NCR

NCR ELECTRONIC DATA PROCESSING

WRITTEN FOR THE LAYMAN

2. What is Binary Arithmetic? #8437

2

What is Binary Arithmetic?


7
8
3

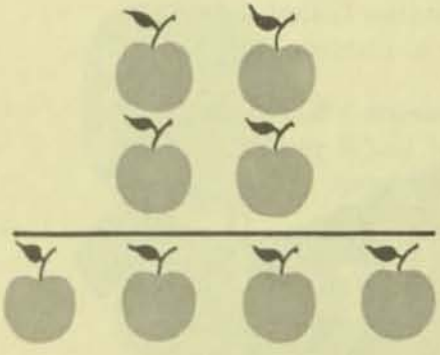
We do not intend to go into a complete history of numbering systems. However, as a student of data processing, it is important that you become familiar with a few of the basic methods of counting . . . and how certain machines perform these functions.

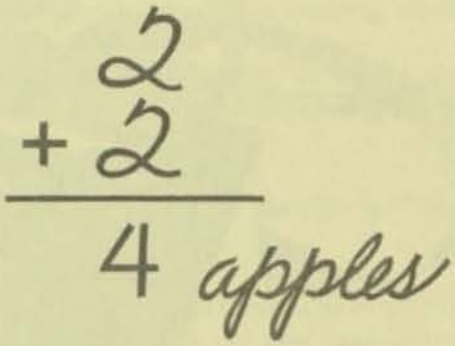
For example, as a child, our friend Pete first learned to count using his fingers.

A little later, Pete learned to write. Thus, he learned to use a tool called a pencil and a media known as paper. Mentally, he probably still visualized four fingers being raised to represent the quantity four . . . however, he progressed to a point where he did not rely completely upon his fingers.

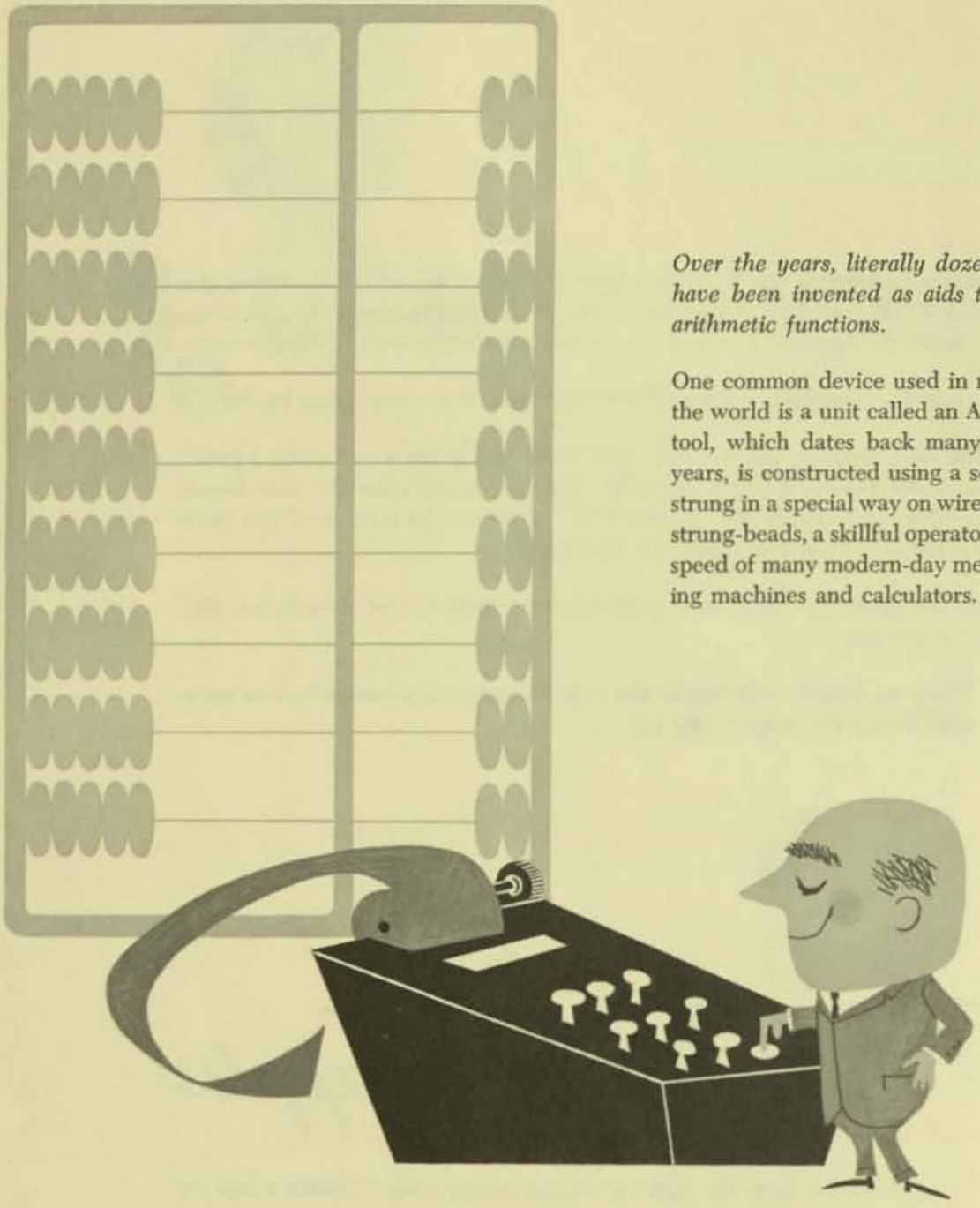
While this discussion might seem extremely elemental, it does serve to establish a very important point:

> When we hold up two fingers, this is just one way of representing two units, each having the value of one (1) . . .





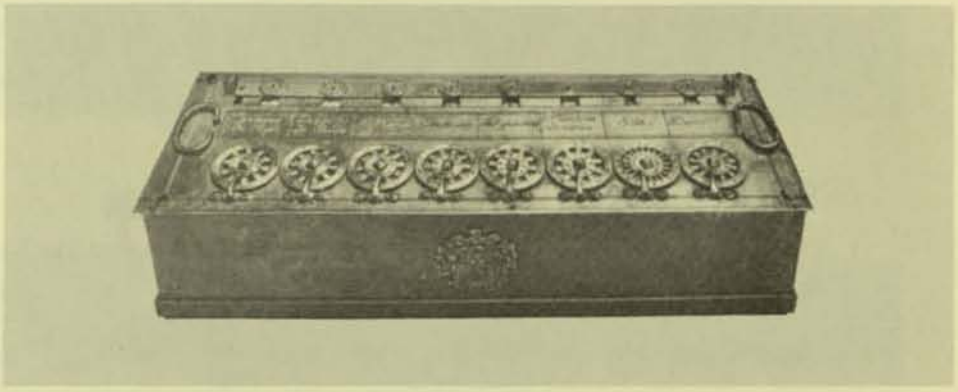
. . . or, when we write the digit "4," we are merely using a symbol which we humans recognize as representing four units, each having the value of one (1).



