SCHAUM'S OUTLINE SERIES

THEORY AND PROBLEMS OF

COMPUTERS and PROGRAMMING

FRANCIS SCHEID

INCLUDING 535 SOLVED PROBLEMS

SCHAUM'S OUTLINE SERIES IN COMPUTERS

McGRAW-HILL BOOK COMPANY

SCHAUM'S OUTLINE OF

THEORY AND PROBLEMS

of

COMPUTERS and PROGRAMMING

•

BY

FRANCIS SCHEID, Ph.D.
Professor of Mathematics
Boston University

•

SCHAUM'S OUTLINE SERIES
McGRAW-HILL BOOK COMPANY
New York St. Louis San Francisco Auckland Bogotá Guatemala Hamburg Johannesburg
Lisbon London Madrid Mexico Montreal New Delhi Panama Paris
San Juan São Paulo Singapore Sydney Tokyo Toronto

FRANCIS SCHEID is Professor of Mathematics at Boston University where he has been a faculty member since receiving his doctorate from MIT in 1948, serving as department chairman from 1956 to 1968. In 1961–62 he was Fulbright Professor at Rangoon University in Burma. Professor Scheid has lectured extensively on educational television and the videotapes are used by the U.S. Navy. His research now centers on computer simulations in golf. Among his publications are the Schaum's Outlines of *Numerical Analysis* and *Computer Science*.

Schaum's Outline of Theory and Problems of
COMPUTERS AND PROGRAMMING

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 SH SH 8 6 5 4 3 2

ISBN 0-07-055196-0

Sponsoring Editor, John Aliano
Editing Supervisors, Lester Strong, Margaret Lamb
Production Manager, Nick Monti

Library of Congress Cataloging in Publication Data

Scheid, Francis J.
Schaum's outline of theory and problems of computers and programming.

(Schaum's outline series)
Includes index.
1. Electronic digital computers—Programming. 2. Programming languages (Electronic computers) 3. Electronic digital computers—Programming—Problems, exercises, etc. 4. Programming languages (Electronic computers)—Problems, exercises, etc. I. Title. II. Series.
QA76.6.S383 001.64'2 81-18572
ISBN 0-07-055196-0 AACR2

To Barb
my girl since we were ten

Preface

One main goal is pursued throughout this book: problem solving with the help of computers. The opening chapter is not typical, being a brief introduction to computers, their history and terminology. Chapter 2 then presents a framework for algorithm development, and the next two chapters translate these ideas into programming languages. The remaining chapters cover problem analysis to programming, using examples of varying complexity and background. Classic data structures and procedures of computer science are introduced as needed, but the goal is always to find a trail from problem to solution.

Four programming languages have been used: Fortran, Basic, Pascal, and PL/I, but no effort has been made to develop any of them fully. Instead their common features and purpose have been emphasized to encourage development of at least a reading acquaintance of all four. Since, however, various dialects do exist, local variations may have to be made in some programs.

Often enough problems have been worked out very patiently with explanations and analysis covering several pages. The same explanations have not usually been repeated within the eventual programs. That is, documentation is frequently a part of the text rather than part of the program. It woud be good exercise to incorporate vital segments of such preliminary work as program comments, and the reader is urged to take on this assignment.

This book is for all those who enjoy thinking things through from start to finish.

FRANCIS SCHEID

Contents

CONTENTS

Chapter 1

Information Processing

1.1 Information

Computer science involves the development and use of equipment and procedures for processing information. Information is presented to the processor in one form and returned in a different, presumably more useful, form. The former is the *input information,* or *data;* it is the raw material of the process. The latter is the *output information,* or *finished product* (Fig. 1-1).

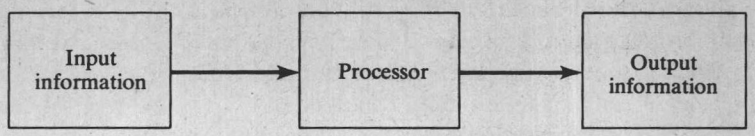


Fig. 1-1

Example 1.1 Compute 12*43.

