From the user's viewpoint, the fourth (or fifth, depending on your perspective) generation of computers promises more flexibility with little more cost.

KORNEL SPIRO

Many different opinions on the future of computation can be heard in the offices and hallways of large user corporations. There are, in fact, a great number of factors affecting the further development of electronic data processing (EDP). Lacking a universal future-predicting model, as well as the belief in its feasibility, we have to rely on our adductive intuition and extrapolation from computing history and the state-of-the art of computer technology.

A few of the many factors indicating future changes in general-purpose business data processing systems—represented, for axample, by the S/370 product line—are:

* Progress in computer hardware and data communication technologies.

* Advancement in sofware engineering.

* Need for development of better man-machine interfaces (languages) which would reduce the development and operational cost of new EDP applications.

* Requirement of data privacy, security, and integrity.

* Growth of data communication and data sharing needs of societies as well as individuals.

* Development of economic, social, and legislative action, which could ultimately result in either monopolization or decentralization of the data processing industry.

On the other hand, there are also factors that will moderate progress. Forces of tradition, as well as the inertia of inherited data, programs, and EDP personnel, will participate in shaping the forthcoming computer systems.

The time frame

What is the "future generation"? The S/

360 is commonly thought of as the third computer generation. At the time of its introduction, several novelties came into widespread use, notably integrated computer logic circuits, special multiprocessors called data channels, comprehensive multiprogramming operating systems, the data management concept, and high-level programming. Later on, in the early 1970s, some significant new hardware technologies together with some cosmetic architectural changes and additions appeared in computers—for ezample, semiconductor memories, medium-scale logic integration, and virtual storage (a concept or a program that allows for increased storage capacity by utilizing less costly auxiliary memory units). Some people call these upgraded systems the "three-and-a-half generation," some call it the fourth generation, some consider them to be still just the third generation. Without resolving the dilemma, experience says that the closer one is to the user side of the EDP business the more conservative one is crediting these recent innovations with an advancement in the generation count.

What does the future generation, be it the fourth or the fifh one, mean in the context of this article? We still have three or four years to go with the current geenration. (Remember, we are talking about medium- and large-scale general-purpose systems.) On the other hand, I believe that systems appearing on the market in the 1982—84 period will sufficiently differ from the 1975 systems to deserve a new generation number, so let us call them *the future generation.* There is some uncertainty as to whether systems of 1978—81 will resemble more the current of the future generation. In any case, as far as the truly new system generation is concerned—its exact timing and whether we will get there in one distinct giant step or a number of smaller ones—will be left open, although the latter seems to be more probable.

As tradition requires, the user is again going to get more processing power, more storage, new functions, and more headaches in the process of transition to the next system generation.

Yet, with a little bit of good luck, we might witness a few new positive trends: (1) Finally, at least from the user's viewpoint, systems will be simpler to understand, use, and operate; (2) users' current data and program bases will not become obsolete abruptly. A possible scenario is that the systems of 1978—81, while offering a "compatible mode" with the current program and data base, will strongly encourage use of highlevel structured languages for development of new EDP applications. The systems analysts and users of the post-1981 period will not have to worry too much about tolay's machine and assembler language programs and data.

Semiconductor development

These new trends will be influenced by the new flexibility gains that will occur with the added storage and logic circuit capacity of semiconductor components. However, an important observation about semiconductor development is that while density of circuits on a silicon chip will increase impressively, causing an equally impressive circuit cost decrease, the maximum switching speed of high-performance logic circuits used in high-performance computers of the next 10 years will increase only slightly. Sometime in the future there will be another technological breakthrough pushing those speeds to further frontiers, but it is very unlikely that such technology will be massively used within the next decade.

Although the cost reduction for logic elements at the semiconductor chip level will be impressive—a factor of 100 may be realistic —the cost of main memory bits might drop even more, perhaps by a factor of 300-500. Not all this improvement will necessarily be reflected in prices and performance of processors and memory systems, because costs of new-system development, packaging, manufacturing, marketing, and so forth might not fall that sharply. Nevertheless, the memory bit price to the user may drop by a factor of 50 to 100 before the next decade is over.

Low-end systems

What will be the cost consequences across the system size spectrum? Clearly, microcomputers will be all over the place, in our automobiles, household appliances, TV sets, toys, pocket calculators of unheard-of capabilities, and so forth .There will be a widespread proliferation of autonomous, self-sufficient, appli-

cation-oriented mini and small computing systems. These are systems in the price range of, say, $1,000-$100,000 with no or only occasional need to communicate with other systems or external data bases. Procesing and storage capacities of these systems will be comfortably comparable with present small and medium-sized systems. They will be equipped with easy-to-use application software, will requiere no operators besides the actual user himself, and will requiere minimum maintenance. (This has already partially come about with the addition of IBM's S/32.)