Over the years, literally dozens of devices have been invented as aids to performing arithmetic functions.

One common device used in many parts of the world is a unit called an ABACUS. This tool, which dates back many hundreds of years, is constructed using a series of beads strung in a special way on wires. Using these strung-beads, a skillful operator can rival the speed of many modern-day mechanical adding machines and calculators.

THE COUNTING WHEEL



Modern mechanical machines date back to the 17th century when many of the famous mathematicians of that time invented mechanical computing devices.

We won't go into a discussion of these machines. However, we should mention Pascal's Counting Wheel because it introduced basic principles which are still used in many modern-day machines.

Pascal used ten teeth on a wheel to represent the digits "0" through "9." This type of rotating disc is called a "decimal counting wheel."

With this device one could turn the respective wheels corresponding to the number of digits to be added . . . or subtracted . . . after which the sum (answer) would appear in the windows (dials) of the unit.

MECHANICAL ADDITION

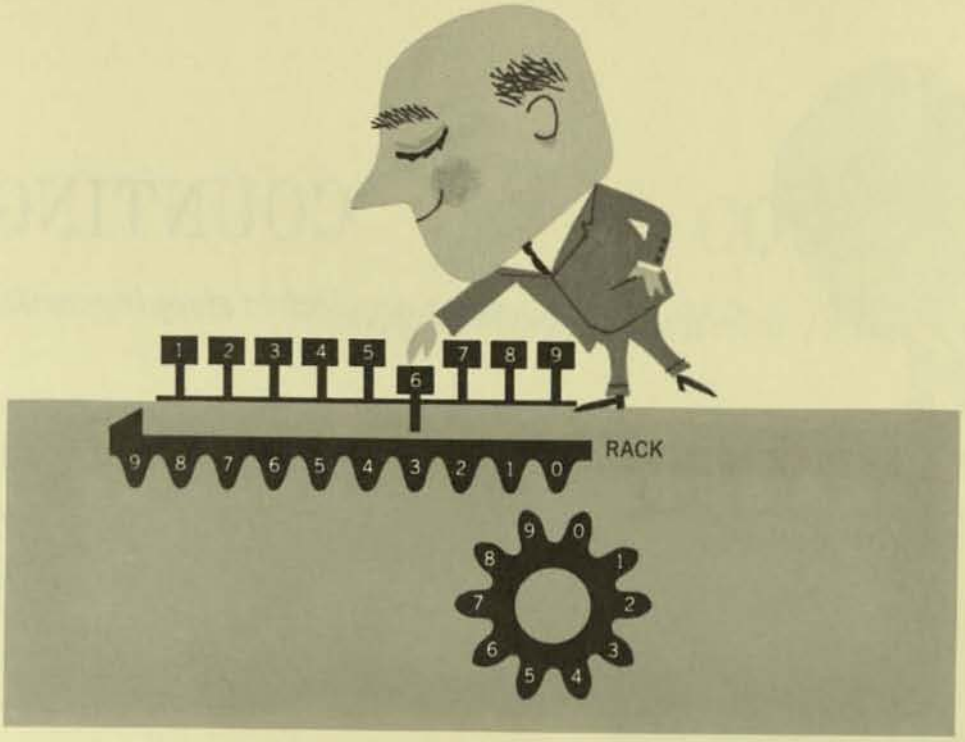
Assuming that the "6-key" has been depressed on the keyboard of the adding machine . . . and assuming that the Motor Bar has been depressed . . . here basically is what happens:

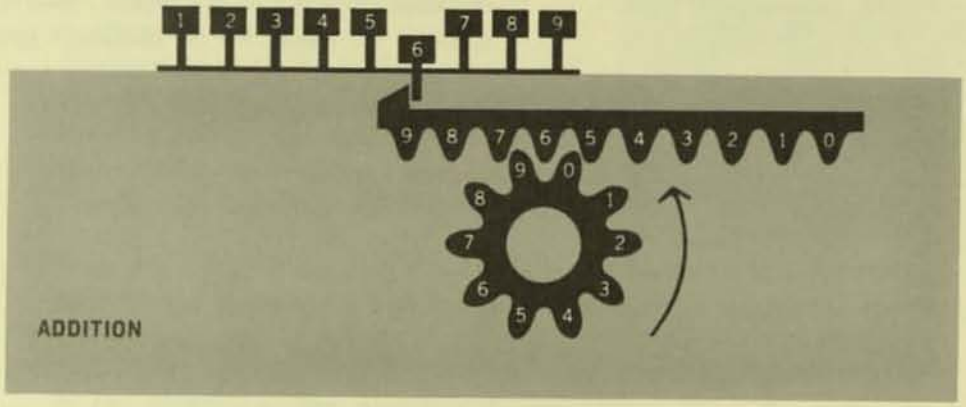
ADDITION

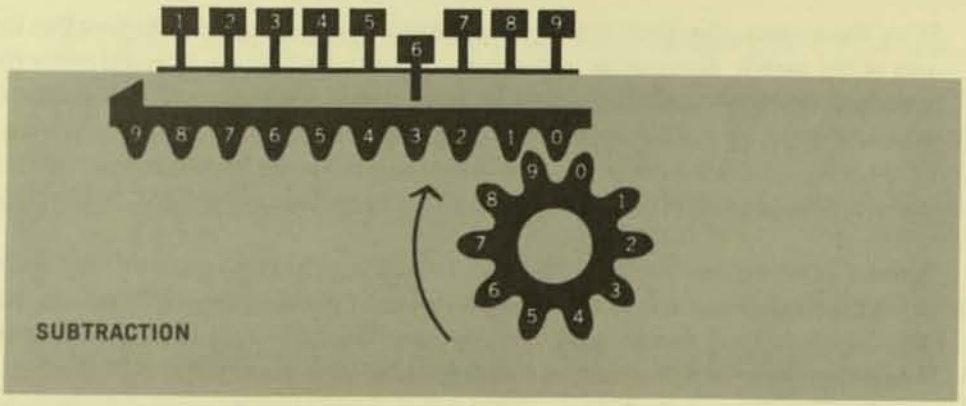
1. The rack moves to the right until it is halted by the key that has been depressed on the keyboard.
2. The rack and counting wheel then "mesh" . . . come together.
3. The rack moves back to its home position, thus it revolves the counting wheel the proper number of positions.