This statement is the input information. An elementary school child will understand it and can do the processing. Here is one familiar procedure:

$$\begin{array}{r} 12 \\ \underline{43} \\ 36 \\ \underline{48} \\ 516 \end{array}$$

There are other ways of arriving at the same result, and this is typical of information processing. One could write twelve 43s in a column and add them up. Still another method consists of continually doubling one factor while halving the other, ignoring the remainders, until a factor of 1 is reached. Where the halved factors are odd numbers, the doubled factors are starred and then added:

12	43	*12	43
6	86	*24	21
3	172*	48	10
1	344*	*96	5
		192	2
		*384	1

Their sum will be 516. So a variety of methods is available for processing 12*43 into 516. Notice, however, that anyone unfamiliar with decimal symbols and procedures would find no meaning in any of the information involved in this example.

Example 1.2 Arrange the names JONES, HOGAN, HAGEN, SARAZEN, and NELSON in alphabetical order.

The five names and the instruction to alphabetize them are the input information. For anyone familiar with the language, it is no great challenge to process this information and generate the output shown in Fig. 1-2.

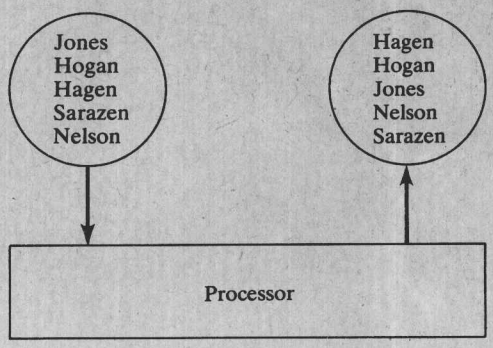


Fig. 1-2

It is not easy to give a simple, precise definition of the term "information." The idea is related to other elusive concepts such as "thought" and "meaning." Information can be exchanged between intelligent sources and translated into different languages, and this is one of the troubles. Unless one understands the language, the information may be "uninformative." This remark applies to both electronic *and* human information processors. Computing machines do exhibit certain aspects of intelligence. They are capable of receiving, remembering, processing, and outputting information, provided the input is in a language that they "understand." A number of languages have been developed to facilitate the communication of information between person and machine, and some of these will be explored in the following chapters. The success of such communication efforts can be measured in part by the explosive growth of the computer population (essentially zero in 1950) and our increasing dependence upon it. Questions such as

Are computers really intelligent sources?

Is the processing that they do a form of thinking?

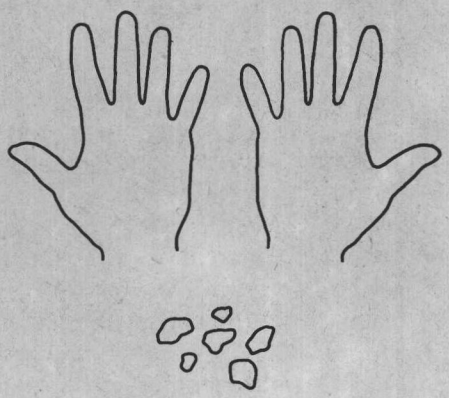
Can they have original ideas or learn from experience?

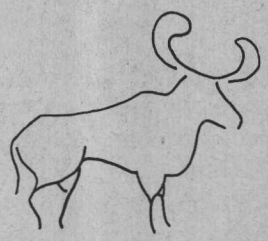
What is the future of the person-machine relationship?

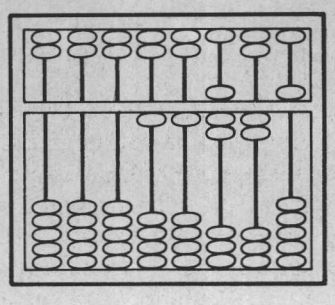
are provocative and will be discussed to some extent in the final chapter. They belong to what is known as "the field of artificial intelligence." For the most part, however, our time and energy will be spent on more functional aspects of our subject, particularly the preparation of input information that will allow computers to help in the solution of problems.

