1968

A Study of Computers and Computer Programming

Janet Moffett
Ouachita Baptist University

A STUDY OF COMPUTERS AND COMPUTER PROGRAMMING

A Paper

Presented to

the Department of Mathematics

Ouachita Baptist University

In Fulfillment

of the Requirements for H491,

Honors Special Studies

---

by

Janet Moffett

May 1968

Computers have become a great aid to scientists and businessmen. Their speed and accuracy has enabled much progress in many fields. Although computers are capable of many tasks, they are dependent upon the programmer and can only do what they are told to do. Computers are based on logical organization and act according to orgainzed data.

The logical components of most computers are:

(1) Input

(2) Internal Memory

(3) Arithmetic Section

(4) Console

(5) Automatic Control

(6) Output

(7) Auxiliary Memory

There are many forms of input used by computers today. Some of the most common forms of input are by typewriter, punched cards, magnetic tape, punched paper tape, and console switches. The typewriter is usually used in accordance with other types of input media. It may be used to correct data which is being read into a computer or to insert needed variables in a standard program. Punched cards are the most common form of input data and will be discussed in detail later in this paper. Magnetic tape has its major function as a form of internal memory. It has a disadvantage in that data must be stored in some specific order. It has the advantages of speed and ease of handling. Punched paper tape is somewhat of a cross between punched cards and magnetic tape. On magnetic tape, data is received or recorded

through electrical impulses on the tape. Punched paper tape records data by punching holes in the tape like those of punched cards. A sample of paper tape is illustrated below.

Punched Paper Tape

Paper tape does not have the speed of magnetic tape but its data is easily transferred to punched cards or can be transmitted over teletype. Console switches are on all computers and permit the operator to enter information directly into the computer. This form of input is very slow as compared to computer speeds and is used only for exceptional situations.

The internal memory of a computer may be compared to that of a file. It stores data and brings it forth when needed. The data, however, is stored much more compactly and may be found in a much shorter time. There are various forms of internal memories. Magnetic tape has already been discussed. Similar to magnetic tape is the magnetic drum. The drum, which is cylinderically shaped, revolves thousands of times per second. Read and write heads are located above the drum surface as shown in Figure 1.

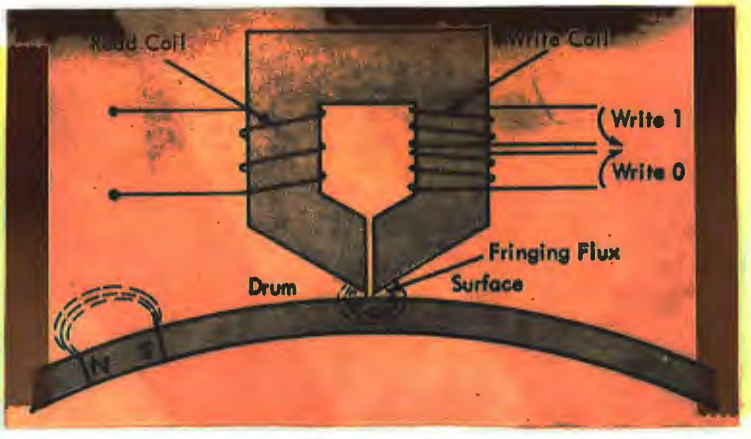


Figure 1
Drum Recording

The computer may read or write information on the drums. The drums are organized into bands and futher into tracks or words. Each track within each channel may be called upon when needed. The arrangement of tracks and bands is illustrated in Figure 2.