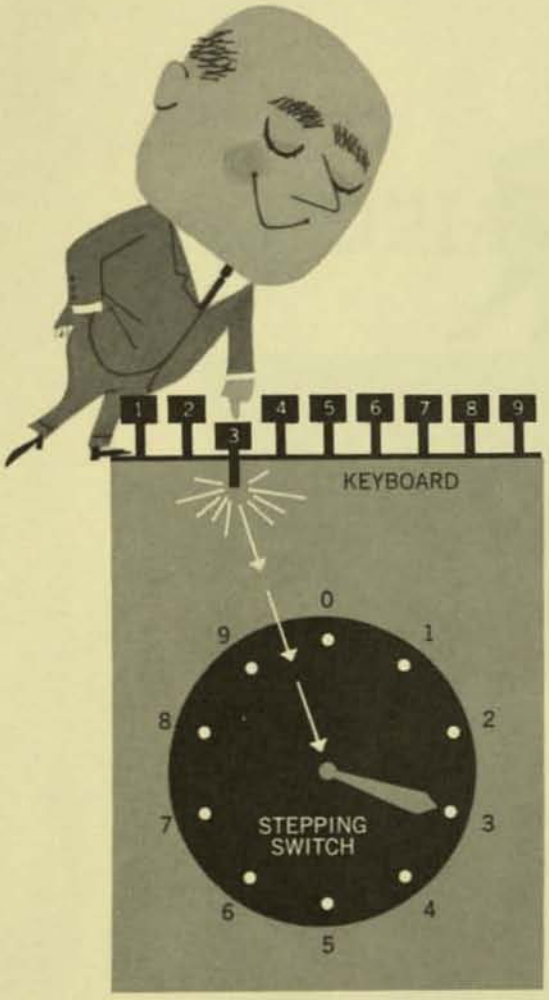
SUBTRACTION

1. Before the rack starts its forward movement, the rack and the counting wheel "mesh" . . . come together.
2. The rack moves to the limit setup by the key that has been depressed on the keyboard . . . thus the rack moves the wheel in the reverse direction.
3. The rack and the wheel separate.
4. The rack moves back to home position.


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


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


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



COUNTING

using an electric stepping-switch

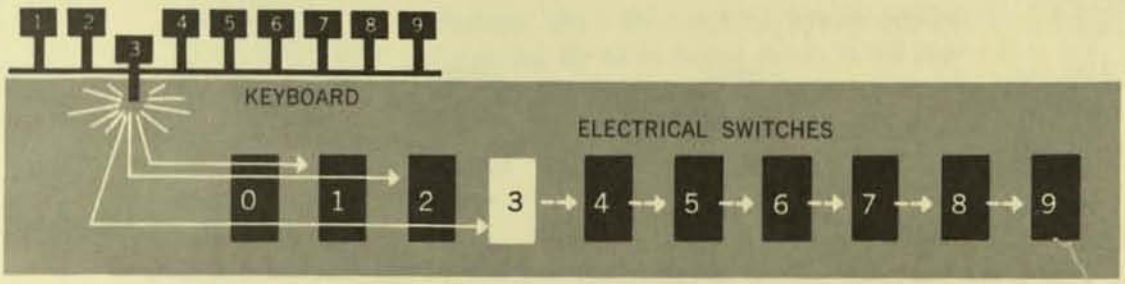
To help you make the transition from mechanical counting to electronic counting systems, we would like to challenge your imagination with this illustration of a hypothetical counting machine.

With this system, the 3-key would cause three pulses of electricity to energize the arm of the switch, thus moving the arm three positions. The 4-key would move the arm four positions . . . the 5-key would move it five positions, etc. In the case of subtraction the arm of the stepping switch would move in the opposite direction. In this way electricity could be used to perform the same functions that the rack and the counting wheel served in our previous discussion of mechanical machines.

As mentioned earlier, this illustration is intended merely to get you over the hurdle, from the mechanical to the electrical. As you study this series of brochures you will find increasing use of electrical impulses. While we will not bore you with detailed engineering drawings, we will, from time to time, interject those things which will enable you to better understand, and thus better utilize the wonderful data processing facilities engineers have made available to us.

COUNTING

using electrical switches (Flip-Flops)



Rather than employ a mechanical rack and a mechanical counting wheel, or a stepping switch with a moving arm, the arithmetical section of our hypothetical counting device could employ a series of electrical switches. These switches, sometimes called FLIP-FLOPS, operate in tandem, and can be turned ON and OFF as pulses of electricity are introduced.

With this system, if the digit three is depressed on the keyboard of the machine, three pulses of electricity will flow to the arithmetical section and the switches will be energized as follows:

1. The electrical current flows into the arithmetical unit . . . this means that if no pulses are introduced from the keyboard, the 0-switch will be in the ON position.

2. The first pulse of electricity for the digit three will turn OFF the 0-switch, and will cause the 1-switch to be turned ON.

3. The second pulse will turn OFF the 1-switch and will turn ON the 2-switch.

4. The third pulse will turn OFF the 2-switch and will cause the 3-switch to be turned ON.

From this brief description of how electrical switches could be used as counting devices, it is obvious that mechanical methods can be replaced by mechanisms that operate at the speed of electricity. In fact, it is interesting to note that the first all-electric computer, the ENIAC, built by the University of Pennsylvania in 1946, used the Flip-Flop system in its arithmetic section.

DECIMAL COUNTING SYSTEM

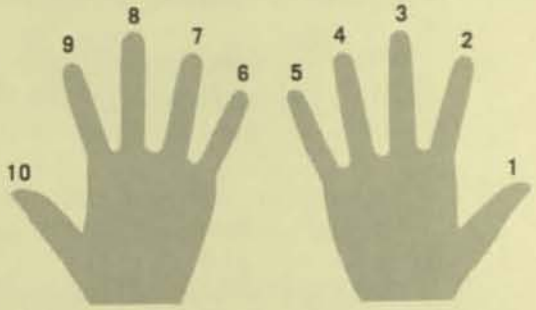
As you know, the decimal system is based on ten digits, "0" through "9."

1	,	2	6	4
THOUSANDS		HUNDREDS	TENS	UNITS

This is just one other way of representing one thousand two hundred and sixty four units . . . each carrying a value of one (1).

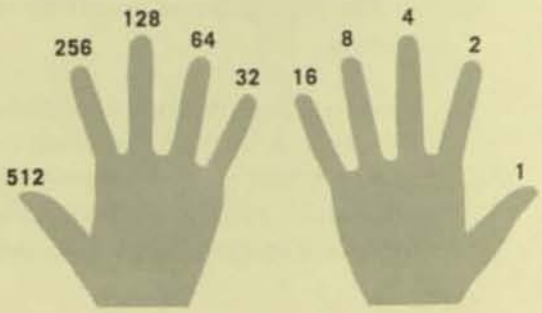
The lowest order position in the decimal system is called the UNITS, and is made up of ten possible positions, "0" through "9." The next position is called the TENS, and is also made up of ten positions, "0" through "9." And this continues on up with HUNDREDS, THOUSANDS, ETC.

