

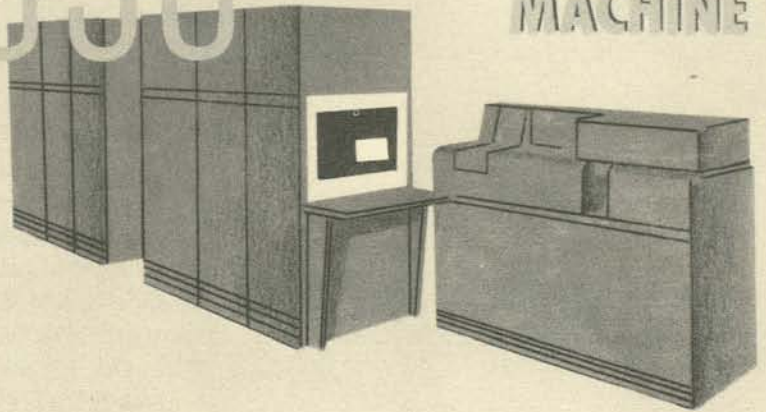
I B M[®]

P R E S E N T S

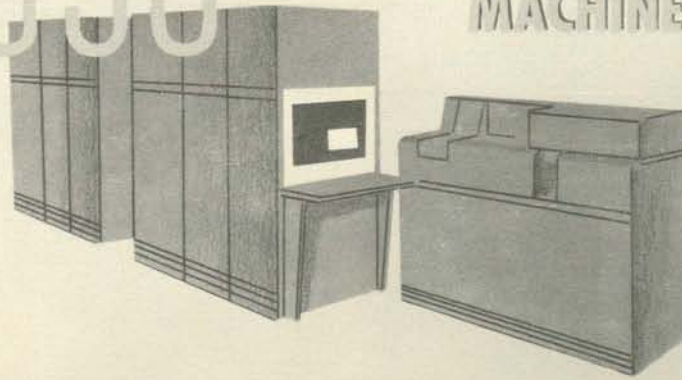
THE

650

MAGNETIC DRUM
DATA PROCESSING
MACHINE



THE 650 MAGNETIC DRUM DATA PROCESSING MACHINE



2

The IBM 650 is a moderate-sized data processing machine designed to solve the various problems encountered in the fields of accounting, engineering, mathematics and research.

The 650 with the Read-Punch feature consists of three units: The Read-Punch, the Console and the Power Unit.

On the right is the 533 Read-Punch Unit. It contains the means of introducing data into and withdrawing results from the 650 in punched card form. In the center is the Console Unit – the 650 itself. It contains the magnetic drum from which the machine derives its name and the arithmetical devices used to perform all calculating operations. The Operator's Console is located on the front of this unit. On the left is the 655 Power Unit. In addition to the power supply, it houses the translating circuitry that converts data from the punched hole language of the card to the electronic language of the 650.

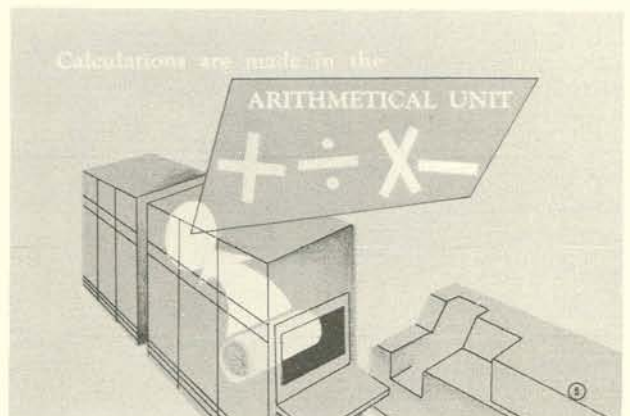
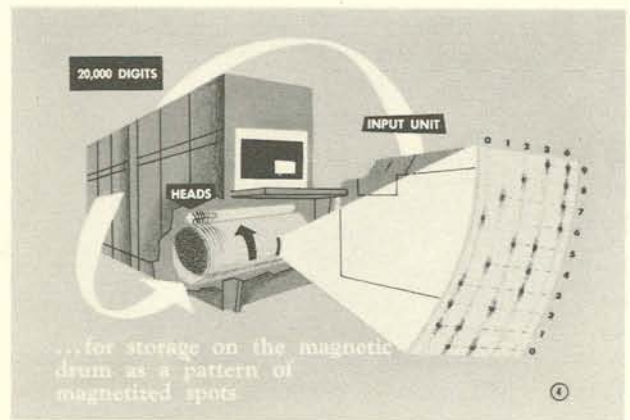
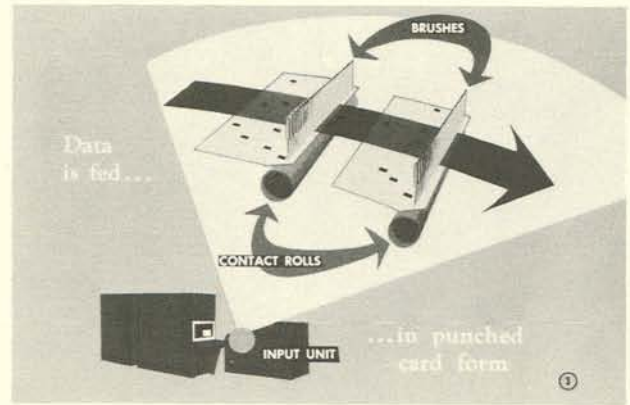
Input data, in punched card form, moves past two reading stations. Each station has 80 individual brushes—one for each column of the card. As the cards pass the reading stations, they move beneath these brushes. The brushes, in turn, “sense” the presence of and assign values to any holes punched in the 80 columns of the cards. Data to be processed is usually read at the second station.

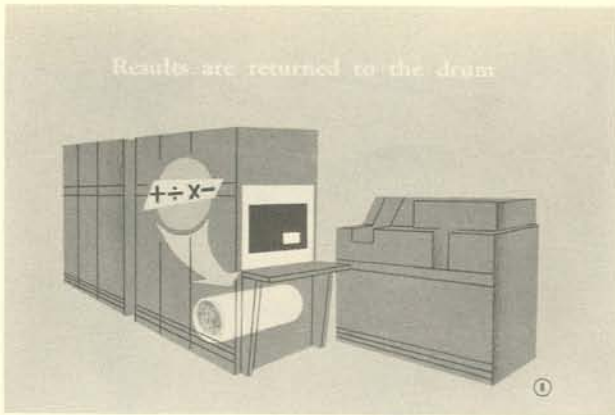
The first reading station is featured because of its contribution to the flexibility of the 650. One of its uses arises when each set of input data is in a group of cards and calculation must be restrained until the entire group is read. In this case, data to be processed is read from the second reading station. As each card passes second reading, the first reading station senses the card that follows it, so that the two can be compared for belonging to the same group. As long as the card passing the first station and the one passing the second station belong to the same group, feeding is continued, but as soon as the one passing second belongs to one group and the one passing first to another, feeding is stopped and calculation is started.

The first reading station also permits each card to be “pre-sensed” before data is read at second. This, in turn, permits intermixing the types of cards being processed, wherein the kind of data, and its location, changes from card to card. In this case, first reading “tells” second—ahead of time—what to read and where to read it.

The data from the cards, changed into electrical impulses by the input unit, is stored on the surface of the revolving drum as a pattern of magnetized spots. The pattern, one for each digit, is deposited by recording “heads” arranged in groups of five along the surface of the drum. Each group operates on the drum as a unit, with the third and fifth heads of a group, for example, combining to produce an eight, and the second and fifth to produce a seven, and so on.

After all of the data associated with a problem has been stored, it is transferred, as needed, from the drum to the arithmetical unit. This single unit, which operates at electronic speed, adds and subtracts to produce a twenty-digit total, multiplies to produce a twenty-digit product and divides to produce a ten-digit quotient.