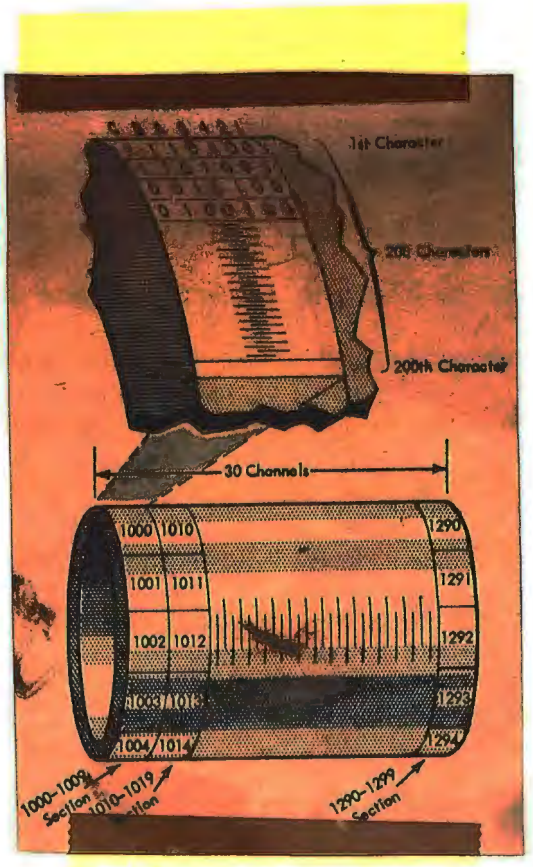
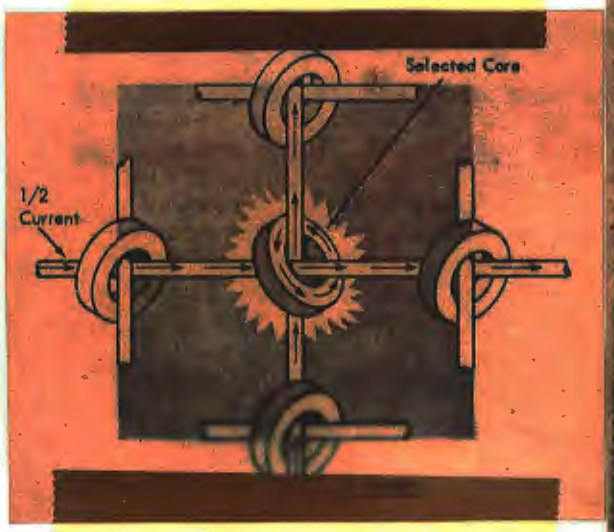


Figure 2
Design of Drum Storage

A very common form of internal memory is the magnetic core. The magnetic core is a tiny ring of metal a little larger than a pinhead. Two wires intersect the core, and a third senses the magnetization of the core. Information is recorded by magnetizing the core in either a positive or negative direction. The magnetic cores are arranged in a grid pattern (Figure 3) so that current flowing over the wires will change the magnetization of only one core. Each grid of magnetic cores is called a core plane.



Magnetic Core

Magnetic Core Plane

Figure 3

The magnetic disk is also used for internal memory. It is not ordinarily employed as a primary internal memory but is used to supplement the capacity of a computer. Each record shaped disk (Figure 4) contains tracks on which information is recorded. A double-headed access arm reads and records data on the disks. The disks are arranged one on top of the other and the access arm move up and down a rod by the stack of disks to read the different disks (Figure 5).

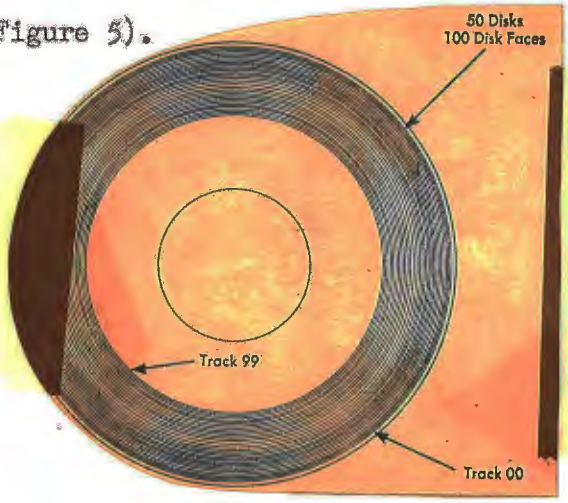


Figure 4
Magnetic Disk

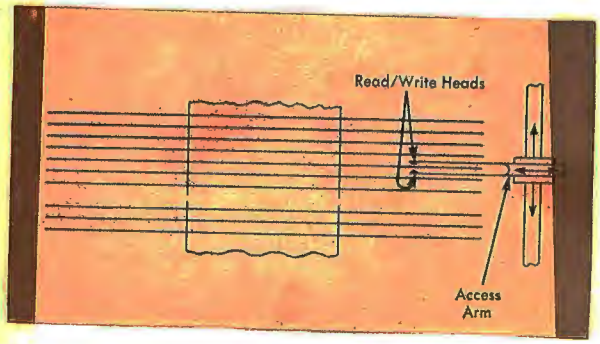


Figure 5
Disk Storage and Access Arm

Internal memories are limited in capacity. Auxilary memories may be used to expand the memory of the computer. The auxilary memories are usually slower and less expensive than central memories.

