DOCUMENT RESUME

ED 051 552 88 EA 003 534

AUTHOR Summers, J. K.; Sullivan, J. E.
TITLE The State of the Art in Information Handling. Operation PEP/Executive Information Systems.
INSTITUTION Mitre Corp., Bedford, Mass.; Operation PEP, Burlingame, Calif.; San Mateo County Superintendent of Schools, Redwood City, Calif.
SPONS AGENCY Bureau of Elementary and Secondary Education (DHEW/OE), Washington, D.C.
REPORT NO M68-10
PUB DATE Jun 70
NOTE 99p.
AVAILABLE FROM Mrs. Elaine Barnes, Director of Education, San Mateo County Office of Education, 590 Hamilton St., Redwood City, California 94063 ($.50)

EDRS PRICE EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS Computer Oriented Programs, *Computers, Computer Science, *Computer Storage Devices, Data Bases, *Data Processing, Data Processing Occupations, Information Dissemination, Information Networks, *Information Processing, Information Retrieval, Information Services, Information Storage, Information Systems, *Management Systems, Time Sharing
IDENTIFIERS ESEA Title III, Operation PEP

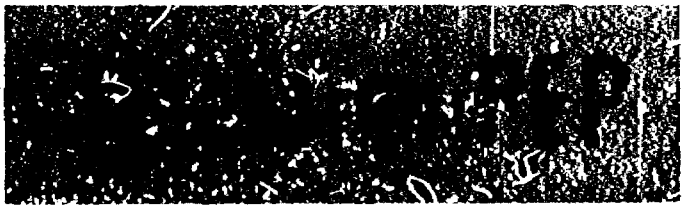
ABSTRACT

This document explains recent developments in computer science and information systems of interest to the educational manager. A brief history of computers is included, together with an examination of modern computers' capabilities. Various features of card, tape, and disk information storage systems are presented. The importance of time-sharing computer systems for educational data management is outlined. Funds for this research were provided by an ESEA Title III grant. (RA)

EXECUTIVE INFORMATION SYSTEMS

The State of the Art in Information Handling

Prepared by
The MITRE CORPORATION for



THE STATE OF THE ART IN INFORMATION HANDLING

J.K. Summers and J.E. Sullivan

June 1970

This paper has been especially prepared for distribution to the California Educational Administrators participating in the "Executive Information Systems" Unit of Instruction as part of the instructional program of OPERATION PEP (Prepare Educational Planners).

THE MITRE CORPORATION

PREFACE

Under a (1968) contract with the San Mateo County (California) Superintendent of Schools, the Information Systems Division of The MITRE Corporation, in conjunction with the staff of Operation PEP* (Prepare Educational Planners), prepared a three-day Unit of Instruction on Executive Information Systems. The purpose of the course, presented in June 1968, was to support Operation PEP in its efforts to introduce some basic concepts of information systems technology to California Educational Administrators.

The presentations included in the Unit of Instruction were augmented by several reports. Three supplemented the discussions by providing general background material. The remainder, prepared as companion handouts for the individual presentations, contained copies of the visual aids (diagrams) used to emphasize significant concepts.

The present (1970) contract between the San Mateo County Superintendent of Schools and The MITRE Corporation calls for the documentation of the concepts illustrated in those diagrams and for the reissue of the supplementary reports. The objective is to provide, in one package, a complete set of references which can be used by Operation PEP in its over-all instructional program. The contents of the package, which consists of eight reports, are identified in the following list.

---

* Operation PEP is funded by a U.S. Office of Education Grant Award under Title III of the Elementary and Secondary Education Act of 1965 (P.L. 89-10).

SUPPLEMENTARY REPORTS

Digital Computer Principles

J. D. Porter

Input-Output Trends

J. Mitchell

Digital Simulation and Modelling

G. B. Hawthorne, Jr.

UNIT OF INSTRUCTION REPORTS

Information System Overview

J. H. Burrows

The State-of-the-Art in Information Handling

J. K. Summers and
J. E. Sullivan

A Framework for the Evolutionary Development of an Executive Information System (in two parts)

J. A. Evans

Persistent Problems in System Development

J. H. Burrows

An Information System for a District School Administrator

S. G. Lewis

Collectively, these reports provide a basic overview of information system technology; individually, they focus on some of the specific aspects associated with the design, development, implementation, and use of information systems.

THE STATE OF THE ART

IN

INFORMATION HANDLING

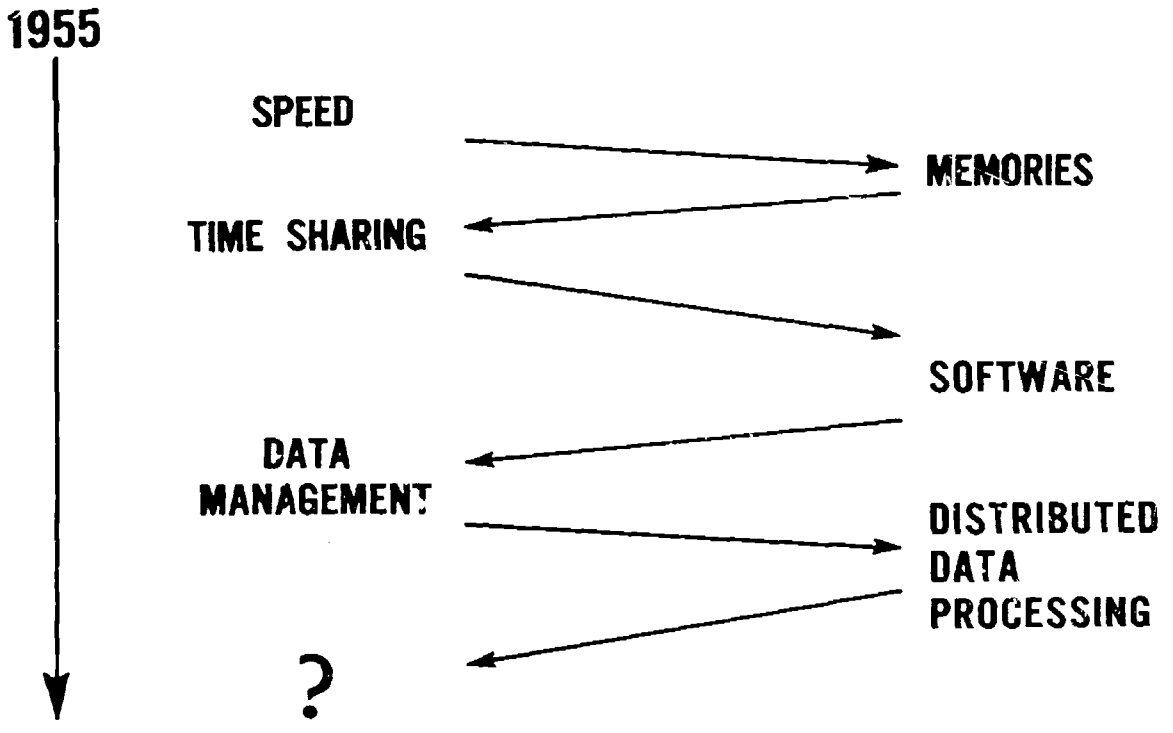
The purpose of this presentation is to discuss "information handling" – specifically, the handling of information by computers. Now, in a broad sense, that's all computers ever do: handle information. But in a more technical sense "information handling" problems are characterized by large amounts of information, so that simply keeping track of it becomes a critical part of processing it. By analogy, a mathematician solving a set of equations in three variables can be fairly informal about it; he'll probably call the variables x, y, and z and dash off the solution on a scrap of paper. But, if the number of variables go up to, say, 100 or so, the problems of scale show up. And it's not just a question of a larger paper supply. Some formal arrangements and procedures must be introduced so that the right number is available at the right time to be entered into the right computation. These arrangements and procedures are called the "handling" of the data, as opposed to the actual calculation. And, if we jump again from 100 variables to problems involving several million pieces of information, then it becomes obvious why "information handling" is an important part of computer technology.

But if we are to take "information handling" in its technical sense, we also should take the words "state of the art" in a definitely non-technical sense. That is, the history of information handling will be covered only in broad outline form; specific projects and other details will be omitted. We will start by assuming that you know nothing about the technical side of computers, even though some of you may know quite a bit about them *

Finally, every state-of-the-art exposition should date itself. The applicable date for this one is circa 1967-8.

* If you have read "Digital Computer Principles" by J.D. Porter (see supplementary reports listed in the Preface), you are laready familiar with most of the concepts that will be discussed.

EXECUTIVE INFORMATION SYSTEMS	A PAGE OF HISTORY



The facing chart presents the history of information handling in terms of the emphasis placed on various concepts, or parts of the problem, as time progressed. If 1955 strikes you as a little too recent to call "history," bear in mind that practical computers were first developed in the late 40s and early 50s, and that technological advances are still quite rapid.