1.2 Early History

The management of human affairs from earliest times right up to the present has required cooperation, a willingness to work together. This in turn has involved the communication, understanding, and evaluation of ideas, using whatever tools were available for the purpose. In modern terminology the tools for handling information include *hardware* and *software*. Roughly put, hardware is equipment and software is ideas. (This rather liberal use of the term "software," which usually is restricted to computer programs, may be forgivable during this brief look at the historical record.) Figure 1-3 is a light-hearted attempt to suggest that initially resources were fairly meager, but that with the passage of centuries important developments did occur. Among the hardware items shown are the digits of the hand, from which the word "digital" is derived, and pebbles used for counting, once called "calculi," from which the modern word "calculator" (among others) comes. Also shown are an information processor that has endured for thousands of years (the abacus), two early media for recording information (knotted cords and a quill with scroll), a more recent storage medium (paper) and, whimsically at the bottom, a device for retrieving information from storage. Software items include three number systems, all still in use. The simulated sticks of the ancients continue to find a place in







$\cancel{IIII}$ III

MDCXX
$+$ MCM $\qquad \alpha, \beta, \gamma$

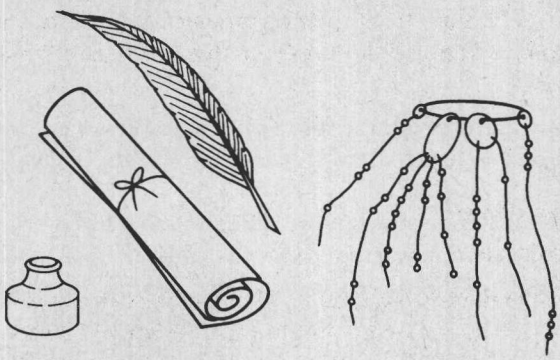
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

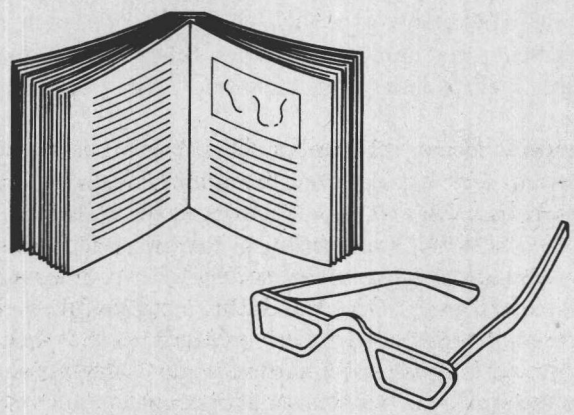
AB · · · YZ

$+ \quad - \quad \times \quad \div$

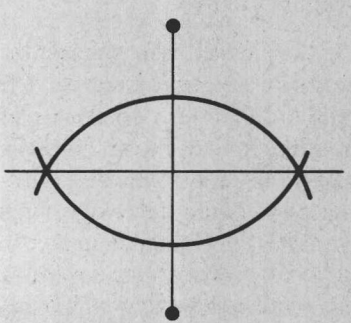
$\frac{1}{2} \qquad 3.14 \qquad \sqrt{2}$

$$\begin{array}{r} 147 \\ 12\overline{)1775} \\ \underline{12} \\ 57 \\ \underline{48} \\ 95 \\ \underline{84} \\ 11 \end{array}$$





Hardware



Software

Fig. 1-3

tallying operations, Roman numerals still add an element of style to cinema leaders and elsewhere, and the decimal system has withstood the challenge of base 2, base 12, and other contenders. Three alphabets are represented, including a picture-character of the cave dwellers, a few Greek letters, and our familiar A, B, C, ... Two examples of procedures for solving problems also appear, one for a long division, the other for bisection of a line segment. These are a bit closer to the modern concept of software. Finding procedures for problem solving, and instructing a computer to carry them out, will be our principal objective. Hardware and software are, of course, pretty much inseparable. After all, ideas have to be recorded if they are to be preserved and communicated. The script and figure visible in the opened book are examples of software recorded on hardware. Similar examples will be provided shortly using other media. In a sort of ultimate sense, a case can be made for viewing the human brain as hardware, with all of one's personal memories recorded upon it.

1.3 Toward the Computer

An important software discovery was made around the year 1600. Comparing the two rows of numbers shown in part as Fig. 1-4*a*, Napier realized that to multiply two of the numbers in the bottom row (take the 8 and 64 for example) it was only necessary to add the numbers above them (3 and 6). The product could then be read under the computed sum (512 is under 9). He had found a way to obtain products by addition. Napier called the numbers in the top row *logarithms.* His computation can be summarized in this form:

$$\begin{array}{lll} & \log\ 8 = 3 & \\ 8, 64 & \log\ 64 = 6 & 512 \\ & \log 512 = 9 & \end{array}$$