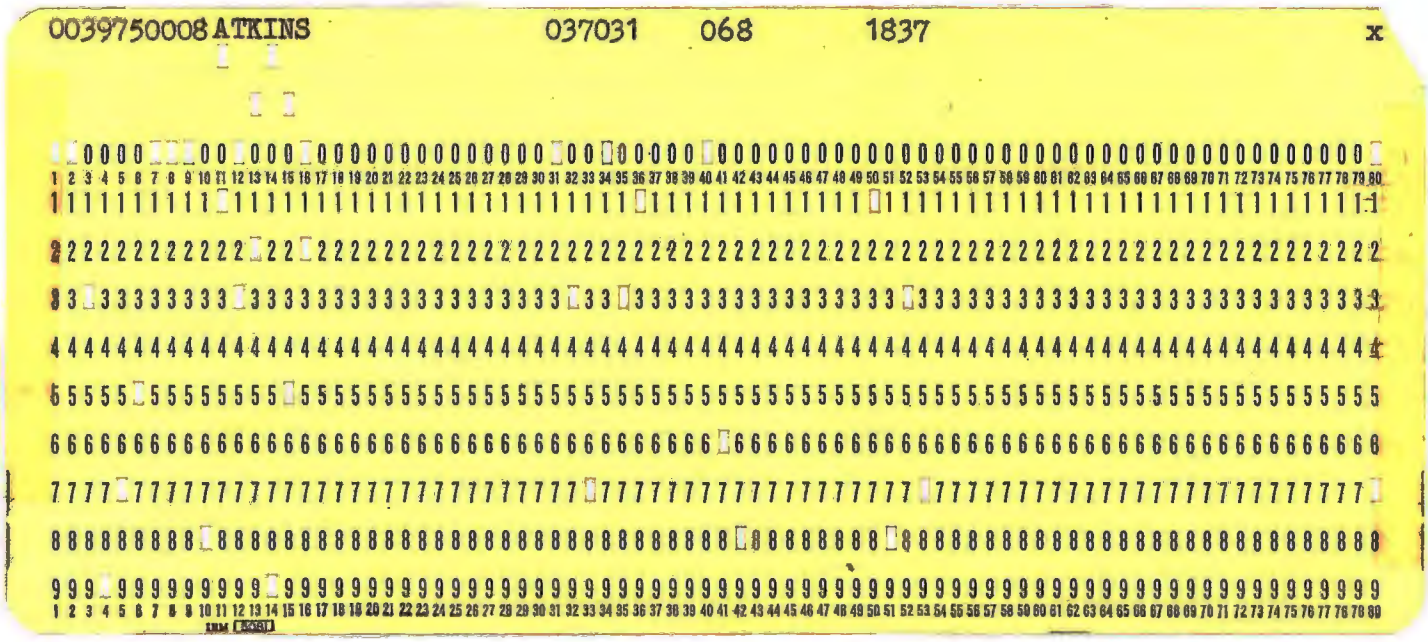
The basis for nearly all internal mathematics in computers is the binary system which uses a base of two. The binary code permits the recording of numbers by the presence or absence of magnetic spots, holes, lights, or electronic impulses. The absence of a hole or magnetic impulse would mean 0 and the presence of a hole or spot would indicate 1. Since 0 and 1 are the only symbols used in a base of two, all numbers may be recorded by the presence or absence of a hole or magnetic impulse.

The control console is used to contact the computer during its work cycle. It is composed of a series of lights, switches and buttons. There are various automatic controls within the computer to keep information being recorded from being confused with information previously recorded and to prevent the recording of incorrect data. The automatic controls turn over control to the control console when some misstatement or malfunction has been detected. The error may then be corrected through the control console.