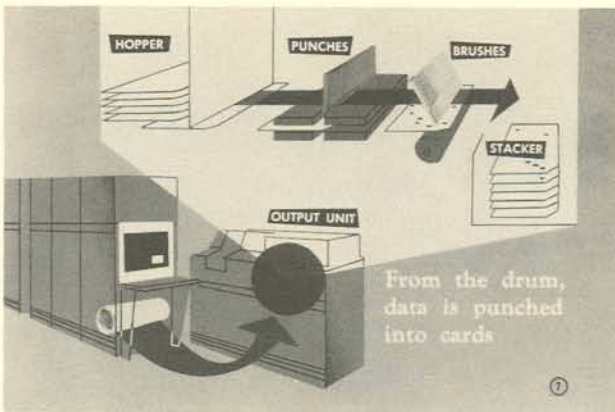




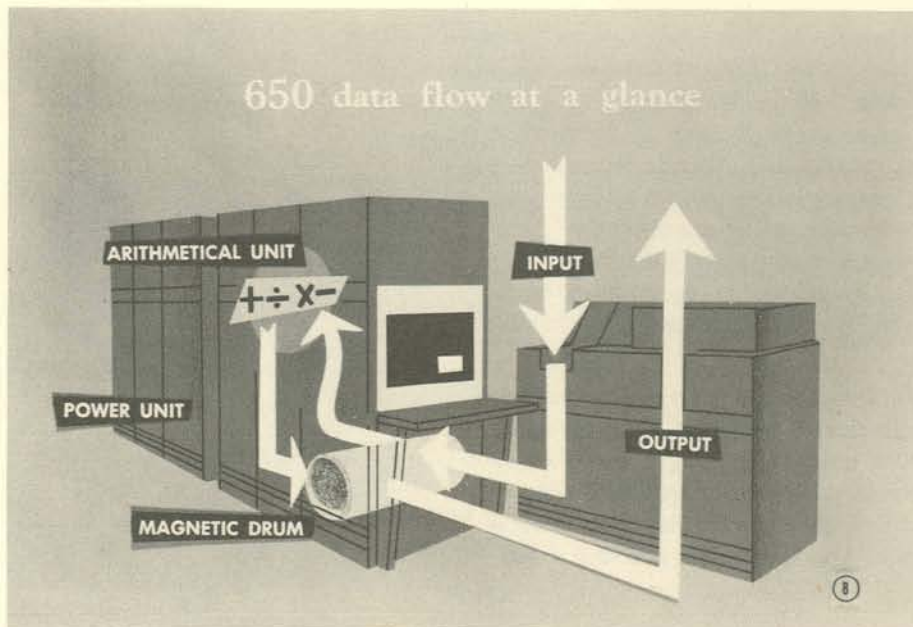
Calculated results are usually transferred back to the drum for storage. They are held there until the problem is completed. Data can be transferred back and forth within the 650—in and out of the arithmetical unit, on and off of the drum—as often as is dictated by the requirements of the problem, for once a factor has been “memorized” by the drum it is always available. This is because each factor of data is permanently magnetized on the drum until a new factor is “read in” to take its place.

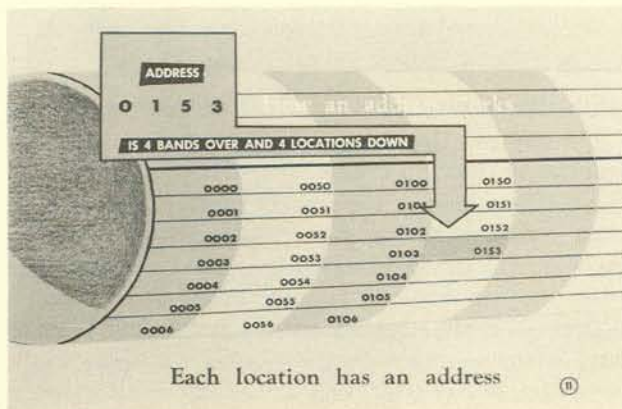
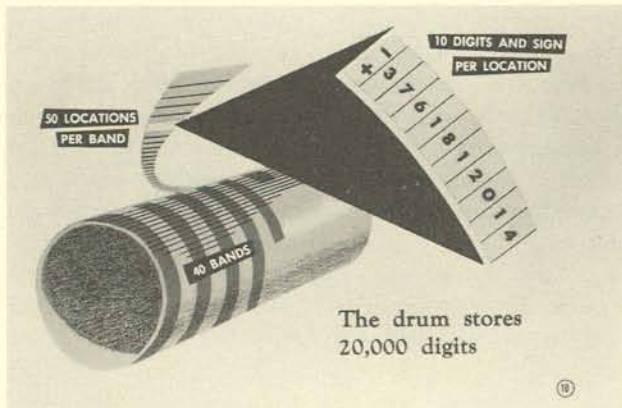
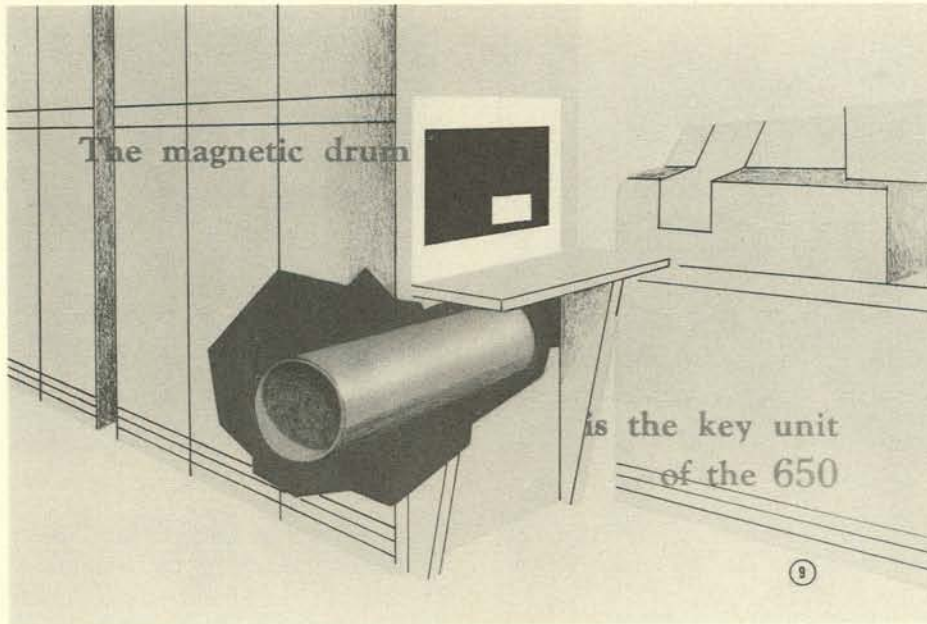
The transfer back and forth of the data is done with the same recording devices that are arranged along the drum. Not only do they write—they also read.

After all calculations have been completed, the results are transferred from the drum to the output unit, where they are punched into cards. Output cards move past two stations, passing first beneath 80 punches which punch the cards and then beneath 80 brushes which read the cards. The reading station permits the operation of the punches to be checked. In addition, it permits repetitive data, such as dates, to be “read back” from a card passing the brushes and punched into a card passing the punches. This eliminates the necessity of needlessly processing such repetitive data through the machine.



Data moves from the input unit, to the drum, to the arithmetical unit, back to the drum and thence to the output unit. The arithmetical unit performs all calculations. It is also used to transfer data from one place on the drum to another.