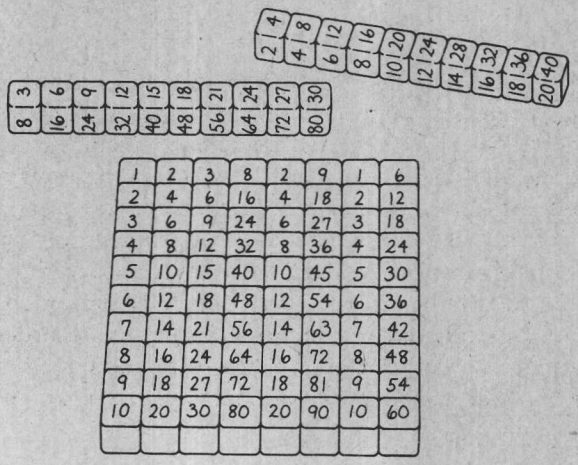
The logarithm idea was developed so that almost any useful product could be found, tables of logarithms being worked out for this purpose. Since addition is easier than multiplication, the tables found extensive use.

New software soon led to new hardware. Napier copied some of his tables onto a set of ivory rods, which became known as *Napier's bones,* and by positioning the rods was able to find products mechanically (Fig. 1-4*b*). The eventual development in this direction was the *slide rule,* which only a generation ago was the trademark of the engineer (Fig. 1-4*c*). The slide rule used logarithmic scales, which made it possible to multiply by moving a sliding scale so that appropriate lengths were added. Because of physical limitations in marking the rulings, and visual limitations in seeing them, about three correct digits could usually be obtained and this came to be called "slide rule accuracy." The slide rule is an example of an *analog processing device,* which means that numbers are represented by positions on a scale rather than by digits. Other analog devices include measuring sticks and dials. Until quite recently most clocks were of the analog type.

In the 1640s Pascal, then still a teenager, designed a mechanical calculator (Fig. 1-4*d*). It was not a commercial success because of frequent breakdowns, and Pascal turned his talents to mathematics instead. A modern computer language bears his name. Several other efforts to build mechanical calculators followed, but with limited impact. In the 1820s Babbage began his lifetime project of building an "analytical engine." It was to be a general-purpose machine, one that could be programmed to evaluate a wide range of arithmetical expressions. One of Babbage's engines appears as Fig. 1-4*e*. His design included several features present in today's computers, such as storage capacity, a "mill" that could do an addition in 1 s, a typewriter for output, and punched cards. Unfortunately Babbage's ideas were well ahead of the hardware technology of his time, and his engine could not be made to perform reliably. In the 1880s the U.S. Bureau of the Census employed Herman Hollerith to develop equipment and procedures for handling census data. The punched-card operation that he devised processed the 1890 census in one-fourth the time estimated from experience with manual methods.

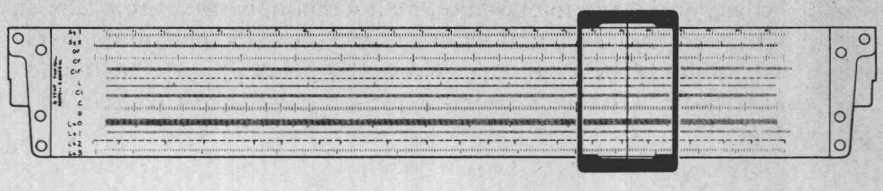
1 2 3 4 5 6 7 8 9 · · ·
2 4 8 16 32 64 128 256 512 · · ·

(*a*) The logarithm idea

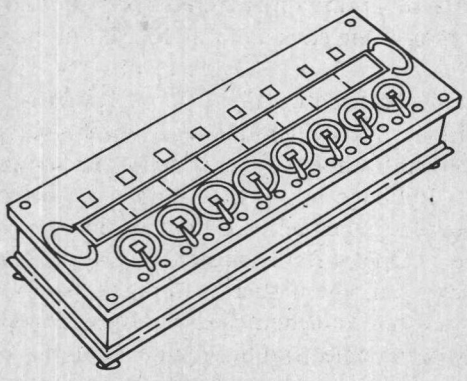


1	2	3	8	2	9	1	6
2	4	6	16	4	18	2	12
3	6	9	24	6	27	3	18
4	8	12	32	8	36	4	24
5	10	15	40	10	45	5	30
6	12	18	48	12	54	6	36
7	14	21	56	14	63	7	42
8	16	24	64	16	72	8	48
9	18	27	72	18	81	9	54
10	20	30	80	20	90	10	60