Output forms are much the same as input forms, however, there are more output forms in use. Some of the most common output forms are typewriters, punched cards, magnetic tapes, punched paper tapes, magnetic disks, movie films, television tubes, and printers. The most widely used method of output is the printer. It has the advantage of presenting data ready for use without additional processing before it can be read by an average person.

A number of computers which have been developed use the punched card as a means of recording and reading data. Machines which sort, read, merge, file, and record punched cards save tedious clerical effort and money and work accurately and rapidly. There are several types of punched cards. The

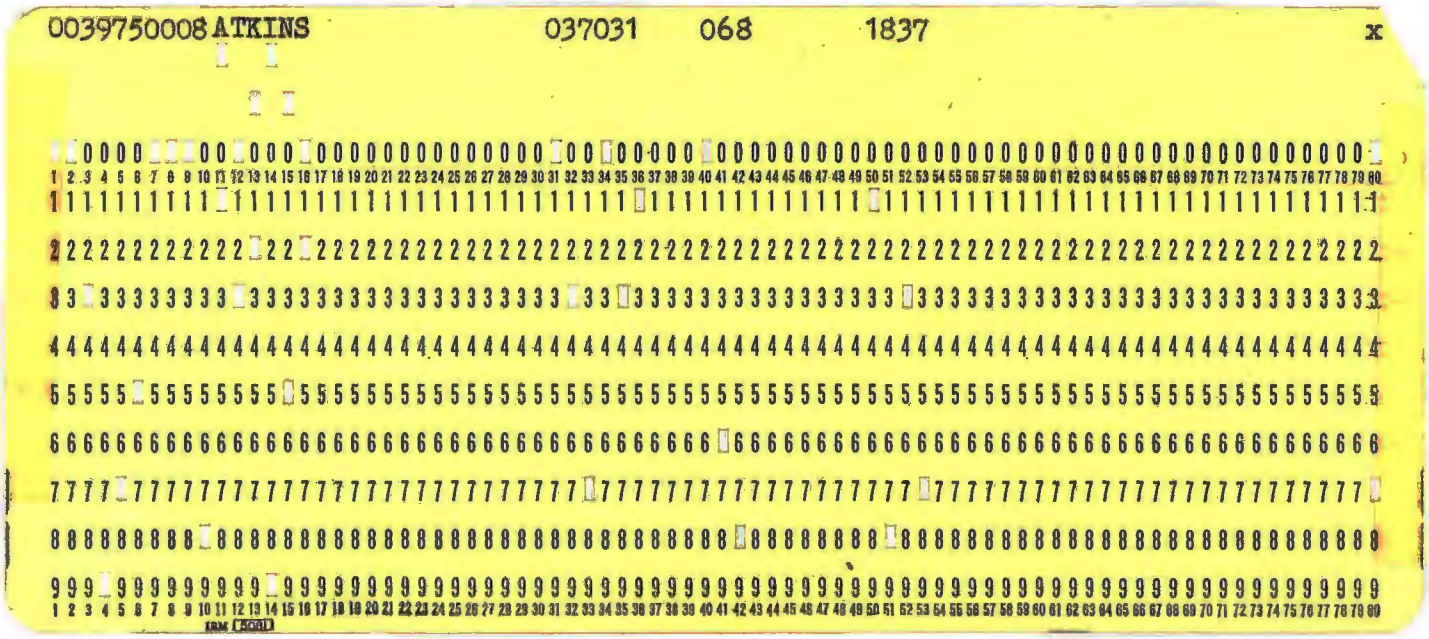
IBM punched card, which is most widely used, has 80 columns accross and 12 rows down. The 12 rows consist of figures 0 through 9 and two rows above the figures which are not printed on the cards. Below is a typical punched card.