The terms used in the chart are computing jargon and, in keeping with the assumption that you are not familiar with computing jargon, we're going to have to define them as we step through this outline. And to define them will require some further preliminaries. But we will see this chart again; suffice it for now to point out that shifts of emphasis generally occur as limiting factors in the previous approach to information handling are isolated and approaches to these new problems are in turn proposed.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

OUTLINE

- COMPUTERS IN GENERAL
- HARDWARE
  - SPEED
  - MEMORIES
- TIME SHARING
- SOFTWARE
- DATA MANAGEMENT
- DISTRIBUTED DATA PROCESSING

The opposite chart shows the logical outline we'll be following. It is almost identical to the historical outline, except for the general topic.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

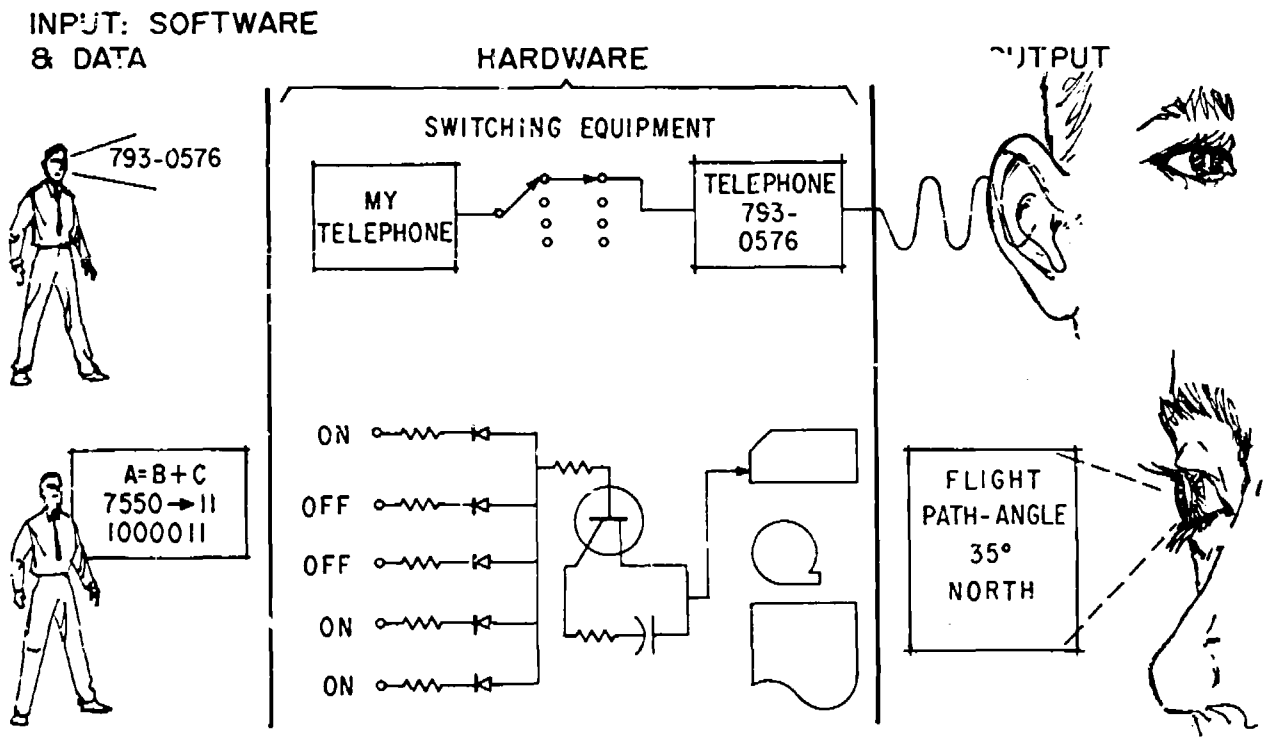
A COMPUTER ADVERTISEMENT

"AN ON-LINE REAL-TIME TIME-SHARING 3rd GENERATION SYSTEM"

Let us turn now to some basic definitions. The opposite chart contains some terms commonly found in technical journals today. We will define these terms presently, but note that the word "computer" is not even present; it is assumed that a reader faced with such an overload of gobbledygook will automatically supply it for himself.

Actually, although one still sees ads like this, the computer industry has come a long way in its public relations. A while back, both the jargon and a mistaken idea of what was good for sales combined to surround computers with an aura of magic. In the public mind, computers were almost human. In fact, there is a body of theory to the effect that an "intelligent" machine — whatever that is — can be built. But even today, the theories and the practicalities are far, far apart.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING



Today, most people know a computer is fundamentally a simple device – that its hardware (that is, its electrical circuits and mechanical parts) can respond only to the most simple commands such as "add." Furthermore, both the commands and the data they operate on are defined by the state of some electrical circuit or mechanical device – just as the position of a cog in a desk calculator determines what number appears in the window. If the circuit or device in question is an integral part of the computer, we call it part of the computer's "memory." If it is more remotely connected – by a cable, say – it is called an "input-output device."

The difference is really quantitative, although of course there are qualitative differences in the technologies employed. Some input devices are designed so that their states are easily manipulated by man; a card reader, for example, reacts to those holes in those cards we are all familiar with. It is through this type of device that the computer is ultimately controlled by humans. Some output devices are

designed so that their states are easily interpreted by man; a printer, for example. It is through such devices that the "answers" are reported back to humans.

A good example of a true computer, one that you are all familiar with, is the telephone network. True, this is a computer with an especially simple set of commands — it cannot add, for example. But it can respond to the command, "connect me to number x" — a command which the programmer — you — enter by picking up the handset and dialing x. In fact, the relationship between computers and the telephone is not as far-fetched as sounds, because the development of early computers was an outgrowth of telephone-switching technology.

But while computers respond only to simple commands, their inherent potential lies in their ability to store and sequence through a set of such commands which, taken together, do something more complex. For example, a sequence of elementary operations may be so arranged that the square root of a number supplied at the start of the process emerges at the end. In the abstract,

such a sequence is called a "program," or "software." This program needs to be composed only once, but it may be run through by the computer whenever and wherever the square root of a number is to be taken. In effect, the hardware is extended by the software so that the "system" now has all the hardware commands plus a square root command. And then this foundation may be built upon, layer after layer, until the "system" is indeed complex and is capable of doing marvelous – but not magic nor even "intelligent" – things.

So the in terms in an advertisement such as the one in the previous chart are reducible to fairly simple concepts by applying some definitions.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

AMERICAN STANDARDS ASSN. DEFINITIONS

ON-LINE -PERTAINING TO EQUIPMENT OR DEVICES UNDER DIRECT CONTROL OF THE CENTRAL PROCESSING UNIT

REAL-TIME - PERTAINING TO THE ACTUAL TIME DURING WHICH A PHYSICAL PROCESS TRANSPIRES

TIME SHARING - PERTAINING TO THE INTERLEAVED USE OF THE TIME OF A DEVICE

THIRD GENERATION - NO DEFINITION

The American Standards Association supplies standards for many different types of materials, and definitions for technical terms. These are their definitions for the terms appearing in the advertisement we saw.

"On-line: pertaining to equipment or devices under direct control of the central processing unit (CPU)." Now the CPU is what one may call the innermost part of the computer. It is the mechanism which actually obtains each command, determines what command it is, gets the needed operands, performs the command, stores the results, gets the next command, and so on and on. And so those devices connected to it are the ones which enter into the computing process in a direct, immediate sense. By contrast, devices such as card keypunch machines, which interact with the CPU only indirectly, are called "off-line."

Properly, the term applies only to individual devices, but when one speaks of his system as on-line, rather than just one of the input-output devices in the system, what he means is that the users of the system may sit at on-line devices, called terminals, and interact with the system. This is in contrast with a typical "batch" system, where the user prepares the complete instructions for the computer off-line – usually punching them into a deck of cards – and leaves them with an operator who runs the computer process in the user's absence.

"Real time: pertaining to the actual time during which a physical process transpires." "Physical process" is here contrasted with the computing process, even though, as we have seen, computing is also a physical process. "Physical process" in this context is typified by a rocket launching. If a computer is somehow linked to the launching of a rocket -- and they always are -- we have an example of a real-time application. Typically, the computer would be assigned the job of tracking and steering the rocket. Data from radars and other sensors are fed into the computer, which then computes the position and velocity of the rocket, determines any needed corrections, and transmits the steering commands to the rocket. Now it is obvious that this computational process must, in some sense, keep up with the physical rocket motion: if the rocket starts to go off course, the computer had better "sense" this and get the corrective steering command back before it's too late.

Turning back to the case at hand, it is clear that the system advertisement misuses the term slightly. The point is that frequently the only "physical process" involved with such systems is a user waiting at a terminal someplace. If he has requested a computation, there is no direct physical limit on the time during which the computation must be completed. Of course, there are psychological limits, and there may be indirect limits, such as a report due.

"Time Sharing: pertaining to the interleaved use of the time of a device." The "device" referred to here is most often the CPU -- although other components of the computer are usually

involved as well. As you might imagine, the CPU is quite expensive, as is the memory. And both operate at very high speeds, as we shall see in a moment. Now an on-line system with only one user would almost certainly spend most of its time waiting for the user to decide what he wanted to do next. This is wasteful of an expensive resource, so it is attractive to permit more than one user to be on-line at once. The users are automatically allotted a share of the available CPU time, taking turns or "interleaving" so rapidly that an individual user does not notice the presence of the others except possibly in a slowing down of more complex processes.