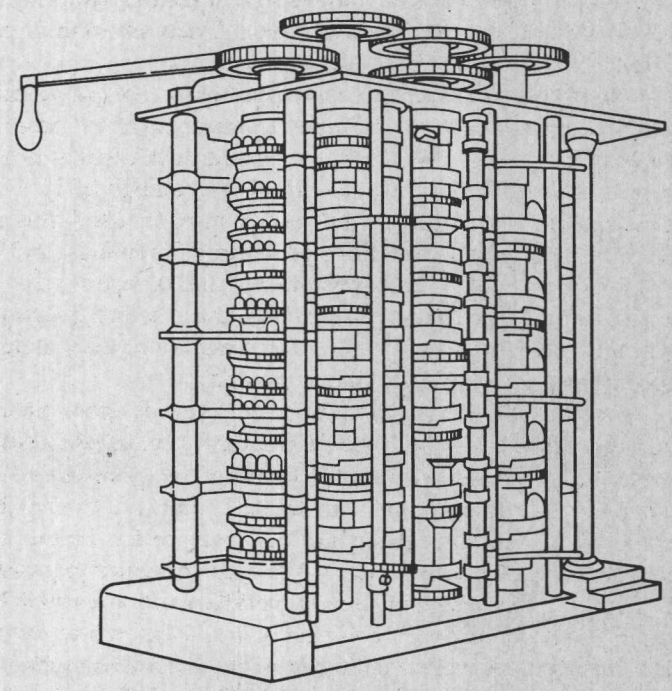
(*b*) Napier's bones



(*c*) A slide rule



(*d*) Pascal's calculator



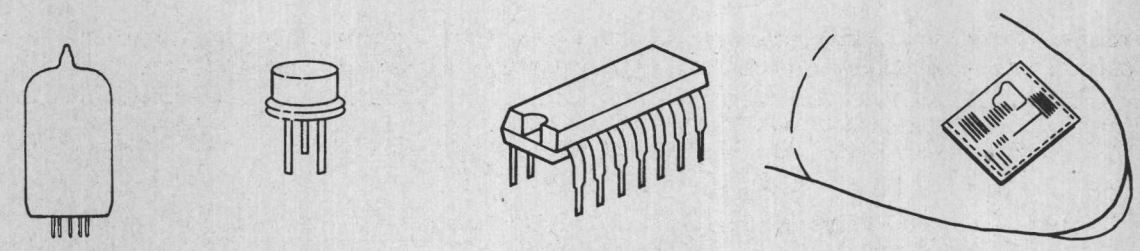
(*e*) Babbage's engine

Fig. 1-4

1.4 Generations of Computers

Electronic computers have existed only since around the middle of this century. In the 1940s ENIAC, the Electronic Numerical Integrator and Calculator, was designed and built at the University of Pennsylvania. In 1949 EDSAC, the Electronic Delay Storage Automatic Computer, was built at Cambridge University. In these machines vacuum tubes took over functions that had previously been performed mechanically. The arrival of electronics upon the processing scene meant that calculations could be performed about a thousand times faster than by hand. An accompanying innovation was the storage of instructions as well as data in the computer's memory. The set of instructions for a given job was called a *program,* and the machines were *stored program computers.* This was a significant step, since formerly instructions were fed sequentially from a mechanical source, slowing operation to the speed of such sources. With stored programs, the calculations could run at electronic speed. Vacuum tube computers are now known as "first generation" machines. In 1951 the UNIVAC became the first commercially available processor of this type, earlier machines having been one of a kind.

First generation computers did a very impressive amount of information processing, both for science and business, but were not without problems. Tubes generate a lot of heat and frequent replacement was needed. System "crashes" were common, requiring engineers to activate their troubleshooting routines for locating faulty tubes and other broken parts. In the late 1950s the solid state transistor was introduced, partly taking over the role of the vacuum tube. Smaller and more reliable, the transistor became the identifying feature of the second generation computer. Technological advances began to come more rapidly. Integrated circuits were perhaps the next breakthrough, complex circuits produced in a single integrated process and capable of performing the same functions as conventional circuits with their numerous discrete electronic components. In the early 1960s these became the mark of the third generation computer. Since then the conspicuous trend in machine design has been miniaturization. Hundreds of entire integrated circuits can now be placed on a single "chip" about an eighth of an inch square. Some chips are designed for processing and others for memory. Their existence is often taken as the sign that a fourth generation of computers has arrived. The importance of miniaturization is evidenced by the increased computing speed and decreased costs that it has brought about. Electricity does move at considerable speed (nearly the 186,000 mi/s of light) but even for electricity a shorter path means a faster passage. Tiny chips holding many circuits have substantially reduced the travel time. Figure 1-5 shows some of the pivotal items of hardware that have just been described.