Punched Card

Figures are indicated on the card by punching out a rectangular hole over the number indicated. Letters are indicated as follows: A through I are made by a punch in the top row and a punch in the number that letter is in the alphabet. J through R are made by a punch in the second row and a punched number of the letter beginning with J as one, K as two, etc. S through Z are indicated by punching the 0 row and a number two through nine with S as two, T as three, etc. A punch in the first row may indicate a plus, the second row a minus. Various other symbols may be made by using the first two lines with the numbered lines. A key punch machine is used to prepare the punched cards. The keyboard of a key punch is much the same as a typewriter with the numbers arranged on the right like that of a ten-key adding machine and operated by the right hand. The cards are read and sorted by sorters by

IBM punched card, which is most widely used, has 80 columns accross and 12 rows down. The 12 rows consist of figures 0 through 9 and two rows above the figures which are not printed on the cards. Below is a typical punched card.



Punched Card

Figures are indicated on the card by punching out a rectangular hole over the number indicated. Letters are indicated as follows: A through I are made by a punch in the top row and a punch in the number that letter is in the alphabet. J through R are made by a punch in the second row and a punched number of the letter beginning with J as one, K as two, etc. S through Z are indicated by punching the 0 row and a number two through nine with S as two, T as three, etc. A punch in the first row may indicate a plus, the second row a minus. Various other symbols may be made by using the first two lines with the numbered lines. A key punch machine is used to prepare the punched cards. The keyboard of a key punch is much the same as a typewriter with the numbers arranged on the right like that of a ten-key adding machine and operated by the right hand. The cards are read and sorted by sorters by

sensing electrical impulses through the holes in the cards. As a card goes through the sorter it passes between a copper drum and a row of electrical brushes (Figure 6). An electrical current is completed when a brush contacts the drum due to the hole in the card.

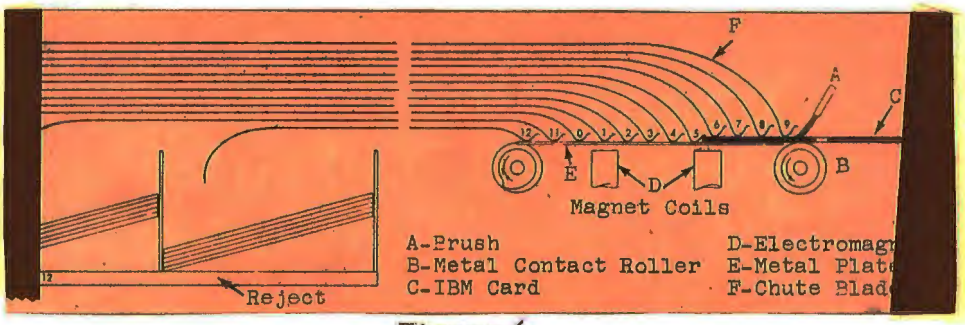


Figure 6
Diagram of IBM Card Passing Through Sorter

Punched cards do not actually go into a computer. They are read by a special card reader which converts the sensed holes into current which travels along wires to the computer. The messages are received inside the computer and converted to magnetic spots or magnetic fields on special media.

The IBM 1620 system uses punched cards as its principle means of input and output. The data is prepared on punched cards and manipulated by instructions stored in memory in the form of numbers. These numbers move into program registers where they are broken down into an operation code and data addresses. The operation code tells the computer what is to be done with the information. The data addresses tell the computer where the information is located in internal memory. The results of the computations may then be transferred from general storage or the product area to the card punch and punched out.

The 1620 uses a magnetic core storage for internal memory. This permits the use of variable word length and permits each digit of information to be addressed individually. The internal memory of the IBM 1620 requires 12 planes of 10,000 cores each. The memory module is arranged so that each

digit of information will be represented by six planes making use of six cores in a line perpendicular to the planes (Figure 7). Six cores, one in each of six planes, are required to show one binary-coded decimal digit.