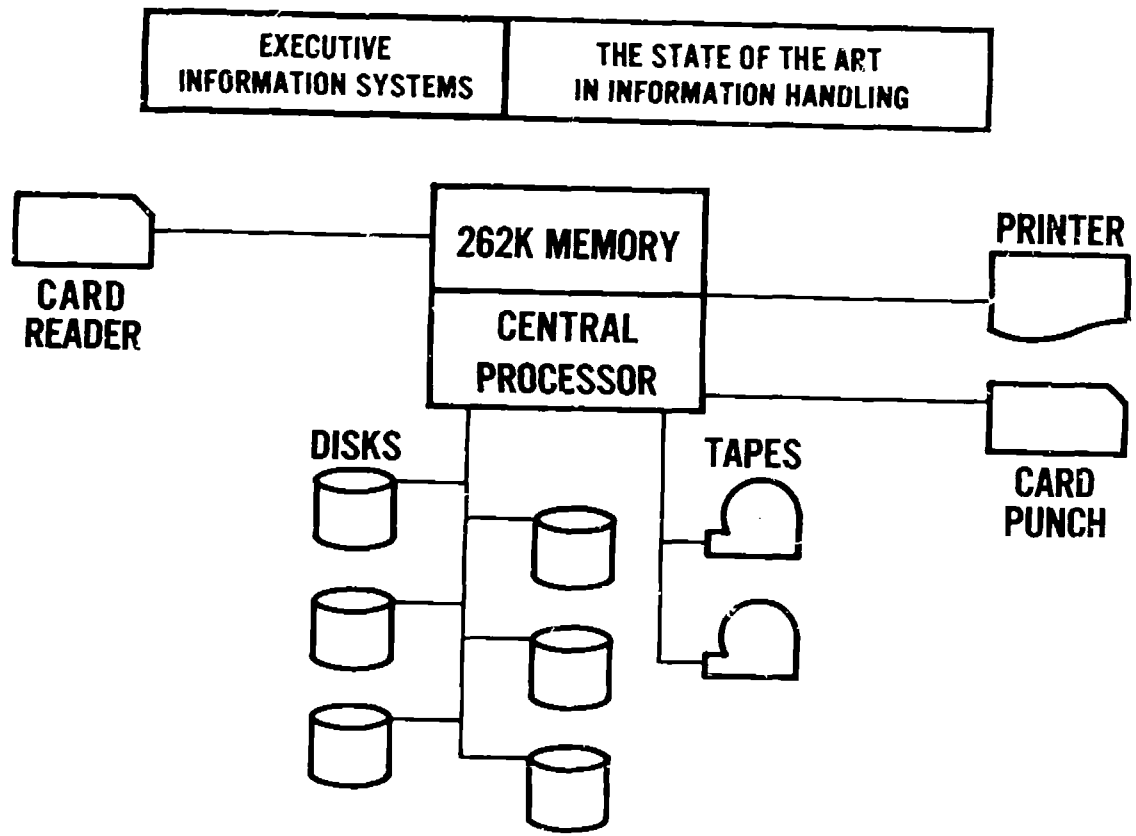
The ASA declines to define "third generation" and, in fact, it is not a precise term. By general agreement, the "generation" of a machine is determined by the technology employed in the CPU and memory hardware. The early machines – the first generation – used vacuum tubes. The tubes themselves were large and required elaborate provisions for cooling, so that computers then might occupy several large rooms. The second generation was ushered in with the development of transistors. It became practical to build far more capability into a computer and, at the same time, reduce the size of the computer itself (not counting the input-output equipment) so that it could fit in a corner of a room. In the thrid generation we have integrated circuits – tiny components, smaller than the head of a pin – that do even more than transistors. And so now we have "desk-top" computers that far outstrip the old "monsters" in computing power, speed, reliability, and operating economy.

EXECUTIVE INFORMATION SYSTEMS
THE STATE OF THE ART IN INFORMATION HANDLING
INPUT
STORAGE
PROCESSING
OUTPUT
TIES TO OTHER SYSTEMS
PROGRAMMER

To recapitulate a bit: the opposite chart depicts the logical structure of a computing system. Two users are shown on the left. One of them is preparing the instructions for the computer — he's called the "programmer" — and the other is supplying the input data for a particular run. Both control the computing process through the input devices, as discussed earlier. Naturally, the representation of the users as two persons is arbitrary, reflecting a division of the work that is commonly encountered. There could be only one user, or many more than two.

That puzzled-looking fellow on the right is examining the results produced on some output device, probably a printer. The output may contain his answers or may only inform him he has made an error that he will have to correct before trying again. By the way, it's only in the better systems that a computer printout is succinct enough to fit on one page as shown in the chart.

The two items in the center are the computer "proper" — the memory and processing unit. We have seen that the memory contains the instructions and the data, while the processing unit is the device that actually carries out the instructions and keeps everything going. "Ties to other systems" are, in effect, input-output devices shared by two or more computers, permitting a sort of "cross-talk."


EXECUTIVE INFORMATION SYSTEMS
THE STATE OF THE ART IN INFORMATION HANDLING
262K MEMORY
CENTRAL PROCESSOR
CARD READER
PRINTER
CARD PUNCH
DISKS
TAPES

A somewhat more concrete system diagram is provided in the opposite chart. Some of the actual input-output devices that typically are hooked up to a computer are shown. The device at the left is a pure input device; those on the right are pure output devices; those in the center may be either. The devices shown are by no means the only input-output devices currently in use, but they constitute a representative sample.

The card reader (input device) already has been mentioned. Upon command from the central processor, the reader passes a card through a group of "electric eyes" so that a hole in the card causes one type of signal to be sent to the computer while the lack of a hole generates another type of signal. The resulting "hole — no hole" pattern is placed in the memory as a series of circuit states, where the state of each circuit is, in some sense, either "on" or "off."

Incidentally, this is what makes "digital" computers digital — the fact that out of a finite number of possible states, the individual circuits are always in one state or another. If the number of states possible were not finite — as, for example, is the case with the slide rule, where the number of positions is infinite — we would have what is called an "analog" computer. However, analog computers are sufficiently different both in principle and application.

that they form a separate topic and we shall not discuss them further. In common use, when someone says "computer", he almost always means "digital computer." Furthermore, all digital computers in practical use today utilize only two-state circuits, so that there is no intermediate position between "on" and "off."

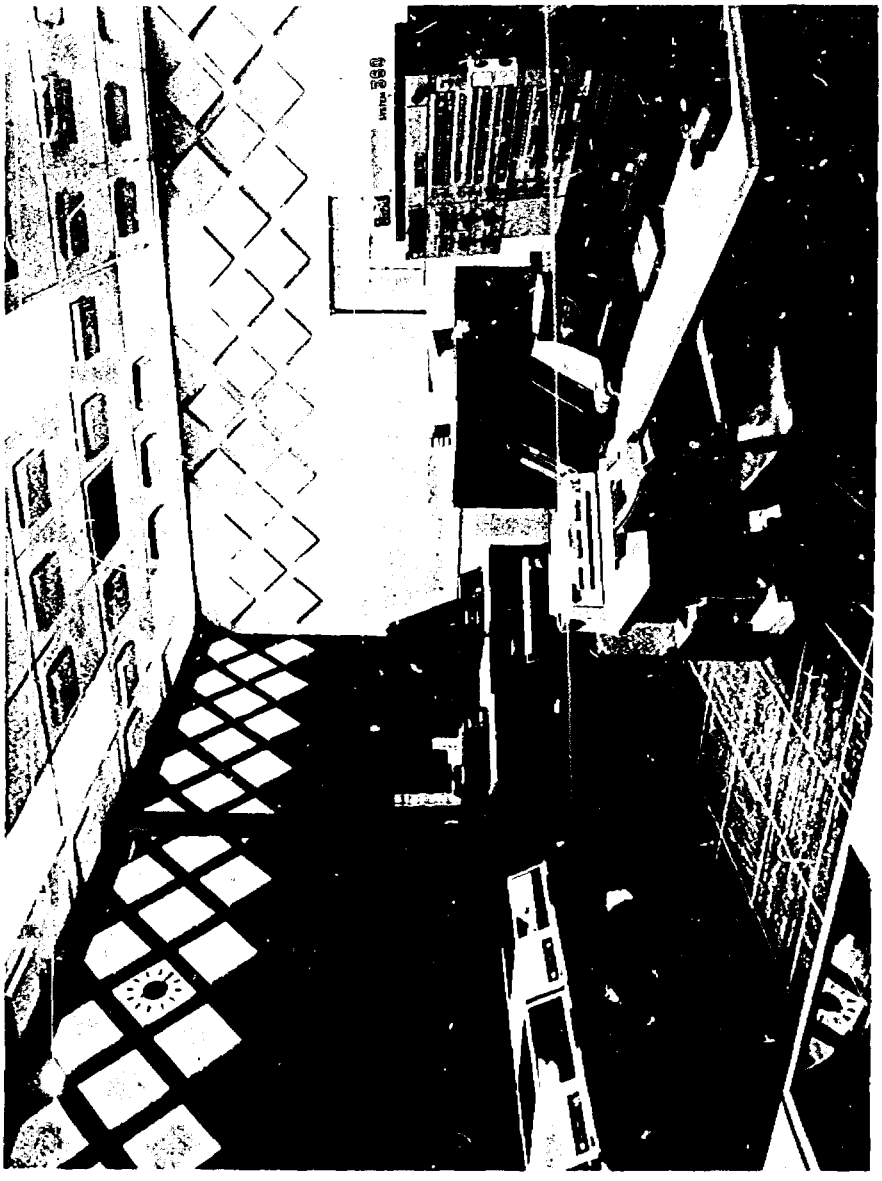
Getting back to the card, now represented in the memory as a sequence of on's and off's, it is important to realize that the "meaning" of the circuit states are quite arbitrary and very much a function of what the program does with this data. In one case, on and off may represent "true" and "false" respectively; in another, "male" and "female", and so on for any dichotomy whatever. In particular, "off" may be taken to represent the binary (base 2) digit "0" and "on" the binary digit "1". By grouping these binary digits or "bits" together, larger numbers can be represented in the binary base. Two bits, for example, permit counting from zero to three: 00, 01, 10, and 11. Three bits permit counting to seven, and so on. In this way a relatively few bits permit the storage of very large numbers; with 32 bits, for example, one can express numbers in excess of 4 billion. Moreover, a number so stored need not represent a quantity of something; it could simply be a code selecting one of a number of alternatives — a letter of the alphabet, for example, or a particular student in a given school. In fact, all information — the program

instructions as well as the data – is stored in this binary form. In this sense all digital computers are binary computers. However, the slightly misleading term "decimal computer" is occasionally applied to computers whose hardware supports a representation of numbers as a string of decimal digits – as in ordinary notation. But the digits themselves are of course represented in the machine by groups of four or more bits, so that even these machines are binary at heart.

The function of a card punch is just the reverse of a card reader; on command from the central processor, bits in memory become holes in cards.

A printer, like a punch, is driven by the processing unit: on command, a group of bits, called a "byte," is sent to the printer. The byte identifies the particular character to be printed.

Disks and tapes are examples of magnetic recording media. The recording principle applicable to a sound tape recorder also applies to them: the magnetization of particles on the surface carry the information. Computer tapes resemble sound tapes, and disks look like phonograph records stacked one on top of the other, with enough distance between to admit a recording/pickup arm.