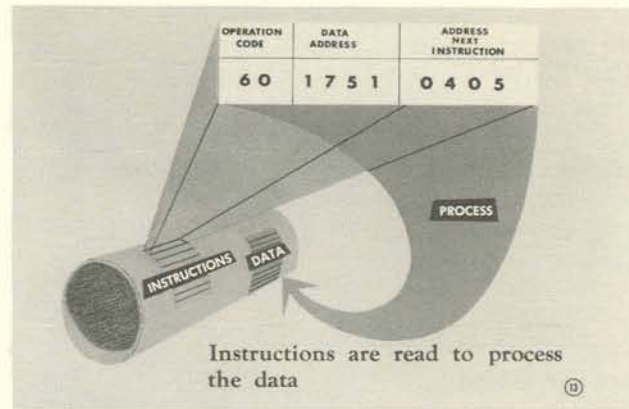
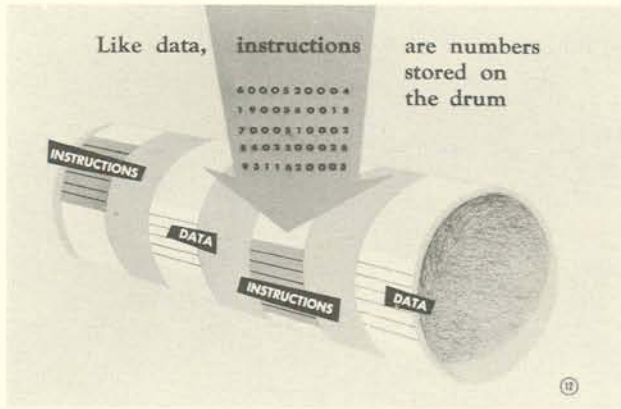


All data being processed by the 650, from the time it enters at the input until it leaves at the output, circulates on and off the magnetic drum. For this reason, the drum is truly the key unit of the 650.

The magnetic drum is a nickel-cobalt coated metal cylinder, four inches in diameter and sixteen inches long, that rotates at 12,500 revolutions per minute. It can be thought of as being made up of individual slices or bands along the drum. The bands, in turn, can be thought of as being divided into individual sections or locations around the drum. A location accommodates ten digits and a sign; there are fifty in each band. Bands and locations are not actually physical subdivisions of the drum—they are areas defined by the operation of the recording heads. The forty bands are the forty slices of the drum that rotate past the forty groups of recording heads. Locations are the fifty areas of the bands that the recording heads operate upon.

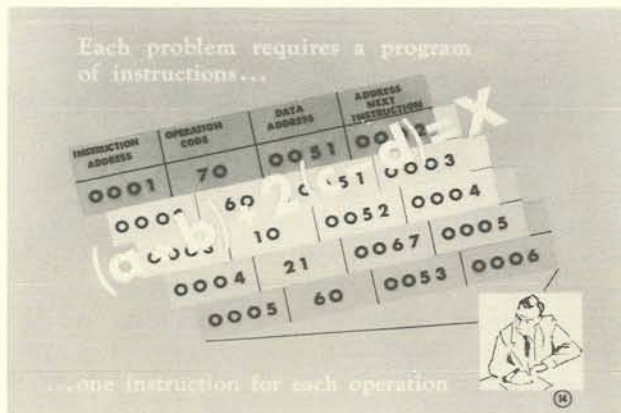
As it processes each problem, the 650 must find those specific locations out of the 2,000 that are being used. It finds them through an internal electronic selecting system or address network.

Addresses are four-digit code numbers that permanently identify the position of each location on the drum. They are not written on the drum. They cause a location to be "found" by "activating" one of the forty groups of recording heads (to single out the band containing the location) during the precise instant that the drum's rotation is moving the specified area beneath the heads (to single out the location). To find a location, the 650 is given its address. This is done with an instruction.

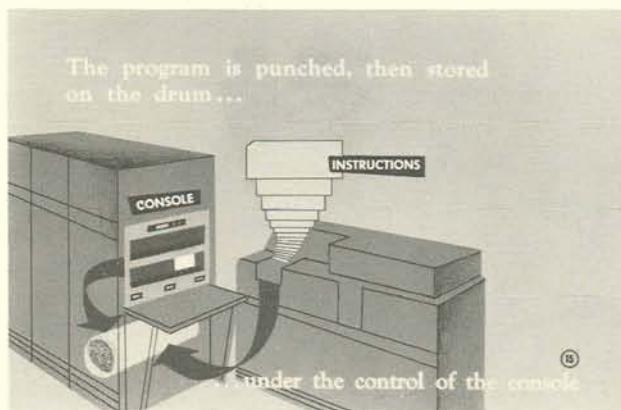


Instructions are ten-digit numbers and, like data, are stored on the drum in individual locations; they are used by the machine to process data.

Each instruction tells the machine the operation it is to perform, where on the drum it is to find the data involved in the operation and where on the drum it is to find the next instruction to be carried out. The instruction illustrated tells the machine to add the data from location 1751 and to find the next instruction in location 0405. As one instruction is being executed, the 650 begins finding the next one.

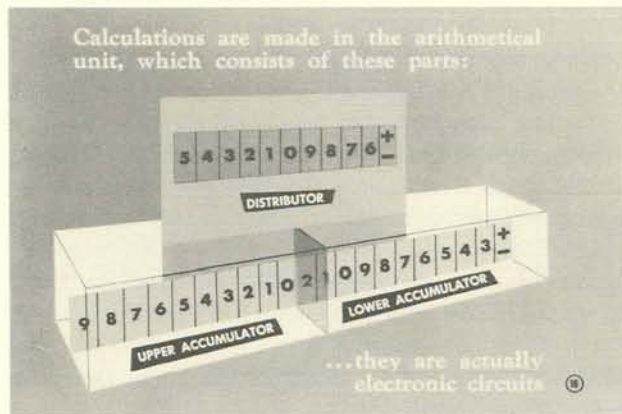


The solution of a problem requires a group of instructions called a program. The program, once stored on the drum, is followed over and over by the 650 to process each set of input data. Three considerations are involved in designing a program. First, the 650 solves a problem step by step—one operation at a time. This involves analyzing each problem to break it down into individual steps (read the data, add "A" and "B" and so on). Second, the 650 follows each individual step repetitively—one after the other. This involves planning the steps in the most efficient sequence. Third, the 650 "looks" for more than an operation to perform on each step—it also "looks" for the location of the data and the location of the next instruction. This involves planning, for each operation, a location to contain the data and a location to contain the next instruction.

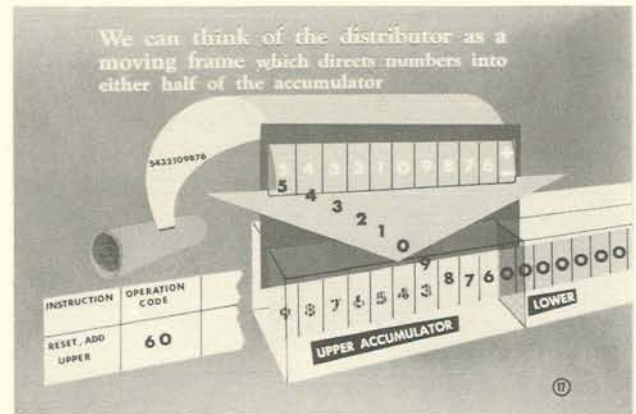


The program is written on a planning chart as it is being designed. It is then punched into a deck of cards, tested and filed for future use.

Each time a program is to be used, for example, to run a payroll, the deck is fed through the input for storage on the drum. The console is used to instruct the machine to start the storing process; the machine then automatically finishes it. After the program has been stored, the data cards are placed in the input feed and the console is again used—this time it instructs the machine to carry out the first instruction of the program. From then on the operation is automatic.