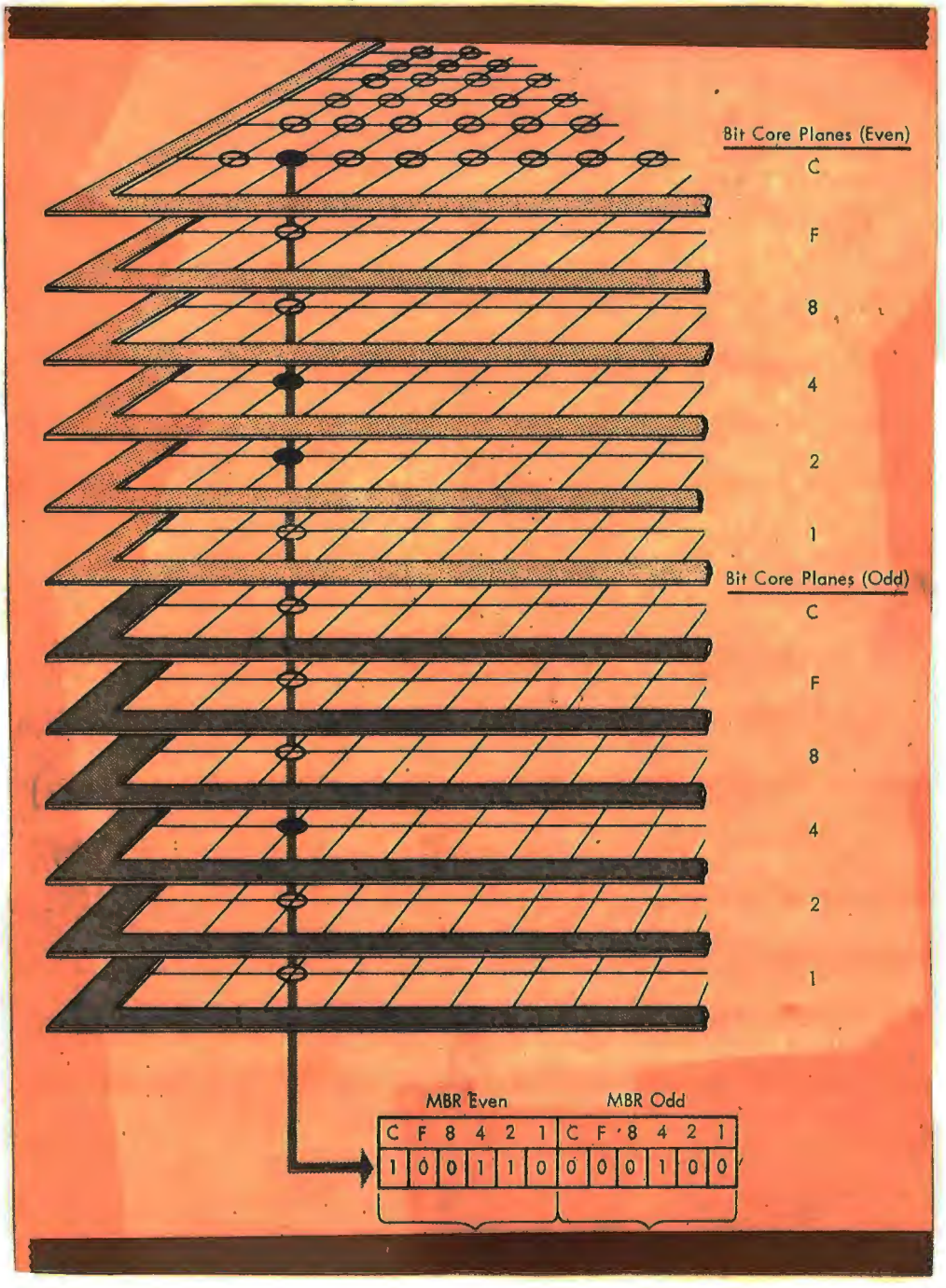


Figure 7
1620 Memory Module

The only items which can be stored in computers are numbers. All letters must be designated by a numerical subsitute in order to be put into memory. All instructions are expressed in terms of decimal digits. There are 34 coded operations which the 1620 system can perform. Each decimal digit in storage must have a location from which it can be called up as needed. The 1620 has a two-address instruction format. Each address consists of five digits. One address is called the F address and the other the Q address. The two five-digit addresses with the two-digit operation code require a twelve-digit instruction format. A mnemonic code may follow the Q address to aid the programmer. This code is not recorded by the computer. A typical program to add two numbers which are already in storage would be:

Code	F address	Q address	Mnemonic code
21	15550	16660	A

The code for addition is 21. The numbers to be added are located beginning at cell 15550 and cell 16660.