The assumption about the self-sufficiency of small systems is an important one because, if they required extensive communication with external data bases or had nonnegligible demand for external computation resources, they would have to be considered as a node in a network of systems. Unless the individual systems in a network belong to several mutually independent organizations, we face a typical "operations research" optimization problem: how to distribute computing and storage resources in order to minimize the overall cost of a network and its operation. It is not clear that an optimal configuration of an autonomous system remains still optimal if such system becomes a node in a network.

Centralization vs. decentralization

If centralization of computing and storage resources caused only a negligible increment in data communication cost, the economies of scale would favor centralization, leaving only the necessary data input and output facilities (and maybe some of their control) geographically distributed. There are, however, several important factors working against centralization—most importantly: (1) the significant rise in communication costs that centralization would require; (2) an increase in local response time, which may be intolerable in some applications, for example, process control; (3) insufficient reliability of the communication line to the central facility; and (4) the potential overcomplication of resource management and the resulting resource waste.

Fortunately, we do not have to predict whether centralized or decentralized computation is the trend of the future—there will be applications whose economics will justify both. Future generations systems architecture will be flexible enough to allow not only either mode of operation, but also easy, nondisruptive transition from one mode to another.

From the foregoing considerations and from a rather simple hypothesis that there always will be applications of quite different performance requirements for either stand-alone systems or system networks, it can be asserted that the spectrum of performance and price ranges of future generation systems will probably be broader than we see today. Microcomputers will extend the performance-cost spectrum at one end, while resource centralization may extend the spectrum on the other end. Furthermore, in the past, every time the hardware performance/price has gone up, the user has consumed the increased performance rather than reduced his hardware expenditures. There is no convincing reason, other than economic recessions, why this should not be so in the future.

Another technology influencing the systems architecture, as well as the population distribution of systems of different sizes, is that of data communication. Data communication using public utility telephone lines is costly and slow, and certainly not satisfactory for future systems Maybe, in due time, private, specialized computer communication services will be available to the data processing community at a reasonable cost. This would significantly influence the way data processing resource will be dstributed.

High-end systems

A typical high-performance computer of the current generation is a uniprocessor in the sense that at any given time it processes only a so-called single instruction stream, that is, instructions for a single task or process. This uniprocessor typically would contain and control several more-or-less autonomus, specialized "subprocessors," such as arithmetic units, data channels, and peripheral controllers. The computer design art has matured enough so that most of the potentials for parallel (overlapped) computations within a single instruction stream

have been sufficiently exploited in present-day computers, either in the from of the mentioned subprocessors or, on a finer level, by so-called pipelining design techniques. An additional increase of logical elements in such a computer will not pay off with sufficient performance increase any more. Thus, availability of cheap future logic elements is not going to speed up the computer significantly. Moderate speed increase of future high-performance circuits, together with some functional and structural improvements, will cause a future high-performance uniprocessor to be faster by a factor of two to four, without going a sphere of quickly diminishing return resulting from overdesign. Whereas this speed may be satisfactory as a single-task performance in most business and many scientific or technical applications of the next decade, it may not be sufficient to provide the aggregate computing power required at a computing center. The solution is an old-fashioned one: Add additional uniprocessors.

However, there is one big difference between today and tomorrow. Today, we would have to add a full new system: computer and memory, including their housing frames and consoles, power supplies, set of peripherals, and so forth. In the future, such multiple uniprocessors would totally share a physical frame, power supply, cooling system, and peripherals, and they may or may not share memory. From the user's viewpoint, such a genuine multiprocessor system may be almost indistinguishable from a uniprocessor system, except for exhibiting far greater multiple-task performance. There are some major advantages of this to the user:

1. He will be able to increase his unit's processing power at the same time he increases his unit's memory size, since a *single* computer will be available in optional increments of computing power.

2. Availability of the computer will significantly improve. Except when there is a power supply failure or a catastrophic event, most uniprocessors will be available around the clock.

3. In many cases the user will be able to nsolidate today's multiple systems into a sin- e multiprocessor system, thus simplifying nputer-room operations.

We might have some problems in defining what should be called a single system and what should be called a multiple system, since the uniprocessors (central processing unit) will not be the distinguishing factor any more. Perhaps strong interconnectivity of system components will serve as a single-system criterion. There will be a practical limit to the number of processors in a single, strongly interconnected system. If too many processors shared a memory, the competition of processors for memory cycles and the memory bus would become a performance-limiting factor. If each processor had its own memory, the competition of individual memories for the input/output bus and peripheral devices would become a limiting factor.

We have arrived at the genuine multiprocessor structure from high-performance computing considerations. In the medium computer size category, we might have a choice: either a small number of high-performance uniprocessors or a large number of medium-performance uniprocessors. Unless a high single-task performance, i.e., a fast response time, is required, the latter will be a better choice, since its design and packaging are less costly. A similar choice would be available in small systems.