The above figure shows an actual computer system installation. The one shown happens to be an IBM 360/40. The operator normally sits in the foreground. Those lights on the console in front of him directly reflect the status of various important bits in the memory: if the light is on, the corresponding bit is on (i.e., "1"). A particular bit is "important" if it is part of a group of bits — variously called a "word" or a "register" — assigned a special meaning by the CPU. For example, one register always tells the CPU where to get the next instruction: that is, it contains the "address" of that instruction. It is called the instruction counter, and is represented on the console.

The typewriter is an input-output device for the exclusive use of the operator. When the program reaches a point where a message must be displayed to the operator, it uses this typewriter as a printer. In response, the operator may type in a reply that is read by the CPU much as a card is read. The telephone, by the way. is in no way connected to the computer.

The memory and CPU of the computer are in a cabinet behind the console. To the rear, near the door, is a printer. The paper is stored inside and fed up through the printing mechanism; it appears briefly in the window on top on its way to a stacker in back. To the left is a card reader. The cards are stacked in the tall slide, and are fed from the bottom through the reading mechanism into one of the stackers in front. The two units to the left of the card reader, and the unit partially visible in the foreground, are disk drives. The particular type of disk used here can be removed from the drives and stored or perhaps carried to another machine with similar drives.

The above view of the same installation shows the tape drives in the background. One of the two reels on each drive provides for permanent storage of the tape. When mounted, the tape winds from this reel over the read-write head — between and below the reels — and onto the take-up reel. The long vacuum columns on either side of the read-write head receive the tape and cushion it against mechanical strain as the speed and direction of motion are abruptly changed.

COST BREAKDOWN

	$ RENTAL/MONTH
PROCESSOR	5,000
MAIN MEMORY	7,600
CARD AND PRINTER EQUIPMENT	3,000
DISKS	4,000
TAPES	2,500
TOTAL	22,100

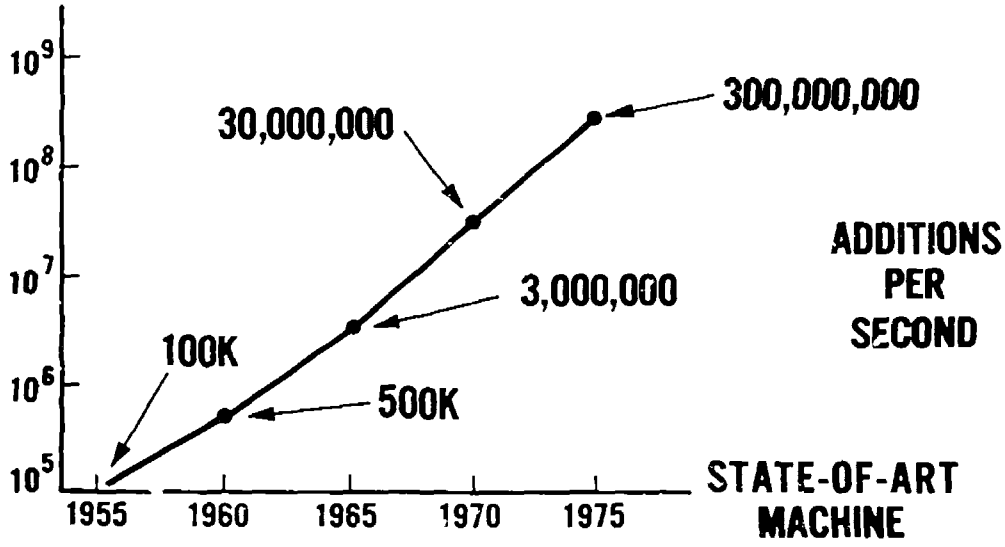
An important consideration in acquiring computer equipment is economic feasibility. And this, in turn, depends on the cost of buying, renting or leasing the equipment.

To get some idea of what computing equipment cost, we might look at the monthly rental figures (see above chart) for equipment such as we have seen. Currently, a computer such as this would be considered "medium" in size and capability.

The cost of buying equipment outright is generally estimated to be 40 times the monthly rental.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

COMPUTING SPEED



The opposite chart emphasizes the most fundamental payoff to be realized from computer hardware expenditures, and the fundamental reason why computers can be applied to large-volume information handling problems: their fantastic speed. The latest computers can reform several million basic operations, such as adding two numbers, in one second. Not only is the speed remarkable in itself, but also in the way it has increased since the mid-1950s, when 100,000 operations per second was the state of the art. This happened because, when problems became too big or too involved to be handled with the current machine, the raw speed of the processor became the obvious target for improvement. In the beginning, speed did stand out as one of the limiting factors, and much attention was paid to it. So machines have become faster and faster, and there is every reason to believe they will continue to do so, as is projected by the above figure of a third of a billion operations per second by 1975.

ADD 2 NUMBERS / SEC

24 HOURS A DAY

TEN (10) YEARS OF YOUR EFFORT =

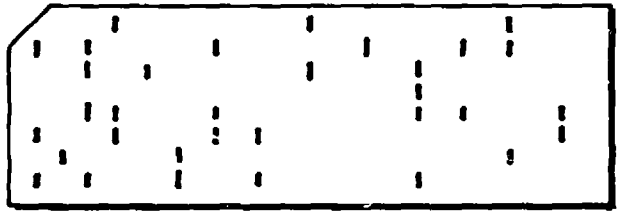
ONE (1) SECOND OF COMPUTER

In order to put this speed in some sort of perspective, we might compare it to human labor. If a person could add a pair of numbers in one second, and in fact did only this for 10 years without eating or sleeping, he would have accomplished what a computer can do in only one second. Of course, these figures can be misleading. For one thing, even a computer cannot spend all its time adding. Typically, it uses up some time just getting the data into position for computation, and some more time determining what to do next — move data, add, or finish the operation.

In a practical illustration, this could alter the equivalent computer time from one second to five seconds or even five minutes. But even more to the point, it must emphasized that the illustration involves only a simple, mechanical kind of operation. It could involve more complex operations only if they are equally mechanical in the sense that they can be reduced to a set of simple operations.

On the other hand, some things are practically impossible for a computer to do but can easily be done by men. Making judgments and spontaneous adaptations are among these. For example, take the problem "starting with zero as the current result, add 2 to the current result for 250 trillion times; what is the final result?". A human would instanly note that this is really a multiplication problem and announce "500 trillion," even if he's not sure what a trillion means. A computer, however, would obediently grind out 250 trillion additions, thereby taking several days to deliver the answer.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING



- PERMANENT STORAGE

- READ INTO A COMPUTER AT 1000 CARDS/MINUTE

We have seen that the processor speed is the basis of the computer's power. But if we concentrate all our efforts on improving that speed, we soon run into another aspect of computer hardware: the memory. We've seen what the memory is; it holds data that is either stored by the processor or is to be used by the processor, or both. We have also seen that many types of memory, usually called "input-output devices," can be attached to a typical computer. According to the technology employed, memories differ widely in many important respects, such as the ability to erase and re-record, or the ability to read the memory nondestructively. Beyond these, the fundamental limits to the use of a given memory are its size in bits, the time it takes to get to a desired item in the memory (the access time), and the transfer time to get the data from the memory to the processor or vice versa (the read/write times).

Cards, for example, offer a non-erasable, human-preparable form of storage. In quantity, they are bulky to handle and notoriously easy to mix up. But their virtues are such that the great majority of all data used by computers is traceable to data punched on cards.

Because of the way card readers are built, cards are always read by the processor in sequential order, never backing up nor jumping over a group of cards. Consequently, the term "access time" is not generally used with cards. And, because the reading of a card is largely a mechanical process, transfer rates are very slow: 1000 cards per minute, or approximately 16,000 bits per second. When processor speed is considered, it becomes clear that the card reader cannot possibly send data fast enough to keep the processor busy.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

STUDENT RECORDS

IDENTIFICATION	10 CHAR.
NAME	30 CHAR.
ADDRESS	50 CHAR.
BIRTHPLACE	5 CHAR.
●	
●	
●	
●	
COURSE AND GRADE	11 CHAR.
●	
TOTAL	1,000 CHAR.

MASTER FILE
OF
10,000 STUDENTS
10,000,000 CHARACTERS

Before considering other types of memory devices, let's take a look at some data that might be stored in a memory. A school might wish to store, for each one of 10,000 students, a "record" something such as that shown above. The details are not important to our example, but we assume a total of 1000 characters, on the a average, for each student's record. A character representation requires 6 bits or more; the current standard seems to be 8 bits.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

STUDENT RECORDS ON CARDS

- 10 CARDS PER STUDENT
- TOTAL OF 100,000 CARDS
- 50 BOXES OF CARDS
- 100 MINUTES TO READ INTO A COMPUTER

If cards are chosen as the only permanent form of storage for the information, we would find that we have a lot of cards: 50 boxes of 2000 cards each. More than 90 minutes per processing run would be consumed to read them into the computer. And if the purpose of a particular run simply was to calculate the grade point average for, say, Zeke Zilch, the last name in the file, we again would find that man can do the job faster: he can just yank the right card from the file while the computer must search all the cards.

The above example illustrates why data originally entered on cards are almost always transferred to some other medium as a first step, especially if access in non-sequential order is necessary.

STUDENT RECORDS ON TAPE

- 1/4 REEL OF TAPE
- 1-1/2 MINUTES TO READ INTO A COMPUTER
- LONG SEARCH TIME FOR 1 RECORD

An alternate storage medium commonly used is magnetic tape. The same student records shown on page 36 will fit on approximately one-fourth of a tape reel and, in this form, can be read into the computer in about 90 seconds (see opposite chart). These figures are very approximate and, to a great extent, will vary with the particular equipment being used and will depend on other technical matters. Still, the savings are obvious.

On the other hand, tapes, like cards, must be processed sequentially. Although a search for a randomly selected item is possible, the associated costs are still very expensive: all intervening data must be included in the search. In our example, the average access time for a record selected at random, assuming that the search starts at the beginning of the tape, is 45 seconds.