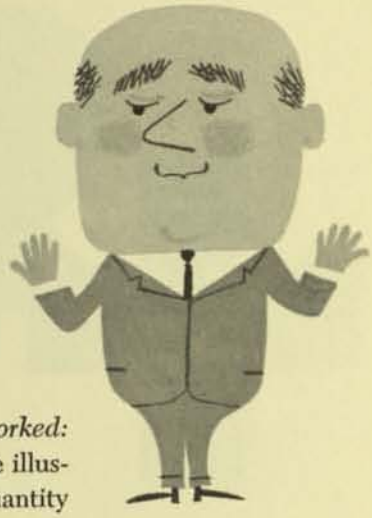
Now, let's go back to the hands. We find that even though young Pete had learned to use a pencil and paper, he still had the desire to occasionally use his fingers as an aid to counting. However, since he considered each finger to have the value of one, he could not conveniently count past the number "10."

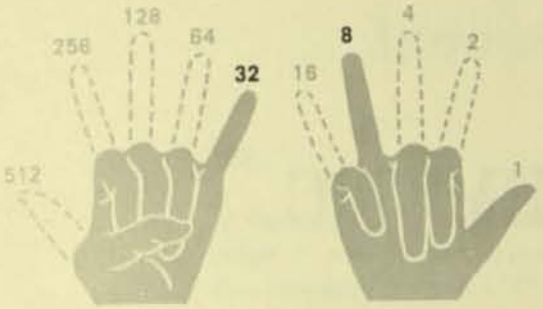


Being a stubborn young fellow . . . but ingenious, we might add . . . Pete experimented and came up with a scheme whereby he could hold up fingers to represent any quantity from "1" to "1,023."

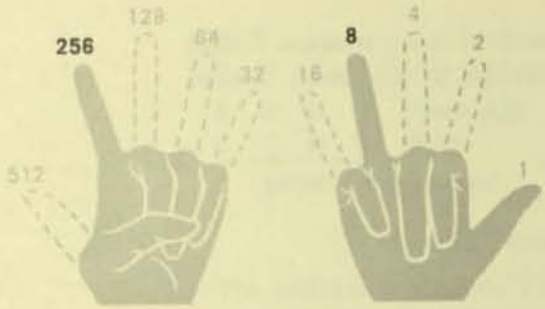
The first thing Pete did was to assign a different value to each of his ten fingers. The thumb on his right hand was given the value of "1." And, by successively doubling the value of the fingers as he progressed across the two hands, he determined that the thumb on the left hand would carry a value of "512."



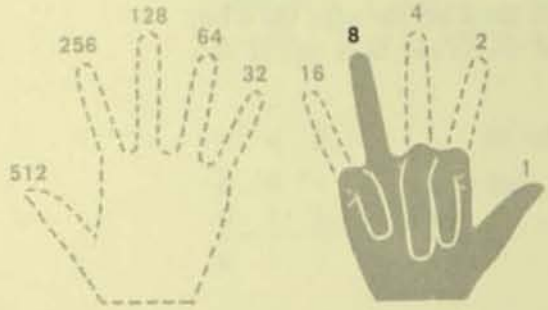




Basically, this is the way Pete's system worked: By raising the three fingers shown in the illustration, he was able to indicate the quantity "41" . . . (32 + 8 + 1 = 41).



Or, by raising the fingers shown in this illustration, he was able to indicate the quantity "265" . . . (256 + 8 + 1 = 265).



Or, by raising the two fingers shown in this illustration, he could indicate the quantity "9" . . . (8 + 1 = 9).

From this brief discussion, it can be seen that Pete materially increased the capacity of his hands to represent quantities in excess of the normal capacity of ten. Obviously, he did not discover a great new counting system which would cause people to go back to using their fingers as counting devices. However, with Pete's example of *just another way of representing numerical quantities* we would now like to discuss a counting system which many business-type computers employ—namely Binary Arithmetic. While we will not discuss all the different Binary Systems in this brochure, we will give you an in-sight to one of the more common systems—the Binary-Decimal System.

BINARY ARITHMETIC

With the Binary System only two conditions are permitted in any position. Unlike the decimal system which employs the digits "0" through "9," the Binary System employs only two digits, "0" and "1."

64	32	16	8	4	2	1	..... value of each position
0	**0**	**1**	**0**	**0**	**0**	**1**	equals **17**

The lowest order position in the binary system is called the 1-bit, and can have only two conditions, "0" or "1." The next position is called the 2-bit; the next, the 4-bit; the next, the 8-bit; the next 16-bit; etc.... each of which can have one of two conditions "0" or "1."

Here's another example:

64	32	16	8	4	2	1	..... value of each bit
1	**0**	**0**	**1**	**0**	**1**	**1**	equals **75**

In other words, this is *just another way of writing the quantity "75," sixty four ones, plus eight ones, plus two ones, plus one one, equals seventy five units (1's).*

Addition

Rule:

If the position (bit) containing a "1" has another "1" added to it, the count in that position reverts to "0," and a "1" is carried to the next bit position.

8	4	2	1	..... bit values	
0	0	/0	1	equals digit	"1"
0	0	0	1	PLUS	"1"
0	0	1	0	equals	2

This is just another way of writing the quantity "2."

The addition of the two "1's" in the 1-bit position produced a result of "0" with a carry of "1" to the 2-bit position. The carry into the second position results in an addition of "1" and "0" . . . the sum of which is "1." Since the result of the addition caused a "1" to be written in the 2-bit position, the sum is read as the quantity "2."

8	4	2	1	..... bit values	
0	/0	/1	1	equals digit	"3"
0	0	1	1	PLUS	"3"
0	1	1	0	equals	6

This is just another way of writing six "1's."

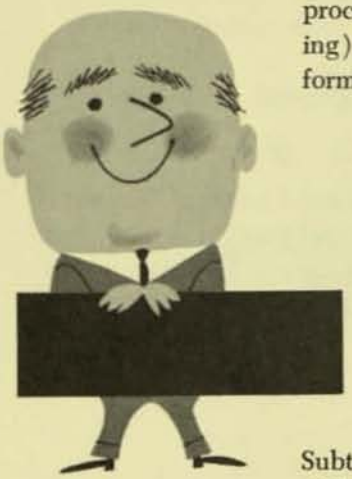
In this example, the addition of "1" in the 1-bit position equaled "0," with a carry of "1" to the 2-bit position. Since a "1" was carried into the 2-bit position this means that three "1's" must be added, the sum of which is "1" $(1+1=0+1=1)$ with a carry to the 4-bit position. The carry into the 4-bit position is then added to the "0" already there, resulting in a sum of "1," with no further carry. Thus, the "1's" written in the 4- and 2-bit positions are read as the quantity "6."

NOTE: (/) *indicates that a carry was made from the previous bit position.*

Subtraction

One of the first things taught in grammar school was that subtraction is the reverse of addition. You learned also that when the digit being subtracted is larger than the digit from which it is being subtracted, you must "borrow" a quantity from one of the next higher positions.