Vacuum tube (First generation) | Transistor (Second generation) | Integrated circuitry (Third generation) | Miniaturization (Fourth generation)

Fig. 1-5

1.5 Representing Information

Information is recorded and communicated using various forms of symbolism or code. From cave drawings to Roman numerals to our present alphabets and number systems, a progression of symbols has been developed. Serious efforts have been made, and are continuing, to teach computers to recognize our alphabetic and numerical characters. There has been some success, the magnetic ink characters appearing on millions of checks being one of the notable examples. A glance at Fig. 1-6 will show that a few concessions have been made to achieve this success, characters being modified to reduce the chances

Fig. 1-6

of confusion. For the most part, however, communication with computers still depends on special codes for representing information devised particularly for that purpose. The punched card shown in Fig. 1-7 is still in extensive use, and shows one such code. Each character is represented by a set of holes in a column of the card. The numerical digits use only a single hole, the letters of the alphabet two, while certain other characters use three. When a card is read, by a card-reading unit, the positions of the holes are sensed electrically and this information is conveyed to other parts of the computer. When a card is to be punched as output, the process is reversed, information from within the computer being translated into hole patterns and sent to a card-punching unit.

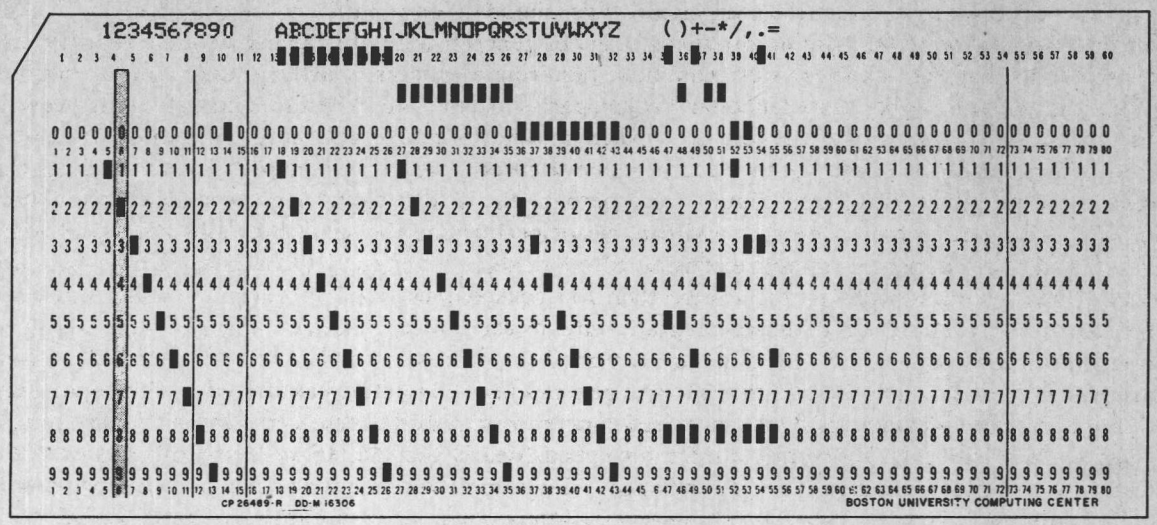


Fig. 1-7

Example 1.3 When a card is punched, the character represented in each column is usually printed simultaneously above the column, to facilitate error checking. The error in Fig. 1-8 is easily detected in this way.