An additional advantage characteristic of tapes is that they can be stepped backward.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

STUDENT RECORDS ON DISK

- PART OF A DISK PACK

- CAN FIND A SINGLE RECORD IN A FRACTION OF A SECOND

Disks represent still another form of storage. They vary greatly in capacity and speed; so there is little point in being too specific. Our file, however, probably would fit on part — perhaps one-half — of a disk pack, and a record selected at random could be located in a fraction — on the order of 1/100th — of a second. Fast access is possible because the read arm may be placed directly over the desired data.

COMPARISON

	CAPACITY IN MILLIONS OF CHAR.	COST CENTS/CHAR.
TAPE	30	0.08
SMALL DISK	5	0.76
LARGE DISK	225	0.16
CENTRAL MEMORY	1	32.0

The opposite chart shows cost and capacity comparisons for some of the components already discussed, plus one more: the central or "core" memory. Core memory is pure electronic storage. Humans cannot conveniently gain access to it, but the processor can both randomly access it and read or write it in a matter of millionths of a second. In fact, it is the only type of memory that can keep up with the processor. Or, to turn it around, there is no point in building processors any faster than the fastest memory, and this is it. This is why improving central memory speeds have been the subject of studies, to the point where an order-of-magnitude improvement has been made.

This comparison points out the next roadblock: cost. Although the capacity of a typical unit has some bearing on the matter, it is not the main problem because we can, within limits, simply hang more units on the computer. But cost is a definite factor: we still pay dearly for high-speed storage, even though the price is down considerably from what it was. And, if we consider information handling applications especially, it is clear that we are still very dependent upon the "input-output" types of memory anyway. This is because such applications, as opposed to "scientific" applications, involve much data with relatively little processing per data item. Because there is a lot of data to read and, further, because the central memory won't hold it all (and so it must be shuttled back and forth between this core memory and slower devices such as disks), information processing programs frequently cannot make use of the speed built into the processor and core memory. They are "input-output bound."

Increased speed for the input-output devices has also been achieved (and, undoubtedly, the limit is yet to be reached), but each type of device is subject to theoretical limits which obviate any drastic improvements.

Emphasis on hardware was necessary and helpful, but the law of diminishing returns soon set in. So, the focus of technological development shifted to the other component of computer systems: the software.

TIME SHARING

DEFINITION

- SIMULTANEOUS USE OF A COMPUTER BY A NUMBER OF USERS

One of the software problems easily identified was the loss of processor time while a program waited for data to be transferred. If some other program whose data was already in core could use the processor during this interval, some of the "lost" time could be recovered. Beside the gain in efficiency, it would be possible for the computer to be accessed by several users simultaneously, thereby yielding further benefits (see page 49). The term "simultaneous" is not, of course, technically exact —— "interleaved" is a better word —— but the effect is one of simultaneity.

Actual implementation of this simple idea, known as time sharing, required a further hardware development: the input-output channel. This channel frees the central processor from continuous involvment in data transfer to or from core, so that the notion of letting another program use the processor becomes technically feasible. The channel is, in effect, a special-purpose processor whose actions are independent of the central processor except at that point where the central processor tells the channel what to do and at that point where the channel notifies the processor that its operations are completed.

Using the input-output channel, it is quite possible for a single user program —— even one dealing with large quantities of data —— to keep the CPU busy. But this requires very careful, and in some ways unnatural, programming. The program must be constructed to anticipate and initiate data transfers well in advance of the point where the transfer must be effected in order for the programs to continue. As a rule it is easier to overlap one program's input-output with another, presumably unrelated, program's processing.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

WHY TIME SHARING

- ONLINE USE OF THE COMPUTER

  ONLINE PROBLEM SOLVING

- COMPUTING POWER AND ECONOMICS

Simultaneous use of the computer by several users is in itself attractive for several reasons. For one thing, no one program bears the responsibility for keeping all that expensive equipment busy. The system becomes idle only when no program is processing data, a circumstance that becomes increasingly improbable as the number of users for that system grows. So it becomes attractive to interact directly with the user — accepting his inputs and displaying his outputs at a terminal. The penalty of idle computer time incurred while he makes up his mind, perhaps on the basis of the last output, before entering his next input is removed.

Thus systems permitting the user to be "on-line" become feasible, where before only the smallest machines could economically be used in this way. And when the user is on-line, he and the computer can work together, each contributing widely different skills, toward the solution of a problem. This is also true of batch systems, but the difference is that the elapsed time from problem formulation to solution can be weeks instead of hours with a batch system and, at times, the man-machine interaction may appear to be more one of contention than cooperation.

We have already mentioned improvements in computing power and the savings that accrue when the computer is used efficiently. And there are others. Programmers tend to get their programs going, and program users tend to get their problems set up, with fewer false starts. This is because continuity of thought can be maintained from try to try; time is not wasted and mistakes are not made in picking up the thread. Also, the wasteful processing sprees incurred as the computer blindly follows an erroneous program — a regular occurrence in batch systems — can usually be noticed by the human user and nipped in the bud.

TYPES OF TIME SHARING

- TIME SLICE
- INPUT OUTPUT BREAKPOINT
- FOREGROUND BACKGROUND

We have called time-sharing a software development because it is actually a program – a sort of "traffic cop" or control program – that decides which other programs are to use the processor at what times, and takes care of other details such as apportioning the use of the core memory. Of course, this control program must be able to preempt the CPU at times and, in general, it is more privileged than the other programs in ways that various hardware innovations permit. However, being a program, it can allocate time according to any one of a number of algorithms. For example, it can simply take some over-all cycle time – say, one second – divide that by the number of users, and arbitrarily assign a "slice" of the resultant times to each user, in turn, who wants it. It's fair, and assures every user a chance at service at least once every cycle, and usually much more frequently. But therein also lies a problem: the control program must inquire as to the state of every user every cycle. Though a single user request may require use of the CPU only briefly, a given user may be polled hundreds of thousands of times during a period when he is not himself interested in using the CPU.

One method of eliminating this essentially non-productive "overhead" is to give users access to the computer for longer periods of time, breaking them off whenever they request input/output and placing them in a queue to await service when the data transmission is complete. This technique is much more directly related to the goal of keeping the CPU busy during input/output operations. And a user causes little or no system overhead while he is pondering what to say next to his program. However, computational programs can tend to monopolize such a system unless corrective measures are taken. One such measure is the assignment of priorities to programs so that the highest priority program always has access to the CPU except when actually performing input/output, at which time the lower priority programs can use the CPU. If the priorities are assigned in decreasing order of input/output requirements, the balance should work out fairly well —— but inequities obviously can and do arise.

Some systems are essentially a combination of batch and on-line programs. The batch programs may be long-run production routines or other jobs requiring neither the presence of the user nor immediate completion of the job. These may be entered into the system in the time-honored way —— by giving a deck of cards to the operator —— or, perhaps, may be initiated from a time-sharing terminal. In any case, the job enters a queue of jobs waiting for batch service and is started in turn —— perhaps many hours after entry into the queue. These jobs form the "background" in a background-foreground system, with on-line users forming the "foreground." In effect, the background consists of users who always want service.

TIME-SHARING COSTS

- COMPUTER PROGRAM DEVELOPMENT
- INCREASE IN OVERHEAD TIME

The benefits of time sharing, of course, have their price. The increased complexity of the control program, as compared to simpler batch process control programs, results in higher development costs. An additional factor, as we have seen, is that time sharing results in a proportionately higher amount of processing overhead.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

TIME SHARING COST PER HOUR OF HOOKUP

BOLT BERANEK & NEWMAN	24.00
DARTMOUTH	29.75
IBM	39.25
CEIR	41.00
G. E.	41.50
APPLIED LOGIC	43.50

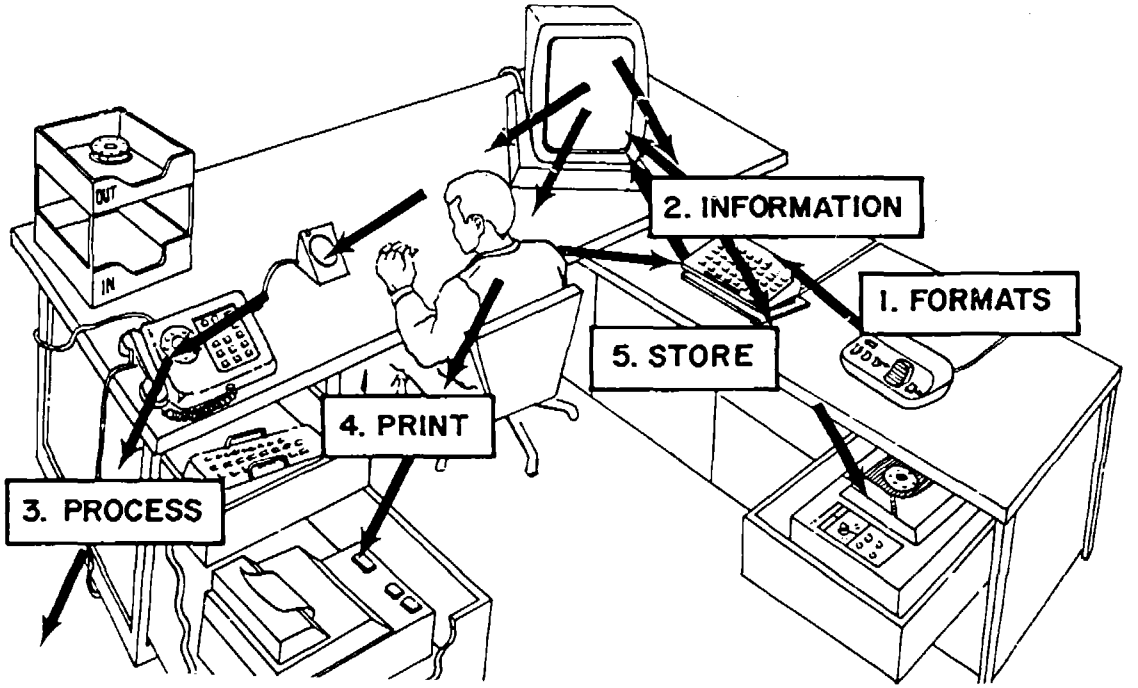
The figures in the opposite chart give some indication of how much it costs a user to "hook" his terminal into a time-sharing system. They apply to the 1967-68 time period.