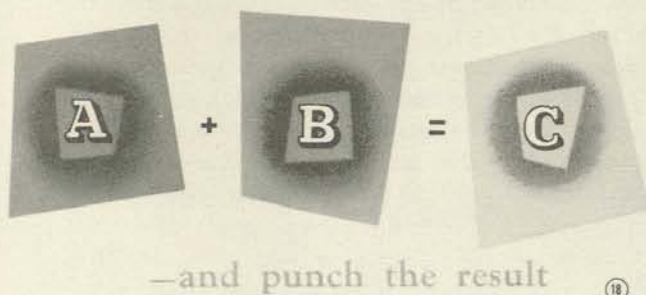


The arithmetical unit, where all calculations are made, consists of a distributor and a twenty-position accumulator. The accumulator, in turn, is considered as having an upper and a lower half. Each half of the accumulator, as well as the distributor, has its own address. A sign position is provided for both the distributor and the accumulator. They operate automatically, as do the sign positions on the drum, so that every time data is transferred within the machine, its sign is transferred with it.



The distributor directs data into and out of the accumulator. It automatically aligns itself with the upper or lower half, depending upon whether upper or lower accumulator codes are used. Operation code sixty resets the entire accumulator, moves the distributor to the left and adds into the upper half. Operation code sixty-five does the same thing for the lower half of the accumulator. Any data that moves into or out of the accumulator passes through the distributor. Operation code sixty-nine sends data straight to the distributor; it eliminates sending the data all the way into the accumulator and is used in transferring it from one drum location to another.

Let's apply what we have learned by working a simple problem:



The first step in working a problem is to design the program. To design it, the programmer selects those operation codes that will solve the problem and writes them down in the proper sequence. In the example illustrated, operation code seventy reads the card, sixty-five and fifteen add the factors, twenty stores

First we design the program of instructions

INSTRUCTION	OPERATION CODE	DATA ADDRESS	ADDRESS NEXT INSTRUCTION
0001 READ A CARD	70	0059	0002
RESET, ADD LOWER	65	0059	0003
ADD LOWER WITHOUT RESET	15	0060	0004
STORE ACCUMULATOR	20	0077	0005
PUNCH A CARD	71	0077	0001

the result for punching and seventy-one punches it. The two addresses needed to make each operation code a complete instruction are selected by the programmer as the program is written. As the machine follows the program, it automatically considers each data address as a location containing a number to be processed and each instruction address as a location containing a number that does the processing. The console starts the problem off by telling the machine where to find the first instruction.

The first instruction tells the machine to read a card. At this time, the factors punched in the card are "sensed" and sent to the drum. They reach the drum by way of any of ten-storage entries. The reason for the ten entries is that each band has a set of ten locations that will "read" factors from the card. The storage entries "align" the card, so to speak, with any set of these "read" locations; they can be thought of as devices that receive the data from the card, then "slide" along the drum to write the data in the read locations of one of the bands. The band they "slide" to is determined by specifying one of the band's locations as the data address of the read instruction. In the example illustrated, data address

0059 aligns storage entry with locations 0051 through 0060 (they are the read locations of the band that contain location 0059). The locations that receive the data are determined by control panel wiring; the data from the card that has been wired to the first storage entry is stored in the first read location, the second entry in the second location, and so on. In the example illustrated, factor "A" enters 0059, the ninth read location, and factor "B" enters 0060, the tenth, because the brushes that read factor "A" are wired to storage entry nine and factor "B" to ten.

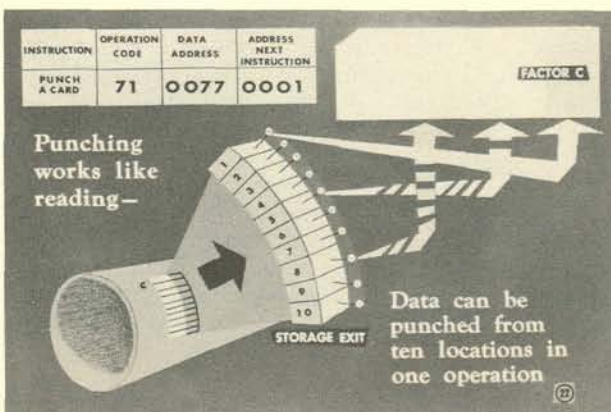
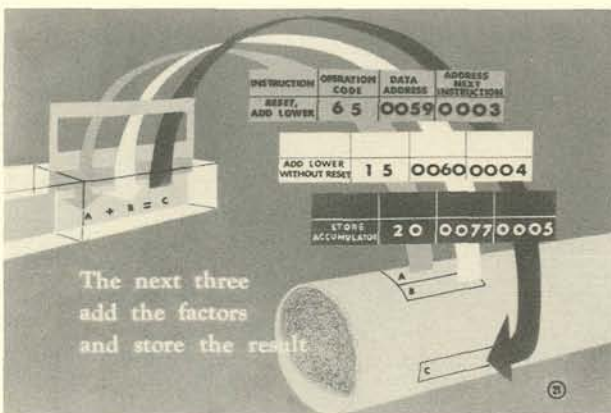
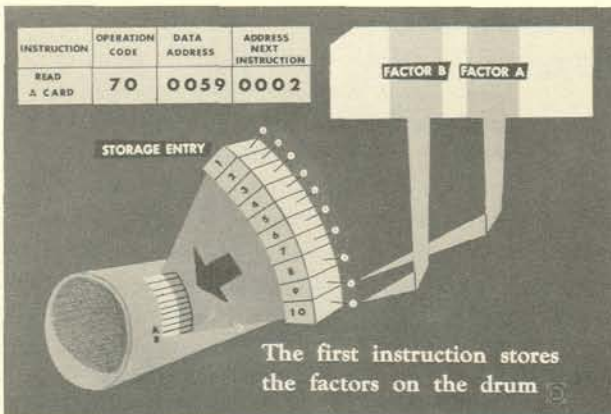
The second instruction, 65 0059 0003, resets the entire accumulator and adds into its lower half the data from location 0059—factor "A". As "A" is being processed, the 650 begins looking for location 0003, which is the address of the next instruction.

This instruction, 15 0060 0004, in turn, adds into the lower half of the accumulator, without resetting, the data from location 0060—factor "B". As in the case of the previous instruction, the 650 begins looking for its next instruction in location 0004 while it is processing factor "B".

The instruction in location 0004, the fourth one, stores what is in the lower half of the accumulator—the total "C"—in location 0077 and sends the machine to location 0005 for the next instruction. Location 0077 is specified to receive the result "C" because it is one of the ten locations on the band being used from which data can be punched.

The fifth instruction punches the results. The punching operation is the exact reverse of reading. All of the factors to be punched in one card are stored in the "punch" locations of one band (each band has ten; they are the punching counterpart of the "read" locations). Whichever band has been used to store the results is connected to the ten-storage exits by specifying one of its locations as the data address of the punch instruction. In the example illustrated, the punch locations selected are 0077 through 0086. The first punch location, 0077, is automatically connected to the first storage exit; the second location, 0078, to the second exit and so on through 0086 to the tenth. Control panel wiring determines which exits will punch. In the example illustrated, exit one is wired to the punches because it contains the result "C" from location 0077. Had additional information been wanted in the output card, it would have been stored in the same set of punch locations with "C" prior to the punch operation and additional control panel wiring would have connected the proper exits to the punches.

The instruction address of the fifth instruction tells the machine to read the first instruction in location 0001—and so the process is automatically repeated for the next set of input data.



Reading, calculating and punching overlap. While the 650 is reading one set of input data, it is calculating the previous set and as soon as it begins punching the result, it starts reading the next set.

An add operation code designates a plus operation and a subtract code a minus, but whether either will cause the arithmetical unit to add or subtract is governed by the sign rules of algebra. When the arithmetical unit is told to subtract a minus quantity, such as in $A - (-B)$, it cancels out the minus sign of the "B" factor with the minus sign of the subtract operation and the "B" factor is added instead. On the other hand, in an $A + (-B)$ operation, the "B" factor is subtracted. This algebraic ability of the 650 enables it to solve a broad range of problems.