Under the Decimal Method illustrated on the opposite page, it was necessary to borrow the "1" in the thousand dollar position. While you have learned to mentally short-cut this process, you could have written the quantity (after borrowing) as nine 100's, nine 10's, and ten 1's. Then when performing the subtraction you would obtain the answer 999.



Subtraction under the Binary System follows the same rules of "take-away and borrow" used in the Decimal System. In the example of binary subtraction, we could not subtract "1" from "0" in the first bit position. So, we moved to the left searching for something to borrow . . . and, the first quantity we could borrow was the "1" in the 4-bit position.

Since we were borrowing the quantity "4," we distributed that quantity over the remaining low-order positions. To do this we placed a "1" in the 2-bit position and two "1's" in the 1-bit position. If you add this distribution, you will find that we still had the quantity "4" represented . . . but, now it was in a condition whereby we could perform the subtraction.

"In subtraction, we take-away and borrow."

DECIMAL METHOD

```
1, 0 0 0          9 9 10
        1 —   or        1 —
-------           -------
  9 9 9             9 9 9
```

BINARY SYSTEM

```
8 4 2 1    .... bit values ....   8 4 2 1
                                      1
0 1 0 0    equals   4             0 0 1 1   ("2" plus two "1's" equals "4")
0 0 0 1 —           1 —           0 0 0 1 —
---------           ---           -------
0 0 1 1             3 .......     0 0 1 1
```

We might look at the above problem another way . . .

The "1" written in the 4-bit position is in reality just another way of writing four "1's." So, by subtracting "1" from four "1's" we obtain an obvious result of "3." And "3" is written binarily as a "1" in the 2-bit position and a "1" in the 1-bit position.

Similarily, under the decimal system, when we write the quantity one thousand as "1,000," this is just another way of writing one thousand units of "1." Then when we subtract one of these units, we write the answer as "999" . . . and this is just another way of representing nine hundred and ninety nine units or "1's."

Multiplication

Most of us learned our multiplication tables early in life. Consequently, when we perform multiplication operations, it is partly from a set of answers stored in the "memory" portion of our brain, and partly as a process of addition.

In the final analysis, multiplication is nothing more than repeated addition. A calculator, for example, when multiplying five times five, will turn a counting wheel five positions, then five more positions, then five more positions, then five more positions, then five more positions . . . thus the result of 25 will be obtained.

$$\begin{array}{r} 5 \\ \times 5 \\ \hline 25 \end{array} \quad \text{or} \quad \begin{array}{r} 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ \hline 25 \end{array}$$

From this we can see that systems based on the decimals "0" through "9" are cumbersome. They require much memory work, elaborate counting wheels, or many repeated additions.

BINARY MULTIPLICATION

Rule: $1 \times 1 = 1$ . . . all other conditions equal "0."

	16	8	4	2	1	..... bit values		
	0	0	1	0	1	equals	5	
	0	0	1	0	1	times	5	
	0	0	1	0	1	equals	5	(First operation)
0	0	1	0	1		plus	20	(Second operation)
	1	1	0	0	1	equals	25	(16 + 8 + 1 = 25)

In the first operation, "1" times "5" equals "5."

In the second operation, "4" times "5" equals "20."

Then, obeying the rules of addition, the sum of the two multiplications equals "25."

Division

In our discussion of multiplication we stated that multiplication is nothing more than repeated addition.

Similarly, division is repeated subtraction. To divide "5" into "20" means: how many times can "5" be subtracted from the quantity "20"?

To divide binarily, we conform to the same rules we have always followed when dividing decimals into decimals. However, we must, at the same time, remember the things we have learned about binary subtraction:

```
4 2 1   16 8 4 2 1   ..... bit values

              1 0 0             4
1 0 1   | 1 0 1 0 0          5 | 20
          1 0 1                  20


4 2 1   16 8 4 2 1   ..... bit values

            1 0 0 1             9
0 1 1   | 1 1 1 0 0          3 | 28
          1 1                    27
              1 0 0               1
              0 1 1
                  1   (4 - 3 = 1)
```

Subtraction

by complementary addition

Most of us have little difficulty subtracting one figure from another using the conventional "take-away and borrow" method. However, many people find the *complementary system* to be difficult and confusing.

For example:

	16.00	
Always ignore the last carry.....	**999,984.00**	Did you know that this is the complement of 16.00? ...Why?...
	(1)000,000.00	Because when you add it to 16.00 the answer is zero.

54.00	**54.00**	Did you know that adding the complement of one amount to another amount accomplishes the same result as subtraction.
16.00 –	**999,984.00**	
38.00	**38.00**	

Here's the formula I use:

16.00

999,984 ← • Write a digit which when added to the first low-order position will add-up to "10."

• Then write digits which when added to the other positions will each add-up to "9."

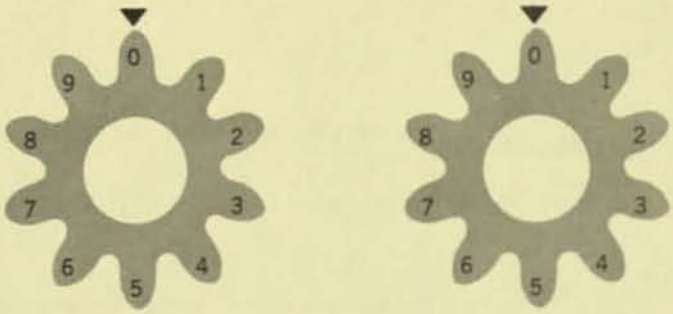
If you were writing the complement of 106.98 on the keyboard of an adding machine that had eight rows of amount keys you would write it as follows:

106.98 ..... this is the amount to be subtracted.

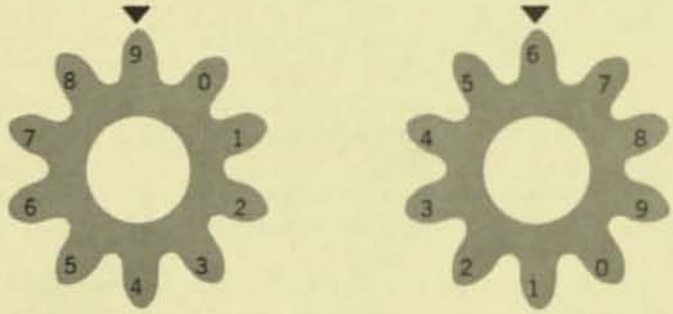
999,893.02 ..... this is the complementary amount you would use.

8	plus	**2**	equals	**10**
9	plus	**0**	equals	**9**
6	plus	**3**	equals	**9**
0	plus	**9**	equals	**9**
1	plus	**8**	equals	**9**
0	plus	**9**	equals	**9**
0	plus	**9**	equals	**9**
0	plus	**9**	equals	**9**

At this point, you have every right to challenge the *complementary system* as one that is cumbersome. However, cumbersome or not from the human point of view, most machines on the market today have the ability to perform certain complementary operations.