Hourly usage is measured in elapsed time while the terminal is on-line, whether or not processing is being done. Although these prices may appear high when non-processing time is considered, they are noticeably less than the hundreds of dollars it would cost to hire the whole machine and pay for its idle time. Incidentally, terminal time is not the only basis for charges; other factors are items such as CPU time, channel use, and core space taken. These may be applied in any combination.

The figures cannot be directly compared, of course, because systems vary widely in power and applicability to particular types of problems.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

WORK STATION



The opposite diagram shows a typical user terminal configuration. The details of the particular system depicted are not important, but it should be noted that the user may gain access to the CPU in one of two ways: via the keys on his specially adapted telephone (left) or via his keyboard/formatter/display scope combination (right). The responses generated by the CPU may appear on his printer, his display scope, or may be recorded on tape. The CPU itself is probably some distance away; the user has voice communication with the computer operator, however, if he needs its.

The above photograph shows an actual terminal, illustrating another possible setup. The girl is using a "light pen," a device that permits direct drawing of figures or selection of characters on the face of a display tube.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

"SOFTWARE" LEVELS

$D = \frac{A+B}{C}$	PRINT PAYROLL	COMMON NOTATION
D = (A + B)/C	OPEN FILE A PRINT PAYROLL ON V CLOSE FILE A	HIGH LEVEL LANGUAGE
01 A + B → R 02 R/C → R 03 R → C	01 XY → 2V 02 ZV → PRI. AREA 03 DV → 01 04 ZV # 75 05 35 $ 35	SYMBOLIC LANGUAGE
010010011110 010111101110 010111101110 010111111000 010101111001 011101010111	25 { 010001000111 111001111110 011101010111	MACHINE LANGUAGE

The software approach to information handling problems was limited by software costs – not only for the control program software but for the user software as well. As these costs began to overshadow the hardware cost, the subject of proper software design gained prominence.

The term "level", as applied to software, has two meanings. In one sense, the level of a program is determined by the extent to which it initiates and controls the activity of other programs. For example, a time-sharing control program is of a higher level than the user programs in that it stops and starts them. On the other hand, when a user program wishes to initiate a data transfer, it does so by invoking a subprogram in the system; in that sense the user programs are of a higher level than the system subprograms. In another, seemingly unrelated sense, the word "level" refers to the closeness of the notation used in writing the program to the user's natural notation. We say "seemingly" because when one writes a program that is of a high level in the notational sense, it generally turns out to be a fairly high level program in the first sense also, both because a number of processors are implicitly called in to reduce the program to the binary level required by the machine, and because even the reduced program, when running, will typically invoke a number of pre-written routines such as the square root program we mentioned.

The chart on page 62 illustrates two problems, each expressed on four different notational levels. The topmost level is not programming notation but rather what a mathematician or business man might write on a sheet of paper or simply say to someone. The second level shows the same problem expressed in a high-level programming notation or "language." Now it must be admitted that many side issues have been ignored for the sake of making a point: what one actually writes down when programming in a high-level language can be very close to a natural notation, at least within the subject area intended by the language designers. On the other hand, when one writes in such a language a very complex program, called a compiler, must be used to scan what the user has written in order to generate its equivalent in machine language. Either that or an equally complex program, called an interpreter, must be used to scan the program and do what the user intends. Which is better depends upon whether the user wants to perform his process many times or seldom. In the first instance the compiler application is probably better because the first step — scanning the user's notation and figuring out what it means — is performed only once.

69

At the next (third) level in the chart, the problem is coded in a language that is very close to machine language in the sense that the programmer is directly aware of the bits and registers he is manipulating in the machine. It is called "symbolic" language because the programmer uses symbols rather than the binary addresses of data or the binary instructions. However, there is a rigid, usually one-for-one equivalence between a symbol and the binary number that is entered into the computer. The program which accomplishes this relatively simple transformation of symbols is called an assembler.

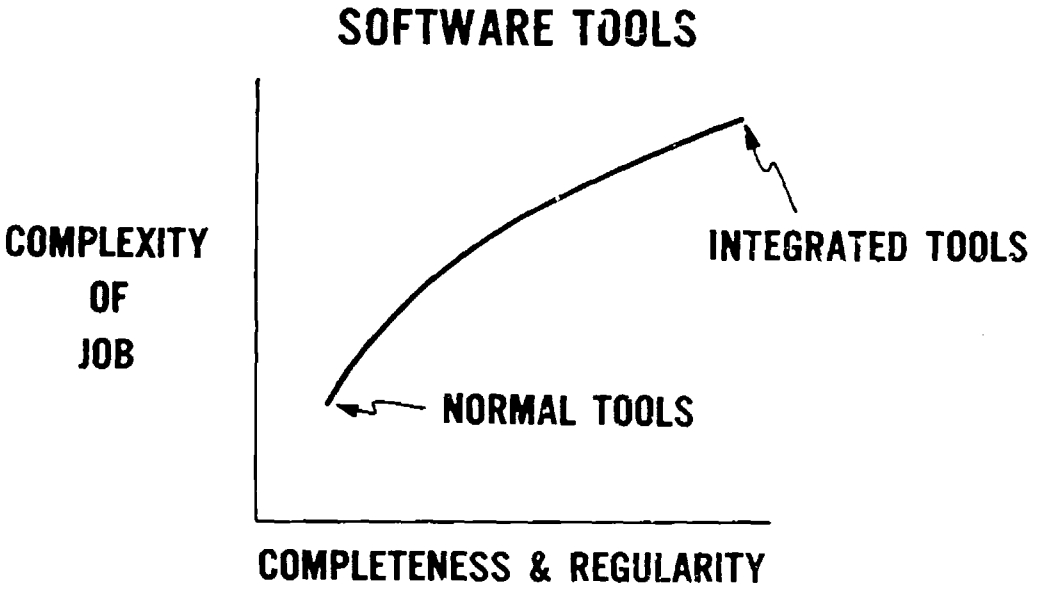
At the lowest lever is the binary language understood by the machine itself, as we have already discussed. Except under very special circumstances, the programmer rarely writes directly in this language today. In fact, the term "machine language" has come to mean "symbolic language" in common usage.

PROGRAMMERS' SALARIES

COMMERCIAL	
1 YR	$8400
1 - 2 YR	9800
2 - 4 YR	12,200
4 + YR	12,900
SCIENTIFIC	
1 YR	10,600
1 - 2 YR	12,100
2 - 4 YR	14,600
4 + YR	16,600
MANAGEMENT POSITIONS	15,000 AND UP

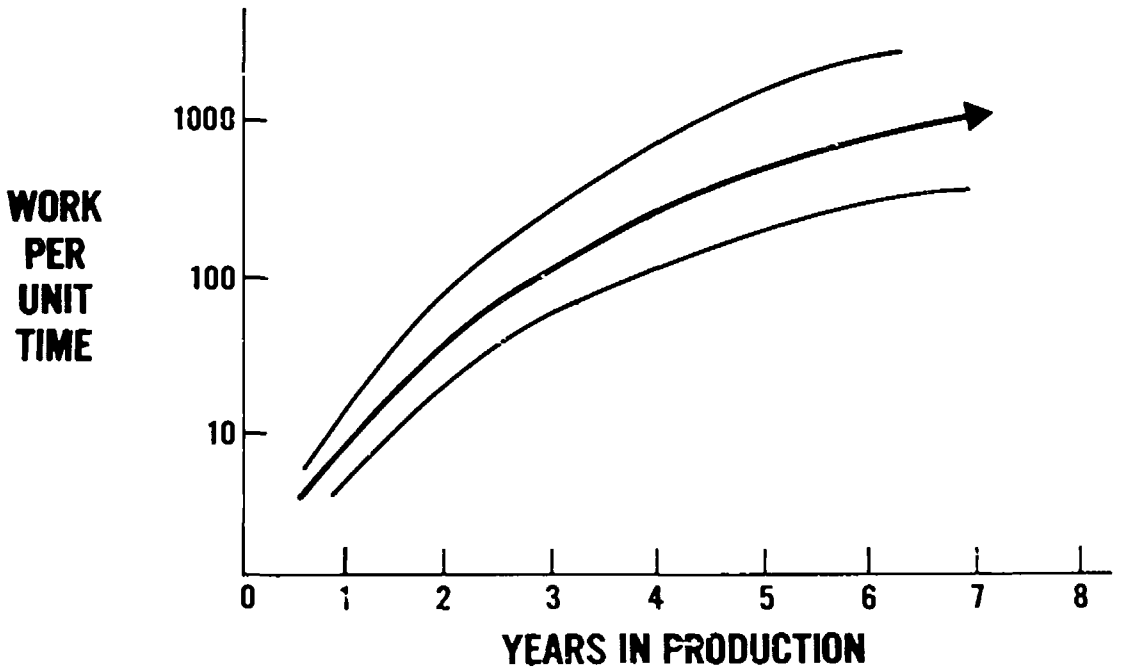
Here the opposite chart provides one of the reasons why software costs are outdistancing hardware costs.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING



Programmers produce programs at the various levels we have seen. Depending upon the level, and upon other factors, a programmer will need various software tools to assist him in getting that program running on the computer. Two essential tools are the compiler and the assembler; they might be called "normal tools." Others might be debugging aids and the like. While the independent development of a compiler is a complex job, the development of a compiler with built-in debugging aids is more complex still. However, the latter tool is more useful because it is obviously more complete and also because it is more regular, or unified, than separately developed tools. Fully integrated tools, which amount to "systems" or "environments" for programming, may take hundreds of man-years to develop.

COMPILER PERFORMANCE



One hundred man-years of effort, moreover, cannot necessarily be generated by putting 200 men to work for 6 months. The development of effective "workhorse" tools such as compilers tends to be an almost evolutionary process over a long period of elapsed time.