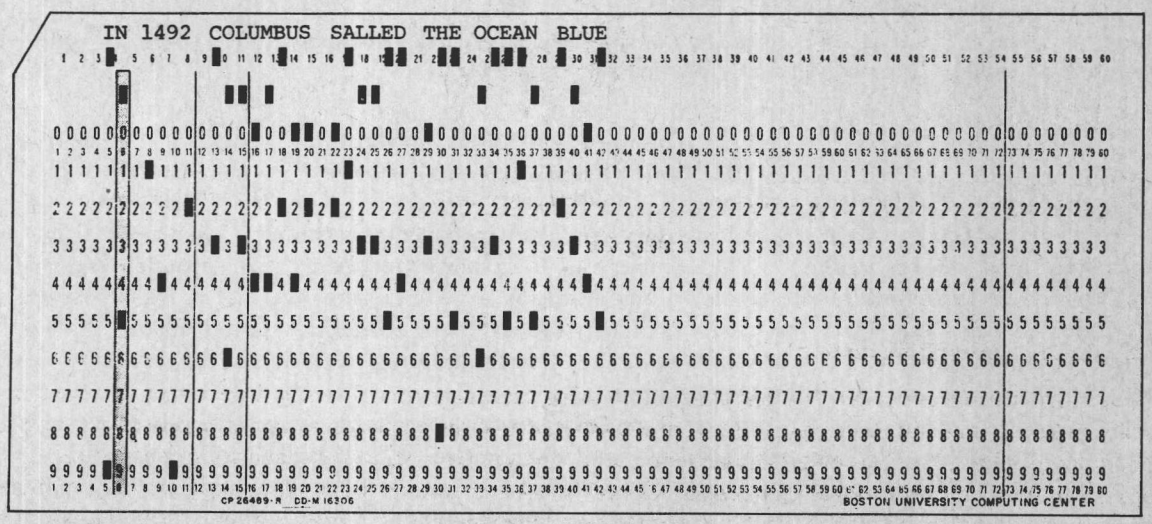


Fig. 1-8

The basic unit of information is the *bit.* It is the amount of information provided by a true-false or yes-no answer, or by an on-off switch. That is, of two possible states one is designated. Symbolically the digits 0 and 1 are often used for the two states. It is easy to see why the bit is a natural unit for working with electrical machines. At a given moment an electrical circuit may be carrying a current, or not carrying a current. The current may be flowing in one direction, or in the opposite direction. A voltage may be high, or low; an item may be magnetized, or not. Each of these is an alternative; of two electrical values, precisely one will exist.

One of the better known applications of bits is the *binary number system,* in which numbers are represented by strings of 0s and 1s. The underlying principle is illustrated by these examples:

Decimal	Binary
0	0
1	1
2	10
4	100
8	1000
16	10000
32	100000
64	1000000
128	10000000

They show that the idea of place value, used so successfully in the decimal system, is again involved. The difference is that a shift of the 1 digit leftward by one place only doubles its value, instead of multiplying it by 10. By combining the above "powers of 2" other numbers can also be represented.

Example 1.4 $3 = 11 \quad (2 + 1) \qquad 6 = 110 \quad (4 + 2)$
$5 = 101 \quad (4 + 1) \qquad 7 = 111 \quad (4 + 2 + 1)$

It becomes evident that 3 bits are adequate for numbers from 0 to 7 (000 to 111).

Example 1.5 Using 4 bits the coverage can be extended:

$9 = 1001 \quad (8 + 1) \qquad 12 = 1100 \quad (8 + 4)$
$10 = 1010 \quad (8 + 2) \qquad 15 = 1111 \quad (8 + 4 + 2 + 1)$

Notice that when all positions are occupied by 1 bits, the next larger number is a power of 2. Its binary symbol is thus a single 1 in the next position leftward, followed by a string of 0s:

1, 2	1, 10	15, 16	1111, 10000
3, 4	11, 100	31, 32	11111, 100000
7, 8	111, 1000	63, 64	111111, 1000000

From the bit point of view, the code used with punched cards seems wasteful. There are 12 rows, or punch positions, in each column. The number 5 is, for example, coded as a single punch in position 8. If punch and no-punch are designated momentarily as 1 and 0, this means that 5 = 000000010000 in this code. Twelve bits are being used. Similarly the letter A is coded as 100100000000. Using so many bits per character would be an extravagant use of machine storage capacity, so conversions to other representations are made for internal operations.

A somewhat more compact code, and one that is extensively used, is partially illustrated in Fig. 1-9. It is called *Extended Binary Coded Decimal Interchange Code* (EBCDIC), and uses 8 bits per character. This means that 256 combinations are available. For the 10 decimal digits 0 to 9, the first 4 bits are set to 1111, the others being the familiar binary symbols. Both uppercase and lowercase alphabetic characters are included, as well as a broad variety of special symbols, and still not all the 256 possible combinations have currently assigned meanings.

