracts, can order copies of the original publications from either CID or some other organization.

The CID information retrieval system has been in operation for about a year, and by early 1968 the organization expects to be processing requests at a rate of 3,000 to 4,000 annually.