The above chart gives an impression of this fact. The labeling on the vertical axis is deliberately vague: the numbers are in a logarithmic scale in units of work accomplished per unit time for a user — it might be student averages calculated per hour for one user and bridges designed per month for another user of the same compiler.

The horizontal axis and the slope of the curve indicate that a simple compiler that eventually gets the job done, perhaps with a lot of grief to the user, may be thrown together rather quickly. But a highly integrated, easy-to-use and efficient tool may take several years to develop. Unfortunately, there is a kind of diminishing return that sets in as limitations implicit in the language design are approached.

COMPUTER PROGRAM TRANSFER

- RECOMPILE
- EMULATE
- SIMULATE
- RECODE

Since it costs so much and takes so long to develop software tools, it is obviously desirable to use tools already available, if they can be applied or easily adapted. This applies not only to general program production tools such as compilers but application-oriented tools such as program packages for the computation of structural stresses. In fact, the difference between the two is only a level of generality.

If the tool you want is available on your machine, it is usually just a matter of learning to use it. With most machines, the manufacturer usually supplies an assembler, compilers for several popular languages, and an application package. If the tool must be obtained by transfer from another computer, the chances of success depend upon the characteristics of the computers involved, the level of notation in which the tool is programmed, and other characteristics of the program. If the two computer processors are identical, and the input-output devices differ in ways that do not affect the program, a direct transfer is possible. A tool coded in assembly language can be transferred with modifications and reassembly to take care of minor

differences in input-output configuration or program function – but not, as a rule, differences in the processor. In general, though, the problem is to transfer between unlike computers, and then one of the four techniques listed in the opposite chart must be chosen.

If the program is written in a higher level language and a compiler for this language exists of the target machine, then sometimes a simple recompilation will effect the transfer. This technique often fails, however, because different compilers typically implement different dialects rather than one unified language and, in any case, many programs written in higher level languages actually rely, explicitly or implicitly, upon specific characteristics of the compiler used to translate them or upon specific qualities of the processor, core memory, or input-output devices that will be used by the generated machine language program. It is not always possible to avoid this kind of "machine dependence" when writing a higher-level program, nor is it always easy to adapt such a program for another machine by changing the dependencies.

When all else fails, it is always possible to pretend that the machine language used by the source machine is a "higher level" language than that of the target machine, and set about building a compiler or interpreter to implement this notation. Compilers of this sort are called "translators" and, for various technical reasons, are very rare. Interpreters of this kind are known as "simulators."

In some cases the processor hardware may be augmented to facilitate the operation of a simulator; the resultant hardware / software combination is called an "emulator." Simulator or emulator, the idea is the same: computer B is programmed to operate on instructions and data as they would normally be encoded in computer A. It is clear though why this approach is taken only when the amount of software to be transferred is extensive, and even then usually on an interim basis only. Beside the cost of building the simulator — a large tool in itself — this is a very inefficient mode of operation.

And, of course, the program simply can be rewritten. This doesn't always mean starting from scratch. Often the logical structure of the original program can be paraphrased.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

GENERAL PURPOSE DATA MANAGEMENT

- INFORMATION RETRIEVAL
- REPORT GENERATION
- SUMMARIES OF DATA

Within the larger realm of software, tools specifically designed for information processing are called "data management" systems. We may summarize what the user of such a tool would need. He has a large amount of data to work with, as we said earlier; he will want to be able to store, modify and retrieve it at will and in a manner natural to him. As an extension of the retrieval process, he will want to generate reports and summaries based upon larger subsets of that data, and perhaps involving complex relationships in the data itself.

PARAMETERS OF DATA MANAGEMENT PROBLEMS

- MULTIPLE SIMULTANEOUS USERS
- CURRENCY OF DATA
- PROCESSING OF DATA
- RESPONSE TIME

The manner in which these functions are carried out is of equal importance to the user. Because the "user" is often actually a group of people, with many different needs and functions with respect to the data, the possibility of having many users simultaneously accessing the same data is an important consideration. Also, the storage and retrieval facilities should be fast enough so that data is kept current and reports are timely. The possibility of specifying extensive processing of the data should not be excluded. And the response time for an individual request should be minimum.

EVALUATION

TWO METHODS OF EVALUATING DATA MANAGEMENT SYSTEMS

- TECHNICAL FEATURE EVALUATION
- OPERATIONAL PERFORMANCE EVALUATION

Two methods of evaluating software of any kind, including data management systems, are most frequently used. One is to go through a sort of checklist of technical features, specifying whether or not each system being evaluated has that feature, or in what quantity. The other is to take a representative or "benchmark" set of problems, and observe the performance of each system on these samples.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

TECHNICAL FEATURE EVALUATION

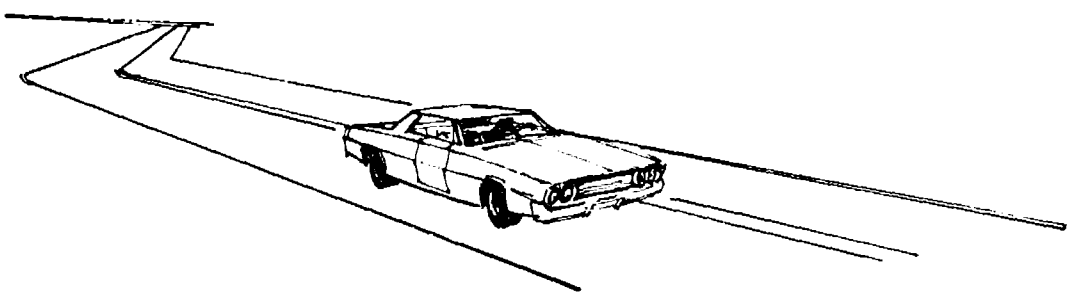
RETRIEVAL CHARACTERISTICS	SYSTEM A	SYSTEM B	SYSTEM C
RETRIEVE ANY ELEMENT FROM ANY HIERARCHICAL LEVEL OF A FILE	YES	YES	YES
RETRIEVE ELEMENTS THAT ARE USED AS SEARCH CRITERIA	YES	YES	YES
CAPABILITY TO BATCH RETRIEVALS IN ONE PASS	NO	NO	YES
RETRIEVE SIMULTANEOUSLY FROM TWO OR MORE FILES	YES	NO	YES

The above chart shows a typical software for system retrieval capabilities. Although useful for thumbnail comparisons, such lists are limited because the concept implied by a listed "feature" may vary a great deal from system to system and frequently does not apply at all. Also the evaluator must somehow rank the features according to his needs, taking into account interdependencies such as those features that make others unnecessary.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