This is a set of counting wheels before an amount is added or subtracted.



This is the same set of counting wheels after the quantity "4" was subtracted.

Since you would not normally want a Credit Balance to print as a complementary amount most mechanical machines can convert complementary totals to positive amounts . . . this is accomplished by subtracting the complement from a string of zeros.

```
0 0 0, 0 0 0. 0 0
9 9 9, 9 9 9. 9 6-
---------------
        . 0 4 CR
```

Binary System

addition compatible with decimal system

Now that you have been exposed to the basic rudiments of Binary Arithmetic, and to the principles of Complementary Addition, it should be a relatively simple matter for you to obtain a basic understanding of the Binary-Decimal System.

8 4 2 1 . . . Bit Values

0 0 0 1	equals	"1"
0 0 1 0	equals	"2"
0 0 1 1	equals	"3"
0 1 0 0	equals	"4"
0 1 0 1	equals	"5"
0 1 1 0	equals	"6"
0 1 1 1	equals	"7"
1 0 0 0	equals	"8"
1 0 0 1	equals	"9"
0 0 0 0	equals	"0"

This chart illustrates the fact that only four binary positions (bits) are needed to represent the decimal digits "0" through "9." Consequently, we can establish the following format to represent UNITS, TENS, HUNDREDS, THOUSANDS, ETC.

Thousands	Hundreds	Tens	Units	
8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	. . . Bit Values

Using this Binary-Decimal format, then, we can apply the rules of Binary Addition to arrive at any sum:

Thousands	Hundreds	Tens	Units	
8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1	. . . Bit Values
1 0 0 1	0 1 1 0	1 0 0 1	0 0 0 1	9,691
0 0 0 0	0 0 1 0	0 0 0 0	0 0 0 1 +	201
1 0 0 1	1 0 0 0	1 0 0 1	0 0 1 0	9,892

It's as simple as this:

1 + 1 = 0 with a carry
1 + 0 = 1 with no carry
0 + 0 = 0 and, of course, no carry.

One problem arises, however, with the "Binary-Decimal System." Note that each group of binary values (8 4 2 1) provides the ability to accumulate up to the quantity "15."

8 4 2 1 . . . bit values
1 1 1 1 equals 15

Obviously, this is six more than is required to represent the digits "0" through "9." Consequently, any time the quantity in any group (units, tens, hundreds, etc.) exceeds "9," it is necessary that "6" be added to the respective group:

Tens	Units		
8 4 2 1	8 4 2 1 . . .	bit values	
0 0 0 0	1 0 0 1	equals 9	
0 0 0 0	0 1 0 0	plus 4	
0 0 0 /0	/1 1 0 1	equals 13	. . . at this point, the quantity in the units group is in excess of "9" . . . so we must add "6" to the
0 0 0 0	0 1 1 0	. . . . .	. . . units group.
0 0 0 1	0 0 1 1	. . . . .	. . . now the quantity "13" is properly written.

There is one other time when the factor "6" must be added:

Tens	Units		
8 4 2 1	8 4 2 1 . . .	bit values	
0 0 0 /0	1 0 0 0	equals 8	
0 0 0 0	1 0 0 0	plus 8	
0 0 0 1	0 0 0 0	. . . . .	. . . Obviously, this answer should be "16." So, when a transfer is made from one group to another, the quantity "6" must be added.
0 0 0 0	0 1 1 0		
0 0 0 1	0 1 1 0	equals 16	

We should point out at this time that the adjustment factor (6) is used only when manually solving "Binary-Decimal" problems . . . in a computer, this is taken care of automatically.

Binary System

subtraction compatible with decimal system

Using the rules for subtraction introduced earlier in this brochure, here's how the quantity "6" would be subtracted from the quantity "69" using the Binary-Decimal System:

Tens 8 4 2 1	Units 8 4 2 1	... bit values	
0 1 1 0	1 0 0 1	equals	69
1 0 0 1	0 1 0 0	complement of	6
/1 /1 1 1	/1 1 0 1		
0 1 1 0	0 1 1 0		
0 1 1 0	0 0 1 1	equals	63

At this point, the quantities in the units and tens groups are each in excess of "9." So we added "6" to each group.

Ignore the last carry

Thus, in subtracting "6" from "69" the complement of "6" was added to "69" to arrive at the sum of "63."

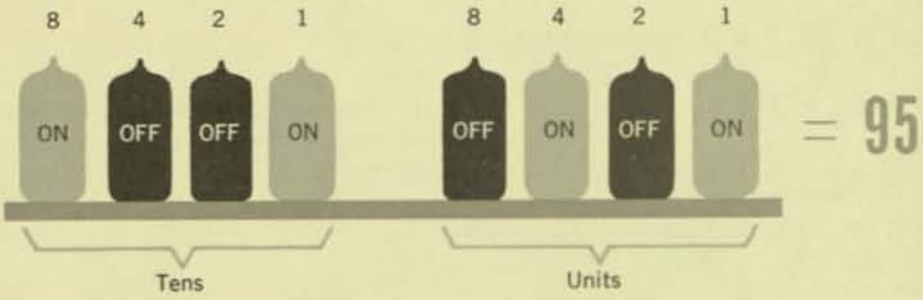
SUMMING UP BINARY ARITHMETIC...

Obviously, this method of using "0's" and "1's" to represent all numerical quantities is not the most efficient, from a pencil and paper point of view. However, in a computer, numbers are usually represented by ON or OFF conditions of tubes or transistors... or MAGNETIZED or DE-MAGNETIZED conditions of magnetic materials. Since these ON and OFF, MAGNETIZED or DE-MAGNETIZED CONDITIONS can be used to represent "1's" or "0's," Binary Arithmetic becomes extremely practical for computer systems.

Let's build

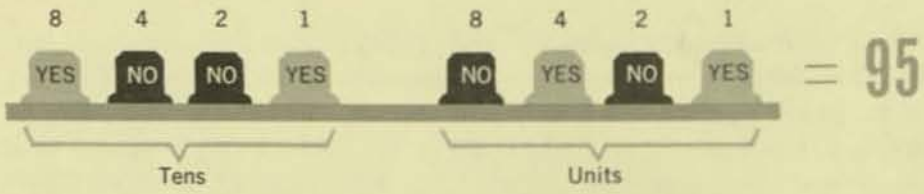
A COMPUTER MEMORY

In doing this, we could use ordinary vacuum tubes . . . the type found in your home radio . . . as data storage devices.



In the above illustration, if we assume that each tube that is ON represents a binary "1," and each tube that is OFF represents "0," we can see how the quantity "95" could be represented electronically.