Character	EBCDIC	Character	EBCDIC
0	11110000	V	11100101
1	11110001	W	11100110
2	11110010	X	11100111
3	11110011	Y	11101000
4	11110100	Z	11101001
5	11110101	(blank)	00000000
6	11110110	.	01001011
7	11110111	(	01001101
8	11111000	+	01001110
9	11111001	&	01010000
A	11000001	$	01011011
B	11000010	*	01011100
C	11000011	)	01011101
D	11000100	,	01101011
E	11000101	%	01101100
F	11000110	?	01101111
G	11000111	'	01111101
H	11001000	=	01111110
I	11001001	''	01111111
J	11010001	a	10000001
K	11010010	b	10000010
L	11010011	etc.	
M	11010100	i	10001001
N	11010101	j	10010001
O	11010110	k	10010010
P	11010111	etc.	
Q	11011000	r	10011001
R	11011001	s	10100010
S	11100010	t	10100011
T	11100011	etc.	
U	11100100	z	10101001

Fig. 1-9

Example 1.6 The message in Fig. 1-10 decodes to the familiar TO BE OR NOT TO BE. (See also Fig. 1-9.)

```
11100011 = T
11010110 = O
00000000 =
11000010 = B
11000101 = E
00000000 =
11010110 = O
11011001 = R
00000000 =
11010101 = N
11010110 = O
11100011 = T
00000000 =
11100011 = T
11010110 = O
00000000 =
11000010 = B
11000101 = E
```

Fig. 1-10

Many other codes have been devised. The American Standard Code for Information Exchange (ASCII) is a 7-bit code. It was developed by the communications industry in an effort to standardize intersystem communications. It offers 128 patterns including these examples, from which many other symbols can be guessed:

0 = 0110000	P = 1010000
1 = 0110001	Z = 1011010
9 = 0111001	a = 1100001
A = 1000001	z = 1111010
O = 1001111	? = 0111111

Because some computers were not designed to accept 7-bit patterns, the effort at standardization was not a complete success.

Example 1.7 The brief message

1000001
1010011
1000011
1001001
1001001

reduces to ASCII.

1.6 Computer Organization

As suggested by Fig. 1-1, information enters a processing system in one form and leaves it in another. The input can be provided in a variety of ways. *Punched-card readers* were one of the first devices used for this purpose. The positions of the punched holes are a coded form of information, and these positions are sensed electrically by reading brushes. Photoelectric card readers have also been developed, and card input is still very common. Cards can be read at rates of several hundreds per minute, the mechanical nature of the process imposing an upper limit. *Magnetic tape* units allow much faster transmission of information, hundreds of times faster than card reading. *Remote terminals* can be used to enter data, at retail sales checkouts, at airline reservation desks, or in university laboratories. Input speed depends upon the typing skill of the operator but is slow relative to that of other devices. *Diskettes,* otherwise known as floppy disks because of their flexibility, are similar to ordinary records and are read by disk drives. The *magnetic ink reader* used for check identification in banking operations is a specialized input device. Even *optical character recognition* is now possible, provided that the symbols to be sensed have been standardized. Ordinary handwriting, for example, comes in too many styles to make machine reading very reliable. In short, there is a wide variety of methods available for entering data into an information processing system. Some of the devices involved are illustrated in Fig. 1-11.

Upon entry, information has to be retained, or stored. A computer storage facility, or memory, usually consists of two parts: *main storage* and *auxiliary storage.* Main storage can be thought of as part of the real "brain" of the system. Depending upon the computer's size, it may have capacity to hold several thousands or several millions of characters. Whatever instructions and data are needed for work in progress will be stored in the main memory. The hardware used for this unit is, therefore, of a type that allows rapid insertion and retrieval of data, so that maximum processing speed can be maintained. The two prominent types of hardware are currently *silicon chips* containing *integrated circuits* and *magnetic cores.* Both types allow any item of stored information to be accessed independently, without the need to work through a file of information until the place of the item wanted has been reached. For this reason they are known as *random access memory* (RAM) units.

The operational requirements of main storage make it expensive, and it is usually necessary to have auxiliary, or secondary, storage capacity available. Cheaper, and with slower access speeds, such units can hold information not immediately needed in processing. Types of auxiliary storage include *magnetic disks* or *drums,* which offer random access to data items, and *magnetic tapes,* which are