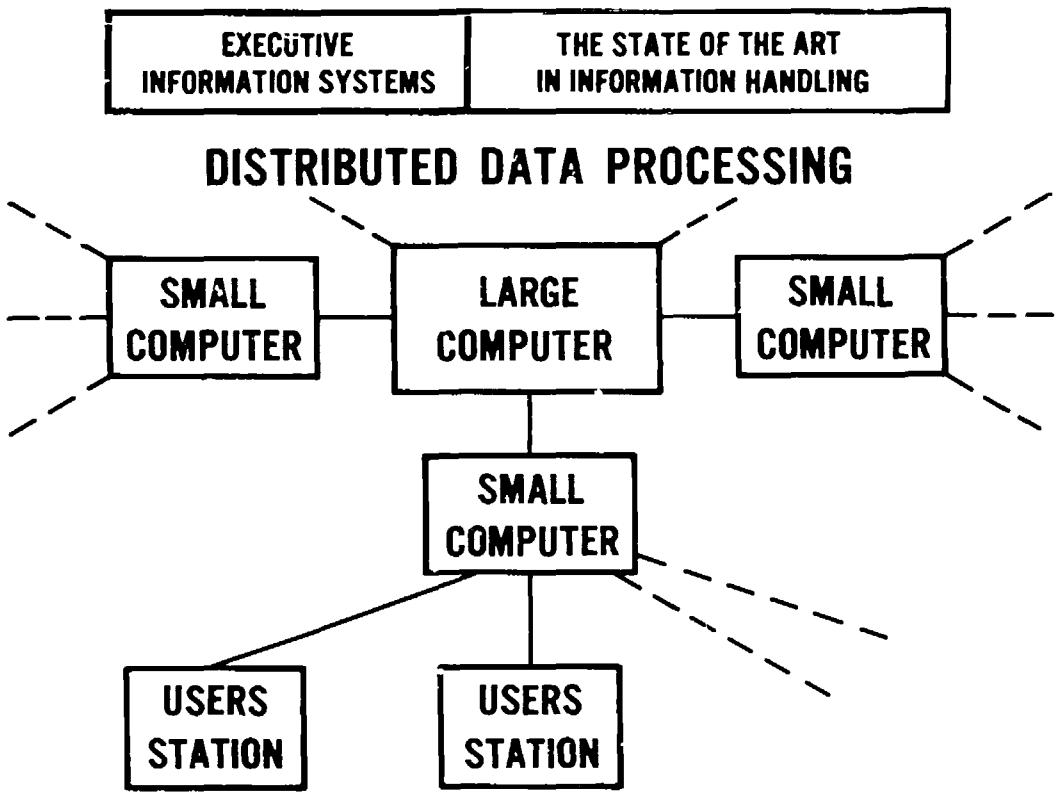
OPERATIONAL PERFORMANCE EVALUATION

TRY IT OUT

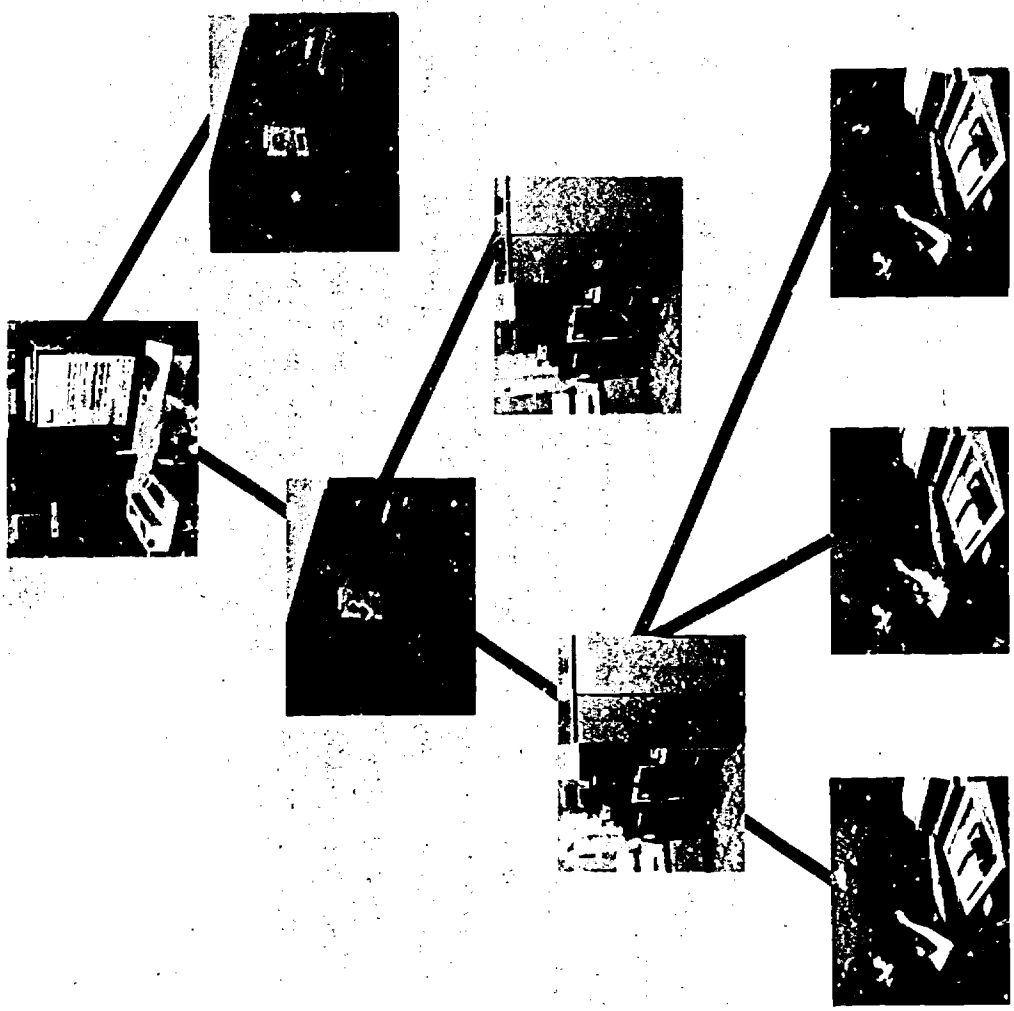


The benchmark approach is generally preferred, particularly when comparing systems for a proposed installation. The benchmark programs can and should be highly personalized to reflect the needs and tastes of the particular user. Then the system can be tried out, much as the buyer of an automobile will generally try several as well as read their specifications. Of course, the benchmark approach is more time-consuming and often more expensive than the checklist technique. And benchmarks do not necessarily provide a faithful measure of how well a given system can solve a particular problem, but rather how well it implements a particular solution to that problem, programmed by a particular person.

89


EXECUTIVE INFORMATION SYSTEMS
THE STATE OF THE ART IN INFORMATION HANDLING
DISTRIBUTED DATA PROCESSING
SMALL COMPUTER
LARGE COMPUTER
SMALL COMPUTER
SMALL COMPUTER
USERS STATION
USERS STATION

The latest data management systems employ not only proven software techniques, such as time sharing, but also a new concept in hardware configuration called distributed data processing. In this configuration computers linked together by communications lines – treating one another like input-output devices if you will. Typically, one or more large computers form the heart of the system. Large files and heavy computing potential are concentrated in these central installations. At some distance, perhaps thousands of miles away, are smaller computers. Except for the added link to the central computer, each of these smaller computers is a time-sharing system with local or remote users' stations such as we have seen. The idea is that the basic servicing of the users' stations, the processing of routine requests, and the accessing of purely local data can be handled by the small computer on its own. More complex processing requests and accesses to central data files can be handled automatically by queries from the local computer to the central installation.



The distributed data processing concept can be extended by introducing "regional centers" or other structural levels as needed. The pictures in the above figure depict the equipment one might see at various levels in a four-level system.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

WHY DISTRIBUTED DATA PROCESSING

- ECONOMIC IMPLICATIONS
- HARDWARE REDUNDANCY
- SHARED DATA ON DEMAND
- ANALYSIS OF DATA AT SOURCE
- ASSIST TO PERSONNEL PREPARING DATA

There are many reasons for turning to a distributed data processing system. Intuitively, it makes economic sense to have a simple, local machine performing simple, local tasks, while a complex, central machine performs complex, central tasks. The system grants access to a large computer for any user who needs it, yet the dependence upon a remote giant for even the most trivial operations is removed. Hardware redundancy is achieved in that the system as a whole is not vitally dependent upon the state of any one component's health. Data to be shared among several users may be stored at the appropriate level and accessed at will. Simple analysis of the data at the local level, including correctness checks when processing the data to the next higher level, frees the central processor from these chores and insures quick turnaround for tasks not requiring the services of the central computer. In particular, it becomes feasible to monitor the preparation of data during the preparation process itself, to provide general assistance to persons drawing up the data, and perhaps to catch errors before they snowball into major headaches.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING

DISTRIBUTED DATA PROCESSING PROBLEMS

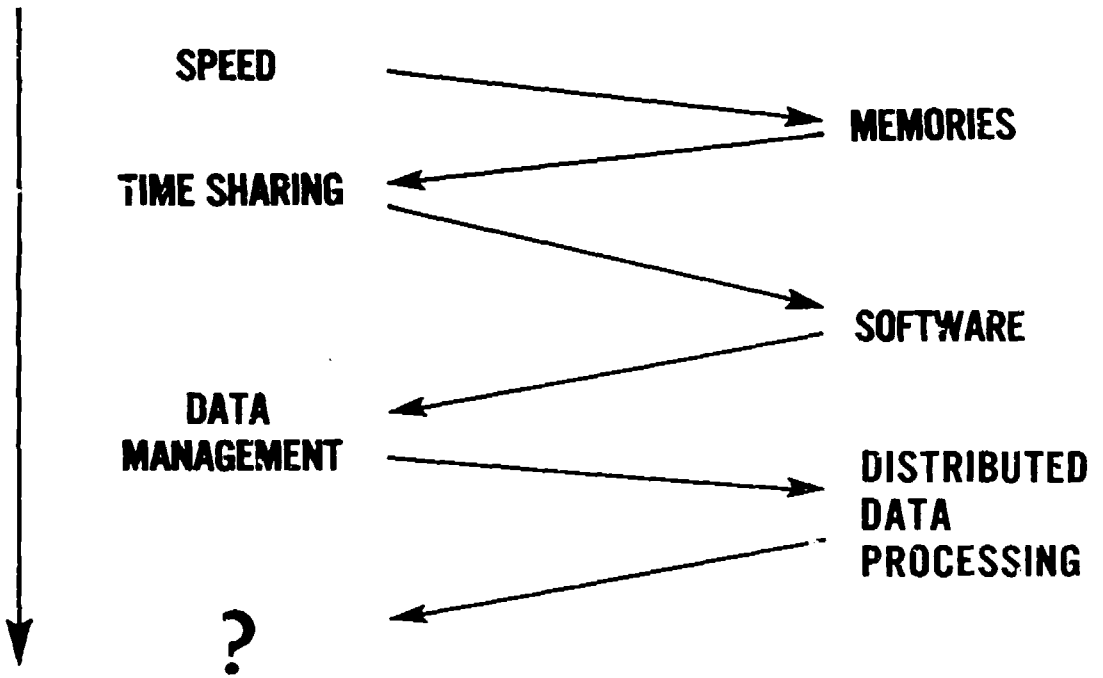
- STANDARD DATA ELEMENTS AND CODES
- COMMUNICATION BANDWIDTH
- INTERCOMPUTER COMMUNICATIONS STANDARDS

Nothing so beneficial as distributed data processing can be implemented without encountering a few problems. One is that the various computers in the system tend to have very different standards for encoding data, to say nothing of differences in programming languages. This problem could easily be avoided by selecting compatible hardware from one manufacturer only. However, this solution ignores the differing needs of various potential users and discards the possibility of incorporating much existing equipment into the network. It also places too much dependence on one source and, in any case, may be politically unacceptable. Data code differences, however, are not usually too serious: hardware or software translators can be built fairly easily.

Less easily overcome is the difficulty of getting large volumes of information transmitted from one place to another. Many systems exist for transmitting data over ordinary voice-grade telephone circuits. However, these are limited to very low transmission rates. Data rates characteristic of computer-to-computer transmission require very high quality or "wide bandwidth" circuits and these, of course, are expensive.

Finally, there are the differences from one piece of equipment to another in the way in which transmission is accomplished. Again, we could standardize on one type of equipment, but this is rarely an attractive method. Usually, this problem is approached by building hardware "interfaces" to smooth out the differences.

EXECUTIVE INFORMATION SYSTEMS	THE STATE OF THE ART IN INFORMATION HANDLING



95

And so here we are at the "state of the art." What purely technical turn we will take from here is a matter for educated guesses, and is a subject all in itself. But one fact is clear: the area of applications for computers is widening at an accelerated pace. This is especially true in the social sciences, which seems appropriate when we recall that the Bureau of the Census was among the earliest punch-card pioneers. We have problems, as we have seen. And more problems, generated by progress itself, lie ahead. Safeguards for individual privacy, for example, must certainly be more fully developed in systems of the future than they are now, simply because otherwise it would be all too easy to invade that privacy. With proper controls, however, there is every reason to suppose that today's problems will be tomorrow's solutions. And tomorrow's problems —— well, we'll have to cross that bridge when we get to it.