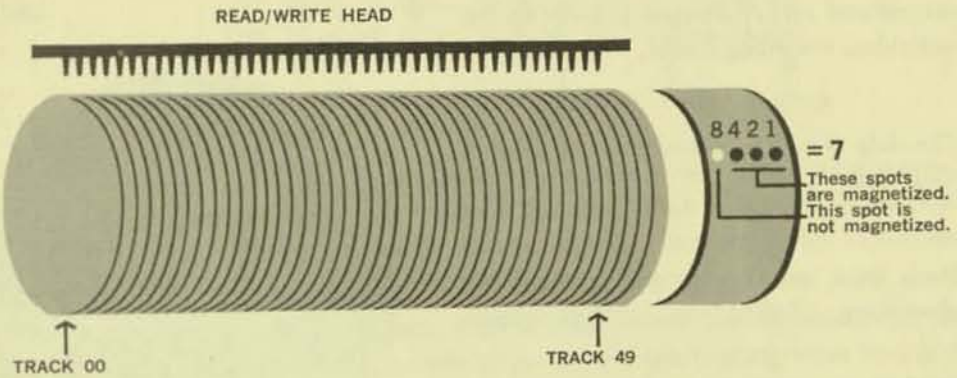
Since tubes are normally rather large and because they generate considerable heat, we might use Transistors instead. Transistors are normally quite small and give off very little heat. Thus they enable more compact design, and to a degree, enable engineers to build a more reliable system.



In the above illustration, all the YES transistors can be considered as holding a certain amount of electricity . . . and the NO transistors as holding no electricity. If we consider the YES's as "1's" and the NO's as "0," we can see how transistors could be used to represent the quantity 95 electronically.

Drum-type memory

Rather than employ tubes or transistors as the memory components of our computer, we might ask our engineering department to design a revolving drum . . . the outside surface of which would be coated with a special magnetic material.



With our especially designed drum, data would be stored in the form of magnetized spots, arranged binarily on individual recording tracks.

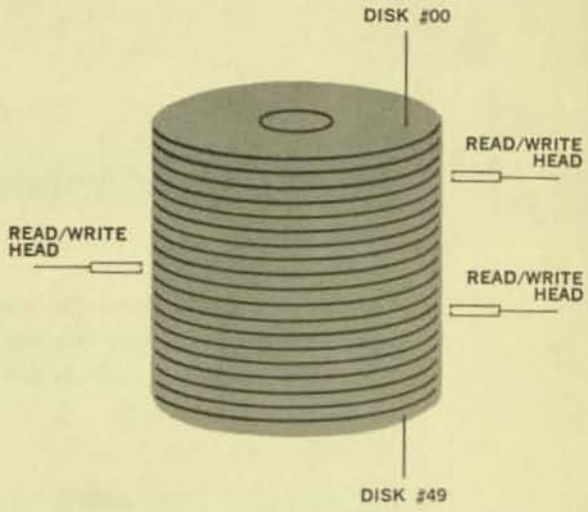
Of course, we would need a motor to rotate the memory-drum . . . and because we are talking about high-speed data processing, we would need to specify that the drum must rotate several thousand times per minute.

Then, too, in order to read and write the data, we would need to design a set of special instruments called read/write heads. Since we would probably want to organize the recording area into addressable locations, we would specify further that a read-write head be assigned to each recording track. In this way, our computer would be able to assign data to . . . or read data from . . . indexed locations on the drum.

In effect, our high-speed magnetic drum would act as an electronic filing cabinet. Unlike conventional filing cabinets however, our drum would have the ability to store large amounts of information in a relatively small space . . . and that information could be placed into the file, or read out of the file automatically – at electronic speeds.

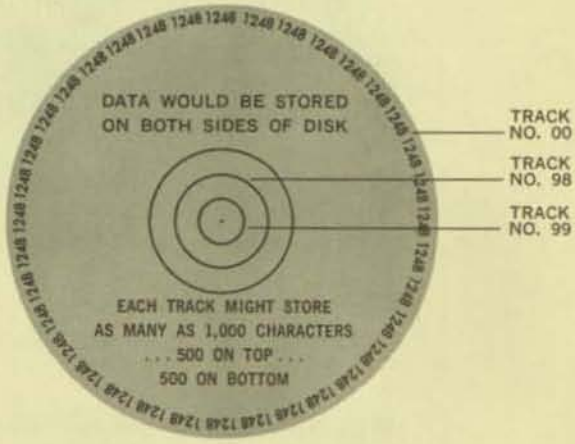
Disk-type memory

Rather than employ tubes, transistors, or drums, we could design the memory of our computer to utilize a series of magnetic disks. These disks might look very much like over-size phonograph records, stacked one on top of another.



Like the drum, each disk would be coated with a special magnetic material, thus enabling data to be stored in the form of magnetized spots arranged binarily on the individual recording tracks.

The data would be stored on both sides of each disk.



Each track would store as many as 1,000 characters . . . 500 on top . . . 500 on bottom.

Special read/write heads would be used to record and read data . . . Disks would rotate at over 1,000 revolutions per minute.

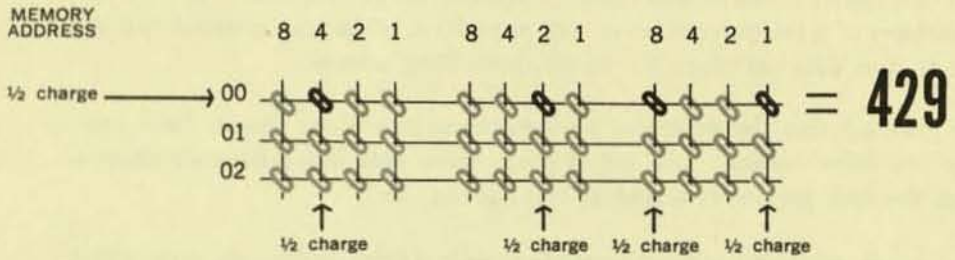
Since the disks in our computer will be rotating at high speeds . . . and since the read/write heads will be able to select the desired track on any disk . . . data will be quickly "accessed" (located)—and will be processed on a random basis.

The term RANDOM PROCESSING can be compared with the manner we humans post to conventional ledger records. Rather than handle every account in a file, we normally by-pass all inactive accounts in a ledger and go directly to the active record. Similarly, random devices, such as disks and drums, are able to by-pass large amounts of data and thus go to the desired records in a relatively short time.

Magnetic core memory

To build the memory for our computer, we might suggest to our engineers that they employ small donut-shaped pieces of "ferrous oxide" material no larger than the lead end of a pencil.

The following illustration shows how groups of tiny cores . . . which can be magnetized and de-magnetized millions of times each second . . . can be strung on wires to form a memory plane for a computer system:



A full charge of electricity is required to magnetize a core . . . the point where a full charge (two ½ charges) is centered will cause that core to be magnetized. Also, the act of magnetizing one or more cores in a given line, called an address, causes all other cores in that address to be de-magnetized.