A - B = C

INSTRUCTION	OPERATION CODE	DATA ADDRESS	SUBSTR. NEXT INSTRUCTION
0001 READ A CARD	70	0059	0002
RESET, ADD LOWER	65	0059	0003
ADD LOWER WITHOUT RESET	15	0060	0004
STORE ACCUMULATOR	20	0077	0005
PUNCH A CARD	71	0077	0001

SUBTRACT LOWER WITHOUT RESET 16

The program for subtraction of the same factors differs by only one instruction

Other arithmetical operations are performed by

REPETITIVE ADDITION and SUBTRACTION

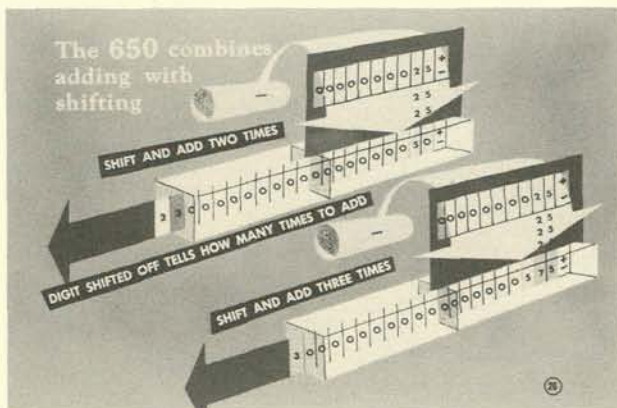
MULTIPLICATION can be done by REPETITIVE ADDITION...

...or by a combination of ADDING and SHIFTING

There are several ways to multiply. One is to add the multiplicand, over and over again, the number of times that equals the value of the multiplier. Under this system a ten-digit multiplier could require 9,999,999,999 adding operations. Another way combines shifting with adding and reduces the number of add cycles to the sum of the multiplier digits. This second system is well suited to the electronic speed of the 650.

When the 650 is told to multiply, it begins shifting the entire accumulator, containing the multiplier, to the left a position at a time. On each shift, a digit moves out of the accumulator to be analyzed by an "extra position". When the "extra position" encounters a significant digit, the machine adds the multiplicand into the lower half of the accumulator a number of times equal to the value of the shifted-off digit. After it has finished "adding in" the multiplicand on one shift, the entire accumulator shifts again and the process is repeated.

Multiplication ends after the tenth shift, or after what was originally the units position of the upper half of the accumulator has been shifted into the "extra position" and the "adding in" process has been completed.



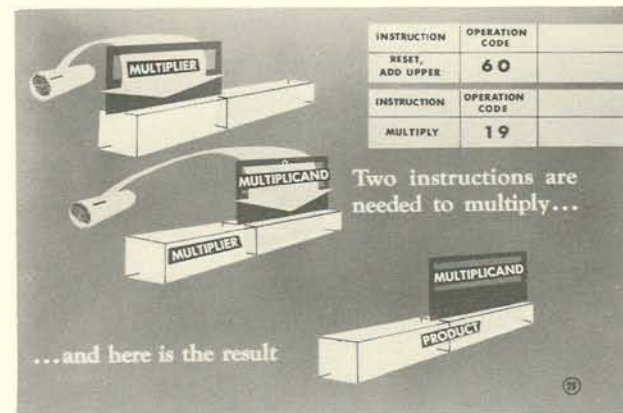
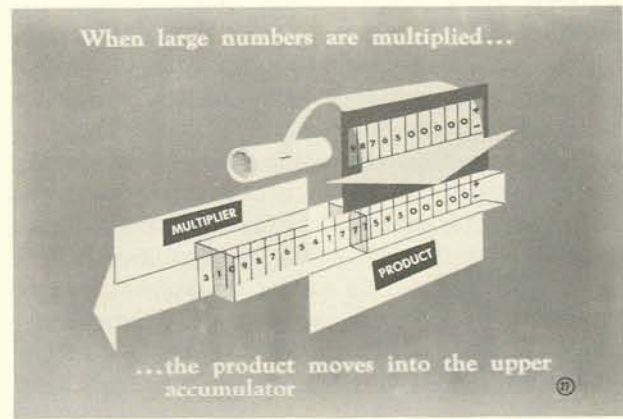
When larger factors are multiplied, the shifting process moves the product into the upper half of the accumulator (behind the disappearing multiplier).

The product never becomes involved in the "extra position" because the furthest it can move to the left in ten-shifting cycles is into the high-order position of the upper half of the accumulator and not into the "extra position".

To multiply, the multiplier is first "reset added" into the upper half of the accumulator. It is "reset added" to insure that it will not be augmented by any factor that might be standing there from some previous operation. The multiply instruction then starts the shifting process and introduces the multiplicand into the lower half of the accumulator to develop the product.

At the conclusion of multiplication, the multiplier is no longer in the upper half of the accumulator and the product is in the lower half. If the factors being multiplied are large enough, the product occupies all or a portion of the upper half, as well as the entire lower half of the accumulator.

The 650 multiplies a ten-digit multiplicand by a ten-digit multiplier to obtain a twenty-digit product. Larger products are conveniently obtained by breaking the factors into components and developing two or more independent products for consolidation.



DIVISION can be done by repetitive subtraction...

575	1
-25	
550	2
-25	
525	3
-25	
500	4
-25	
475	5
-25	
450	6
-25	
425	7
-25	ETC.
23	

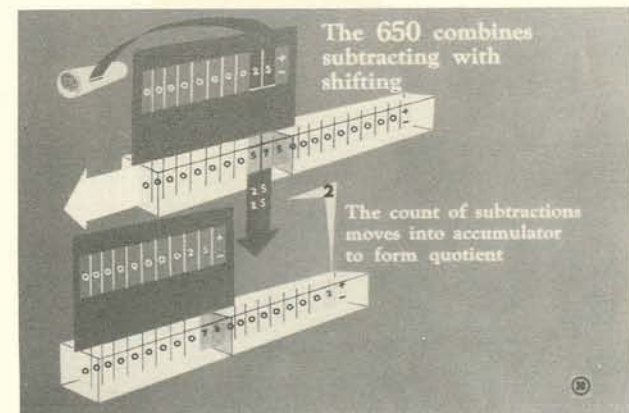
23
25 575
50
75
75
00

...or by a combination of subtracting and shifting

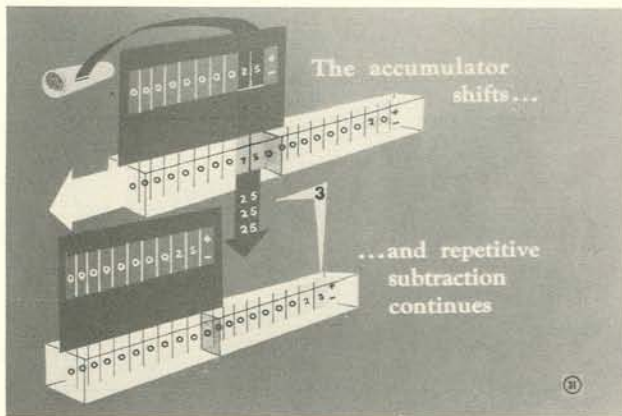
2	575
	-25
	-25
	75
	-25
	-25
	-25
2 3	00

SHIFT

Division can be done by processes somewhat similar to those employed in multiplication. A quotient can be obtained by counting the number of subtractions that are required to reduce the dividend to a remainder smaller than the divisor. As was the case in multiplication, this process is simplified by combining shifting with subtracting and lends itself well to the electronic speed of the 650.



The entire accumulator, containing the dividend, is progressively shifted to the left, a position at a time, moving the dividend through the upper half of the accumulator. For each shifting operation, the portion of the dividend in the upper half is repetitively reduced by the divisor, and the number of subtract cycles required to reduce it to a value less than the divisor is counted and inserted in the units position of the lower half as a quotient digit.



After one quotient digit is developed, the accumulator shifts and the subtracting process is repeated to develop the next one. On each shift, the portion of the quotient in the lower half of the accumulator also shifts so as to "open" the units position for the next quotient digit.