Since the 1620 uses a magnetic core memory and has variable word length, it must have a method of designating the beginning and end of a word. This is accomplished by use of a flag bit. Inside the memory of the computer the flag bit is a one in the flag bit plane for the particular group of cores representing the number in storage. The programmer represents the flag bit by placing a dash above the decimal number. A flag bit at the left of the word denotes the beginning of the word. A flag bit at the right of the word denotes that it is negative. Internal transmission in the computer is from right to left. Therefore, if there is a flag bit at the addressed digit, the computer assumes the number to be negative. The computer then proceeds to the left until it encounters a flag bit denoting the end of the field. The only time the computer proceeds from left to right is during input and output transmission to or from external devices. In input data, field marks

are ignored as limiting devices and a record mark (‡) is used instead. Field marks are included in output data. A record mark is represented on a punched card by a punch in the 2 position and the 8 position in the same column.

The 1620 contains the addition and multiplication tables and refers to them for answers to various parts of problems. This simplifies the component requirements of the system and increases the speed and simplicity of control. The multiplication table occupies storage cells from 00100 through 00299. The addition table is stored in cells 00300 to 00399. In addition or subtraction problems, the answer is stored at the P address which was occupied by one of the factors in the problem. In multiplication or division problems the answer is recorded in a section of storage called the product area. It is located in cells 00080 through 00099. The product area is automatically cleared to zero by the multiply and divide commands. The load area of the 1620 internal memory extends from cells 00000 to 00079. This contains 80 decimal digits which is equivalent to the size of the IBM punched card. A card is read into the computer and the computer automatically proceeds to 00000 for its first instruction.

Programming a computer first involves stating the problem and defining a step-by-step procedure for solving the problem. In order to do this the computer must be contacted in a language which it understands. "Fortran" is a language developed in which procedures can be readily and accurately expressed. It is a procedure-oriented language made up of a small group of statement types. In formulating statements, the rules and regulations which must be followed are analogous to the grammatical and syntactical rules of English. The characters used by Fortran are all numerical characters 0 to 9, all alphabetic characters A to Z, and the following special characters: . , + $ * - /) (= .

Each step of a procedure is a statement. A program consists of a string of Fortran statements. Statements which supply information about the procedure to the processor are known as declarations. Whenever one statement must be referred to by another it is given a numerical label. A typical Fortran program requiring the selection of the largest number in a group of numbers is as follows:

Label	Statement or Declaration	
	DIMENSION A(100)	Declaration
	Z = A(1)	Statement
	DO 20 J = 2, N, 1	Statement
	IF (Z - A(J)) 10, 20, 20	Statement
10	Z = A(J)	Statement
20	CONTINUE	Statement

The declaration DIMENSION A(100) declares that a space in memory must be allocated for up to 100 items in the collection of elements called A. Z = A(1) and Z = A(J) are arithmetic subsitution statements. Z = A(1) says to place the contents of the first of the 100 locations referred to above into the memory location Z. A= A(J) instructs the computer to place the contents of A_J in Z. DO 20 J = 2, N, 1 is the iteration statement. It instructs to execute repeatedly the statements which follow, down to and including the one labeled 20; each time varying the index, J, from an initial value of 2, in increments of 1, up through N. IF (Z -A(J)) 10, 20, 20 is a conditional statement meaning, whenever the expression Z - A(J) has a value less than zero, proceed to statement 10; whenever it has a value equal to or greater than zero, proceed to statement 20. The CONTINUE statement marks the last of the statements under repeat control of the preceding DO statement. The word CONTINUE merely means, do nothing, just proceed from here, and is used to terminate a DO.

The preceding program illustrates characteristics of the Fortran language. It is incomplete, however, because it contains no provision for entering data

or delivering the results. Input and output statements are required. An input statement specifies the items of information which are to be transferred from the punched cards or other input documents. The statement also refers by number to a format code which describes the form of the input information. For example, a statement ordering the computer to read in the N values of A from A_1 through A_N would appear as follows:

READ 11, N, (A(J), J = 1, N)

The format code is 11 and the items are N and A_J for J running from 1 through N. An output statement specifies the information to be punched on the output card or printed out. Printing the result from the location Z is accomplished by:

PRINT 21, Z

The format code is 21, and the item is printed from the memory location Z.