Because there are no moving parts; such as drums, discs, reading heads, etc., and because data is read or stored merely by routing electricity along wires to or from locations in memory, this type of memory is referred to as a "direct-access" type of memory.

A parallel can be drawn by comparing a Direct-Access type of memory with an electric light. In your home, when you turn on a switch, the current flows along a set of wires to the proper electric light . . . thus, through the medium of a switch you have direct access to a given destination, or address of a light.

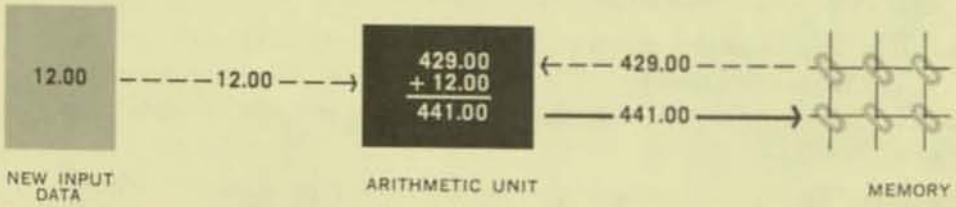
There is one significant difference that should be noted between an electric light and electricity stored in a computer, whether it employs disks, drums, or magnetic cores. When you turn an electric light switch to the OFF position, the light goes off. This is not true of computer addresses, they remain energized even though the current has been turned OFF. Since memory addresses have the ability to store electricity, they can be used to transmit data (in the form of electrical impulses) back to the control section of a computer. In this way, then, computers have the ability to store data and to retrieve data merely by routing electricity over electrical wires, or printed circuits.

In conclusion

You have now been exposed to the subject of Binary Arithmetic, and to some of the ways data can be stored by a computer. It is important to remember, however, that the memory of a computer is never used to perform arithmetic operations—it is a place to store data—in effect, it is an electronic filing cabinet.

Then, too, whether the computer employs tubes, transistors, drums, discs, cores . . . or any other facility . . . the act of placing new data into a memory location, erases the data previously stored in that address.

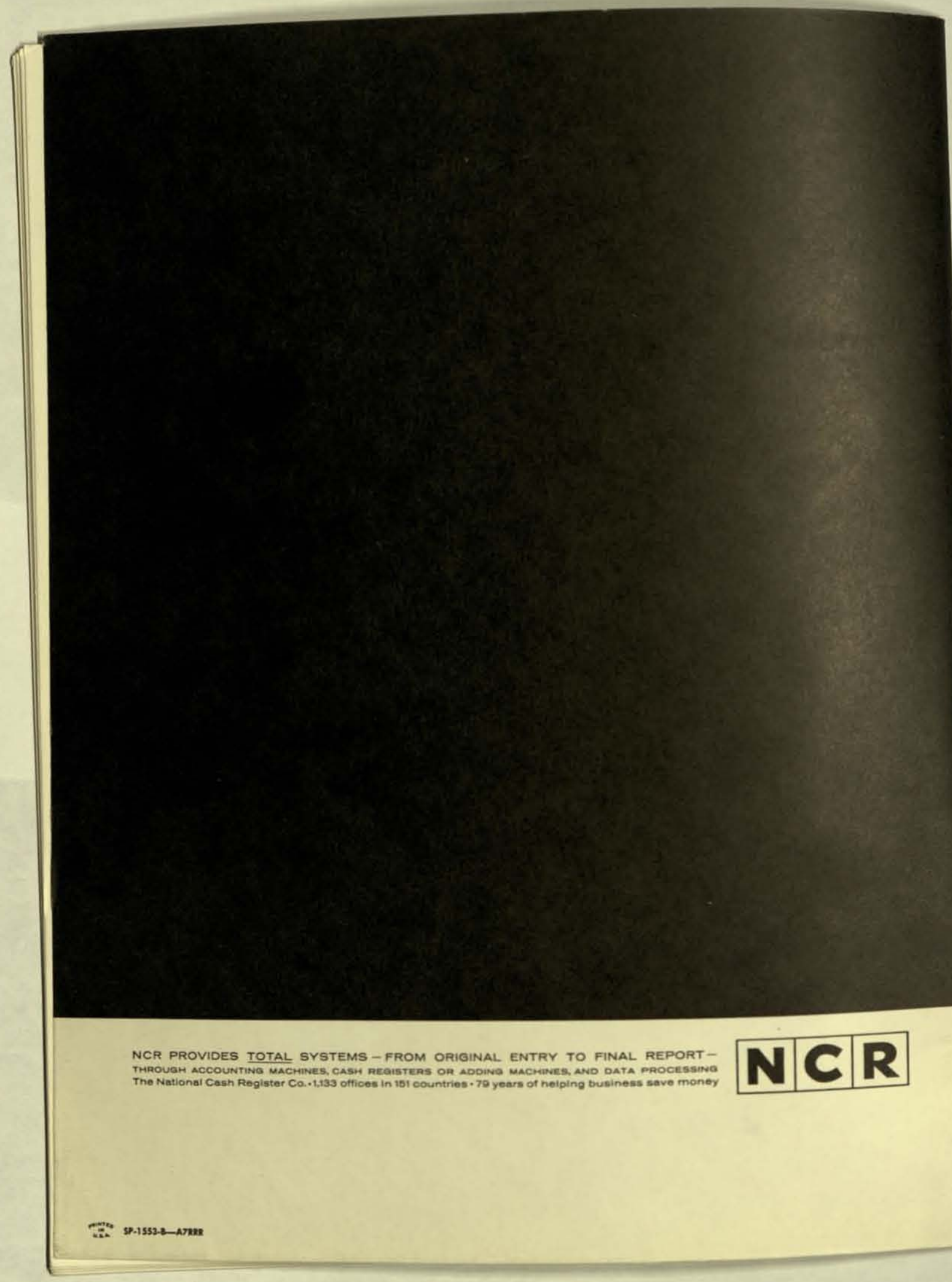
For example, when one or more magnetic cores in a given address are magnetized, all other cores in that same address are de-magnetized. Thus, new data is never added to stored data directly:



Stored data is normally read into an arithmetic unit where it is then added to the new data . . . the sum of which can then be placed back in memory.

As you read the remaining booklets in this series, the things discussed in this brochure will be expanded and added to. However, we hope we have removed at least a portion of the mystery from the subject of Electronic Data Processing.

The next book in this series—What is a Computer?—*takes up the subject of EDP machines, from the layman's point of view.*



NCR PROVIDES TOTAL SYSTEMS – FROM ORIGINAL ENTRY TO FINAL REPORT –
THROUGH ACCOUNTING MACHINES, CASH REGISTERS OR ADDING MACHINES, AND DATA PROCESSING
The National Cash Register Co. • 1,133 offices in 151 countries • 79 years of helping business save money

NCR

PRINTED IN U.S.A. SP-1553-B—A7RRR