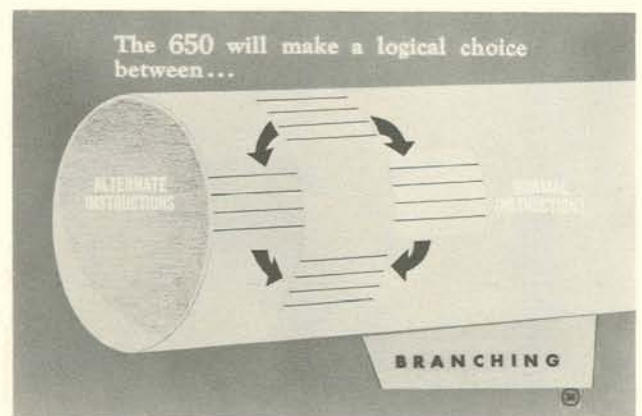
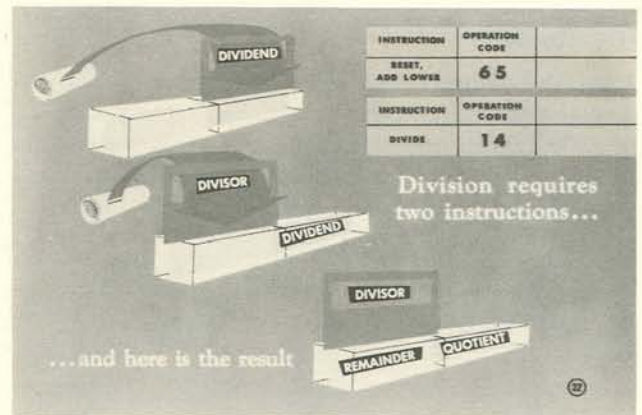
The 650 "learns" that it has developed each quotient digit by subtracting once too often, and "overdrawing" the dividend in the upper half of the accumulator to a negative figure. This signals the shift that precedes the next subtracting process. Before shifting, however, an add cycle restores the figure in the upper half to a positive value and adjusts the quotient digit to the proper figure.

As in multiplication, division ends after the tenth shift. This will have carried the quotient up to—but not into—the upper half of the accumulator.

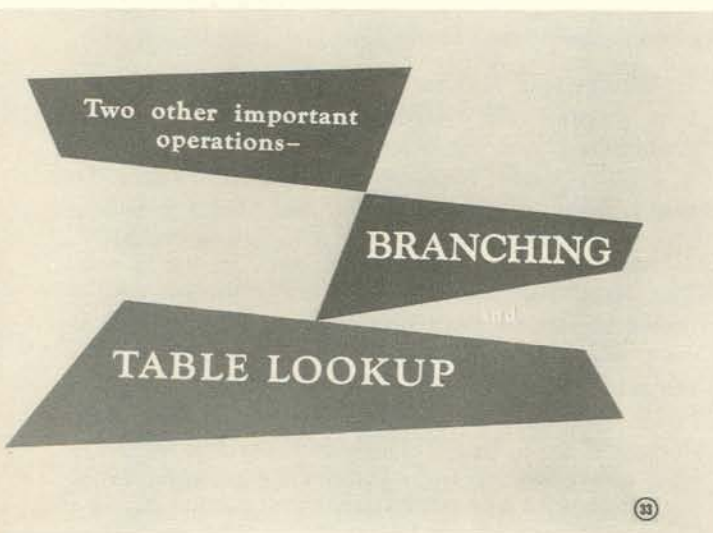
The first step in division is to introduce the dividend into the accumulator. Since division is a subtracting process, the accumulator is reset at this time to prevent the dividend from combining with any digits

left from a previous operation. Where the dividend exceeds ten digits, two instructions are required to read it into the two halves of the accumulator. The next step is division itself, wherein the dividend is progressively reduced by the divisor and the quotient is progressively introduced into the lower half of the accumulator.

At the end of division, the divisor is in the distributor, the quotient is in the lower half of the accumulator and the remainder, if any, is in the upper half. (Through the selection of a different divide code, the remainder is reset to zero at the conclusion of division.) The 650 will divide a twenty-digit dividend by a ten-digit divisor to obtain a ten-digit quotient. Larger quotients are readily obtained by breaking the division down into components.



Branching concerns the ability of the 650 to follow "special instructions" whenever it encounters "special conditions" in a problem. An example of this is found in processing a payroll, wherein it may not be desired to take voluntary deductions from those employees whose earnings fall below a specified minimum. In this case, a "special condition" exists for each employee whose earnings fall below the minimum, and "special instructions" see to it that no deductions are taken and that a record is made of this action.

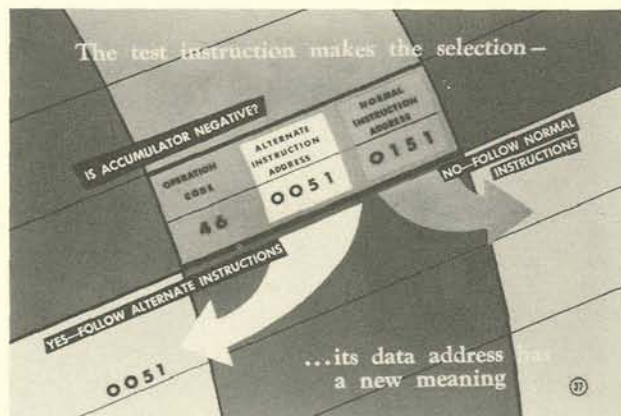
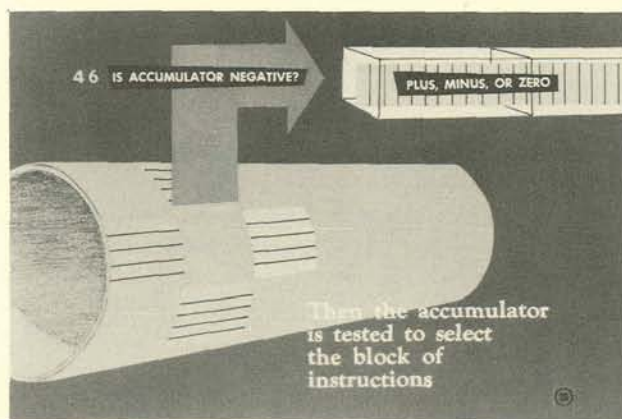
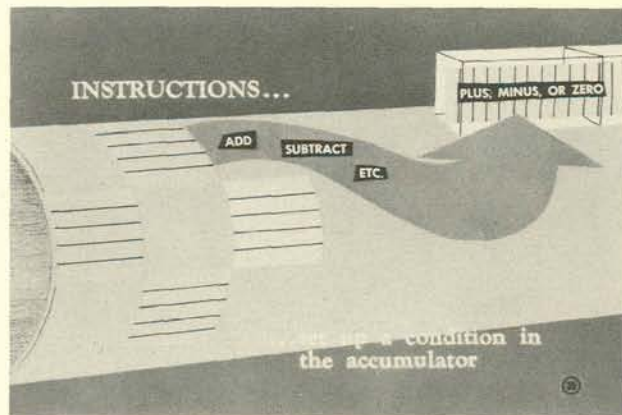


In order that the machine can handle all situations, a normal group of instructions is provided to "take" the deductions and an alternate group to show that no deductions are "taken". It then becomes the duty of the 650 to compare the earnings of each employee with the stated minimum and to "make" a logical decision to follow either the normal or alternate instructions for each employee, as the case might be.

The decision as to whether or not to branch is based upon re-stating the special condition in terms of any of three mathematical results in the accumulator: plus, minus or zero. (By subtracting the minimum from the earnings, in the example cited, a minus result would indicate that no deductions are to be taken.) In order to branch then, instructions are provided in the program that resolve the special condition being watched for into plus, minus or zero figures.