Storage increase

The enormous increase of main memory available on a single future multiprocessor will probably cause a major change in form of virtual storage and a substantial reduction of the processing overhead required to manage it. This not only would simplify operating systems, but also would free the processor's time for useful work. It does not mean that the user will again program in real addresses. It only means that real memory might again become a resource explicity assigned for an extended duration to individual tasks.

Two other significant changes in storage hardware seem to be in the offing. First, technologies are emerging which will probably replace some form of disk storage. Examples of such technology are charged-couple devices, magnetic bubbles, and electron-beam addressed memories. Their future performance and cost may justify replacement of at least the high-performance disks.

Second, there will be further progress in the archive storage area, not only in cost/bit decrease, but also in releasing computer-room operations from the duty of mounting storage volumes. This is significant, since it may allow us to remove all personnel from the computer room, except for computer repair purposes.

Slower and cheaper high-density disk technologies will survive the coming decade, but their elimination by a combination of, for example, magnetic bubbles and future mass storage (today's example being IBM's 3850 cartridge store) *might be in sight.*

Software engineering

The progress rate of software will be much slower and more painful than that of hardware because the functions performed by software are and will be more complex than those performed by hardware. The methols of so-called software engineering will become an accepted practice in software development. Today we know these methods under names like structured programming, top-down design, chief programmer team, and so forth. The result will be shorter development times, more understandable programs consisting of smaller autonomous program modules, more error-free programs, and finally, programs much easier to adapt or convert for computers of lifferent architectures or generations. The operating systems will be, in many cases, simpler and more resistant to errors and system protection penetrations. The major contributing factors to this trend will be (1) structured programming; (2) availability of cheap logic and memory hardware, making simplification of some operating system (OS) control functions feasible—for example, storage and task switching management; and (3) implementation of such simplified OS fuctions in hardware.

Examples of functions which might, at least partially, be performed by hardware are: resource switching from task to task or from user to user; saving and restoring of computation status when a task or a process is suspended; safeguarding the protection of all resources of a user from other users; controlling of fast levels of auxiliary storage (for example, future equivalents of today's paging storage); and input/output supervision.

Man-machine interfaces

One contributing factor to software problems is the imperfect state of man-machine interfaces. Obviously, improvement in many areas—such as machine instruction and command languages, computer operator interfaces, high-level languages, and end-user application interfaces—are desirable.

However, progress in man-machine interfaces may be even slower and more difficult than in the software engineering area. After almost 20 years of high-level language history, we still do not have a language in general use that satisfies our criteria (or intuition) of a good man-machine interface. I think the search for better not-so- universal languages will continue, with some positive results. I do not know how much of FORTRAN, COBOL, PL/I will still be used for newprogram development around 1984, but it would be surprising if they lasted in their present form much longer.

Reasonably good compiler writing techniques are already known, and compiler development starts to lose some of its aura of complexity and enormous cost if you stay away from PL/I compilers. This will facilitate development of several new good languages, and many more bad languages, within the next decade. Perhaps rather than having very few future general-purpose languages in wide use, we may witness an explosion in the number of good special-purpose languages, each obeying rules for well-structured machine-independent languages. Portability of programs from system to system or from system generation to system generation might require recompiling; however, implementation of compilers will then be a task almost automatically performed by computers. Such advances will be a significant improvement over the 1970s in the quality of man-machine interfaces.

Data security

Data privacy, security, and integrity, all limitations of EDP systems, constitute a problem that we somehow always manage to live with—complaining about it but refusing to spend any significant money to solve it. Therefore, we might expect that, despite all the cur-

rent publicity, and except for some special projects, neither manufacturers nor users are going to spend big money trying to solve these problems. Fortunately, as the hardware becomes more sophisticated, data security and integrity will naturally be improved anyway; however, privacy is a broader problem which has more to do with the discretion of the people who have legitimate access to the data than with properties of EDP systems. It is a social problem.

Data access and data communication

The amount of information generated and stored by mankind is accelerating every day. Although individually we can accept only a limited amount of information within a constant time interval, our need for information grows in two ways: We select information from expanding information bases and we increasingly require other people or computers to preprocess the data for us. It has been recognized within the past few yaers that data base management, information retrieval, and data communication are among the most important future computer applications.

Within the next decade the major burden in developing data bases will be on the software rather than the hardware side of the EDP business, and certainly progress will be made. However, future computer hardware development in the areas of volume, price, and performance of mass storage; cost and bandwidth of data communication links; and data integrity and security properties of hardware will facilitate data-base evolution. All of these changes add up to an exciting and open-ended era—one in which the user stands to make large gains in flexibility with moderate increases in cost.