After the accumulator has been "set up" for the special condition, it is "questioned" by a test instruction. The operation code for the test instruction determines which of the three questions is asked—is the accumulator plus, minus or zero.

In the example illustrated, the test instruction asks whether or not the accumulator is negative; depending upon what the answer is, the next instruction followed will either be a normal or an alternate one.



The accumulator will answer "yes" or "no" to the test instruction's question. If it answers "no", it sends the 650 to the next normal instruction (they take the deductions in our example); if it answers "yes", it sends the 650 to the first alternate instruction (they make an indication that no deductions are being taken in our example).

Since the test instruction sends the 650 to either of two instruction locations, it must contain two instruction addresses. This it does—the instruction

address section contains the location of the next normal instruction; the data address section has a new meaning and now contains the location of the first alternate instruction.

Once the 650 has branched, the instruction address of each instruction keeps it on the branch.

In the example discussed, the last alternate instruction re-enters the main program below the normal branch that takes the deductions. The normal branch, in a like manner, bypasses the alternate branch. In this fashion, the 650 does not take deductions and also indicate that no deductions have been taken.

The distributor, as well as the accumulator, can be tested to initiate branching. Regardless of which is used, the principle of the test instruction remains the same—to send the machine to "this location" or "that location" for the next instruction. The only difference lies in usage. The accumulator is normally used to test for conditions that arise during calculation, whereas the distributor is used to test for conditions that can be indicated beforehand.

In a **table lookup** operation, the **650** selects data from tables stored on the drum



The table lookup feature eliminates the tedious process of manually consulting tables of rates or other coded data for needed information. The 650 will store vast tables on the drum and then consult them at electronic speed to obtain desired data.

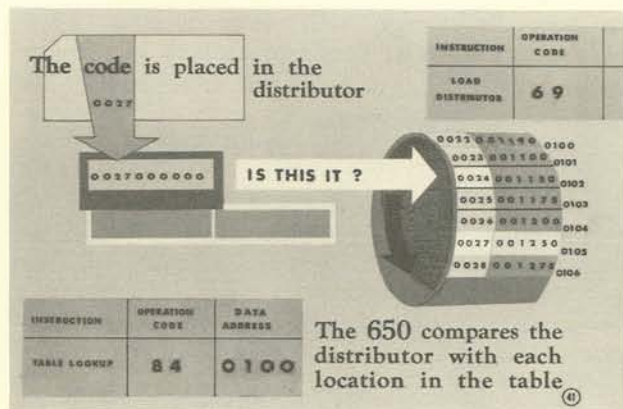
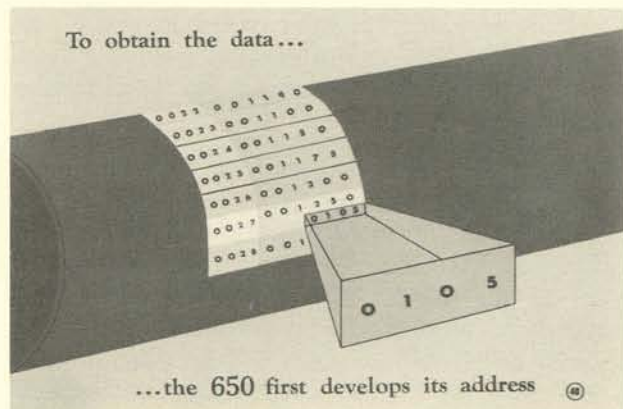
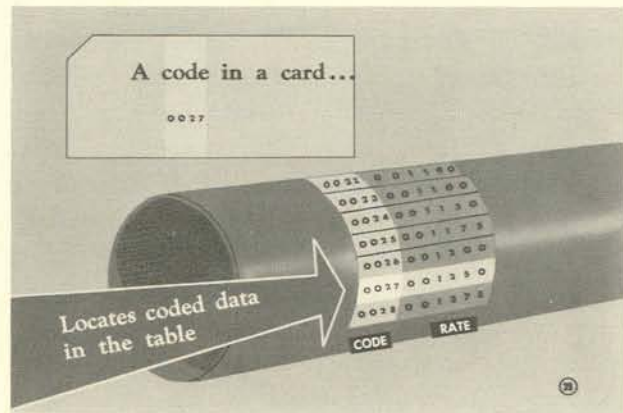
Data is found in the table by means of its code. The code is punched in the input card, together with the factors that pertain to the search.

Each table is stored on the drum in ascending sequence by code number and in as many bands as are required to hold it.

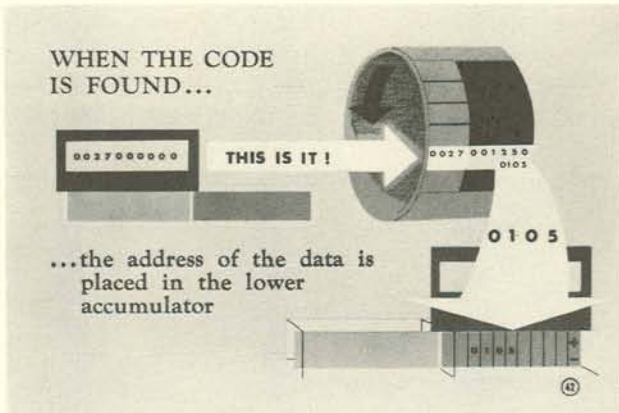
When told to find something, the 650, using the code from the card, searches the table from band to band, left to right, until the information is found. Since a complete band is searched in a single revolution of the drum, and since the search may continue from one band to the next, tables are not stored in the last two locations of a band. These two fly past the recording heads while the search is being switched from one band to the other.

The final objective of the table lookup operation is to obtain the data. To do this, the 650 "develops" its address. The address is then "read" by the 650 to get the data. The obtaining of an address, instead of data, constitutes a very desirable feature. In many table lookup operations, data is found on the drum so that it can be updated and then put back for later use. This summarizing operation would be very cumbersome—if not impossible—had the table lookup delivered the data instead of the address.

The development of an address involves special abilities of the distributor and accumulator. Before starting the lookup, the code of the data being searched for is placed in the distributor; when the 650 is then instructed to start the table lookup, it progressively compares, as the drum rotates, the



contents of each location in the table with the contents of the distributor. The search starts at the top of the band that contains the location specified by the data address of the table lookup instruction.



When a number is found in the table that is equal to or higher than the number in the distributor, the address of its location is automatically inserted in the fifth through eighth positions of the lower half of the accumulator. The 650 develops its address by adding the number of unsuccessful comparisons in the table to the data address of the table lookup operation.

Where both the data and its code are in the same location, the address developed is the address of the data. In tables where the number of digits required by the code and data exceed ten, the address developed can be the location of the code. It is common practice in such tables to store the codes in one sequence of locations and the corresponding data in another—so that all data is, let's say, 200 locations away from its code. Had this been the case in the example illustrated, 0105, the location of the code, would have been developed by the table lookup, not 0305, the location of the data. To obtain 0305, an instruction would have been placed in the program to add 200 to the address developed by the table lookup operation.

The objective of the table lookup, in positioning the data address in the lower half of the accumulator, is that the data address be completed into an instruction to process the data. This implies that the 650 can "look into" the accumulator for an instruction. This it can do, for both halves of the accumulator are addressable, just as are the locations on the drum.

So, following the table lookup operation... an instruction in the program... causes something to

be added onto the data address in the lower half of the accumulator... that will make it into a complete instruction to process the data.

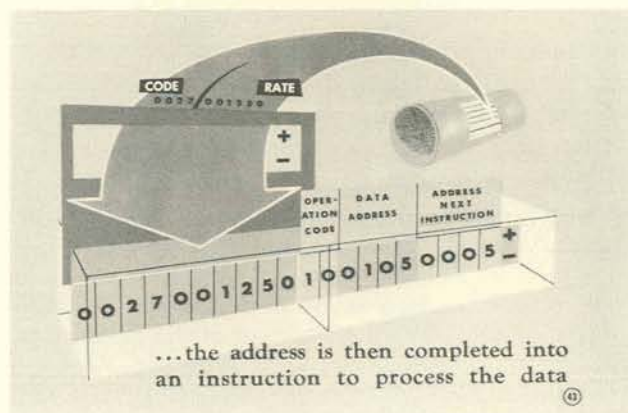
In the example illustrated, it was desired to add, into the upper half of the accumulator, the data from location 0105. Therefore, the instruction in the program that followed the table lookup added 10 0000 0005 into the lower half of the accumulator—and sent the 650 to the lower half to read the instruction. The 650 did this, and saw there 10 (add into the upper half of the accumulator) 0105 (the data from location 0105) 0005 (and find the next instruction in location 0005).

The exact nature of the instruction that was "manufactured" in the lower half of the accumulator was planned as the program was written. This same planning determines what the instruction in 0005 does to process the data, now in the upper half of the accumulator.

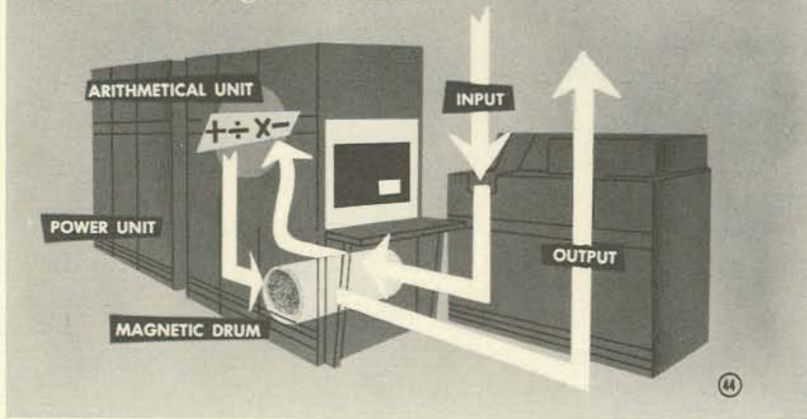
The 650 performs many additional operations; only those of a fundamental nature have been explained, for an understanding of those that are presented carries over to those that are not.

A list of 650 operation codes follows. They reflect the ability of the 650 by itself—without regard to the impact of special devices and additional features.

Individual codes, chosen from this list, and assembled in the proper sequence, transform even the toughest problems, as we know them, into the easy electronic language of the 650—the language it knows so well.

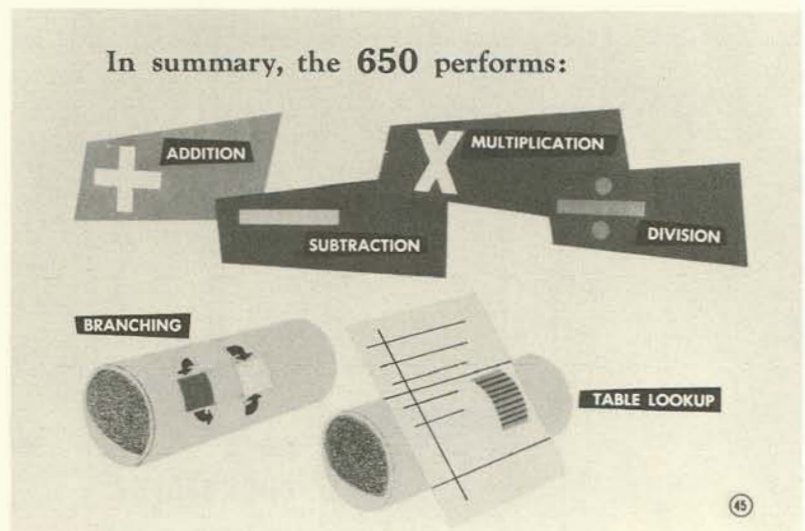


Let's review the flow of data through the 650...

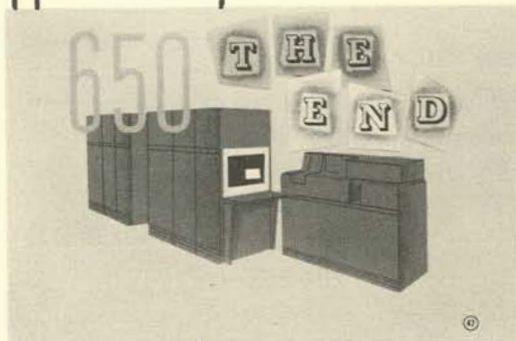
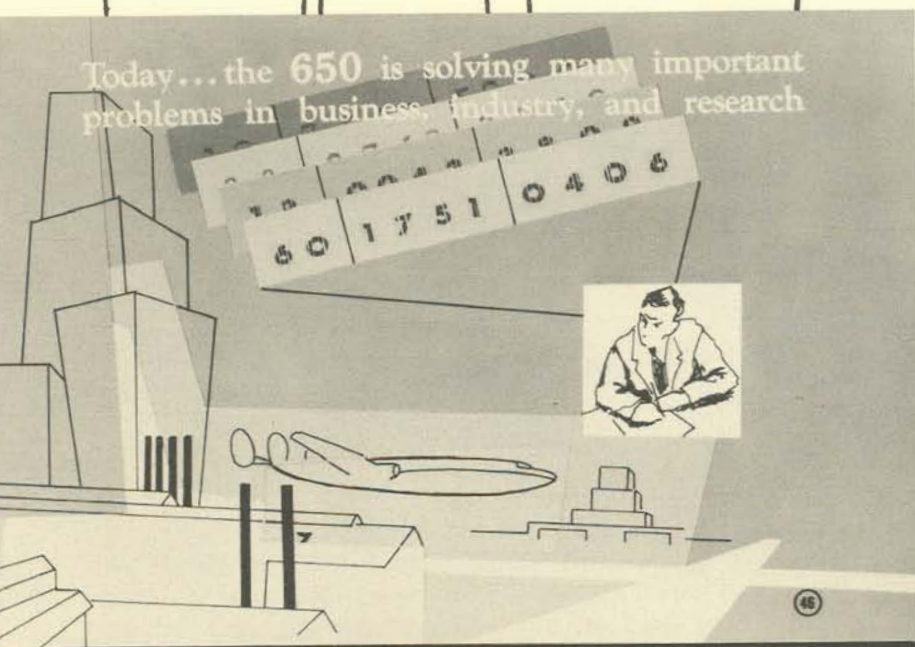


- 00 No Operation
- 01 Stop
- 10 Add Upper Accumulator
- 11 Subtract Upper Accumulator
- 14 Divide Without Reset
- 15 Add Lower Accumulator
- 16 Subtract Lower Accumulator
- 17 Add Absolute
- 18 Subtract Absolute
- 19 Multiply
- 20 Store Lower Accumulator
- 21 Store Upper Accumulator
- 22 Store Data Address
- 23 Store Instruction Address
- 24 Store Distributor
- 30 Shift Right
- 31 Shift and Round
- 35 Shift Left
- 36 Shift and Count
- 44 Branch Non-Zero Upper
- 45 Branch Non-Zero
- 46 Branch Minus
- 47 Branch Overflow
- 60 Reset Add Upper Accumulator
- 61 Reset Subtract Upper Accumulator
- 64 Divide - Reset Upper Accumulator
- 65 Reset Add Lower Accumulator
- 66 Reset Subtract Lower Accumulator
- 67 Reset Add Absolute
- 68 Reset Subtract Absolute
- 69 Load Distributor
- 70 Read
- 71 Punch
- 84 Table Lookup
- 90-99 Branch on Distributor 8

In summary, the 650 performs:



Today... the 650 is solving many important problems in business, industry, and research



IBM DATA PROCESSING

INTERNATIONAL BUSINESS MACHINES CORP.
590 MADISON AVENUE, NEW YORK 22, N.Y.