The Fortran language deals with "integers" and "floating point" numbers. The computer uses a different kind of arithmetic for each type of number. Integers are used for indexes in the control of repetition or for identification of individual elements. In arithmetic with integers the results are always rounded to the nearest integer. The results of integer arithmetic are always exact values and never approximations. Floating point numbers are real numbers and are a computer version of "scientific notation" of numbers. There are two basic external forms of floating point constants. Internally, in memory, all floating point constants have the same form. They consist of an exponent and a mantissa. The exponent is comparable to that of the exponent in scientific notation. The mantissa supplies the actual value. By expressing constants in two parts, numbers of much greater range may be stored using a limited memory capacity. The decimal point of the mantissa is always assumed to be at the far left. The exponent is an integer

code which represents powers of 10. A range from 10^{-50} to 10^{+49} can be represented by exponents coded 00 to 99 as seen in the following table.

Internal Form			
Sign	Exponent	Mantissa	Value
+	51	10000000	$.1 \times 10^{1}$
+	50	99999999	$.99999999 \times 10^{0}$
+	52	12345678	$.12345678 \times 10^{2}$
-	60	10000000	$-.1 \times 10^{10}$
+	99	10000000	$.1 \times 10^{49}$
-	40	10000000	$-.1 \times 10^{-10}$
+	00	10000000	$.1 \times 10^{-50}$

Externally, floating point constants may be written in a normal manner with a decimal point contained in the number. Such as 0., 1.5, -0.05, +100.0, -4., -123456.78, .00059. They may also be written in exponential form known as E-notation. The constant contains one or more digits with a decimal point, followed by the exponent.

Fortran Notation	Scientific Notation
.05E-2	$.05 \times 10^{-2}$
-.05E-02	$-.05 \times 10^{-2}$
.05E2	$.05 \times 10^{2}$
+.05E2	$.05 \times 10^{2}$
.5E2	$.5 \times 10^{2}$
5.E2	5×10^{2}

A variable is a symbol or name which refers to a place in memory where the value represented by that name is stored. A variable consists of one or more alphabetic or numerical characters, the first of which must be alphabetic, and the last must not be the letter F. The computer distinguishes between integer variables and floating point variables by beginning all names for integer variables with I, J, K, L, M, or N. A group of variables which belong to a single class or collection is known as an array. Each variable in an array is an element. A string of numbers in a single row or column is considered as one-dimensional or a linear array. Each element in a linear

array may be identified by a linear subscript. For example, the elements of the array A would be represented in Fortran notation as A(1), A(2), A(N), etc. Arrays consisting of several rows and colums are two- or three-dimensional arrays and the elements are identified by two or three subscripts. The elements of an array are either all floating point numbers or all intergers.

The special characters +, -, *, /, and ** are used to signify the following:

A + B	means	A plus B
A - B	means	A minus B
A * B	means	A multiplied by D
C/D	means	C divided by D
Y**X	means	Y^X
A**2	means	A^2

The sequence in which individual terms of an expression are to be evaluated and collected must follow a set pattern. When unaltered by parenthesis, arithmetic operations are performed in order of precedence as follows:

Symbol	Operation
**	Exponentiation
-	Negation
*, /	Multiplication, Division
+, -	Addition, Substraction

A + B/C + D**E * F - G means $A + \frac{B}{C} + D^E \times F - G$

A * B/C * D/E * F means $(\frac{(AB/C)D}{E})F$ or $\frac{A*B*D*F}{C*E}$

Parenthesis as used in algebraic expressions may be used to override rules of precedence and to clarify expressions.

One advantage of Fortran is the ease in which expressions referring to functions of one or more variables may be written. To the computer, a function is a separate subordinate program designed to perform a specific task when given the arguments of the function. The main program calls upon the subordinate programs to perform certain functions or tasks. The names of mathematical functions included in Fortran are:

Name	Meaning
ABSF	Absolute value
SQRTF	Square root
LOGF	Logarithm - base e
EXPF	Powers of e
SINF	Sine of an angle in radians
COSF	Cosine of an angle in radians
ATANF	Arctangent in radians of a tangent value

All these functions require floating point values, and are characterized by ending in F. Predefined functions of [illegible] values must begin with X and end in F.

Decisions are made by the computer by testing conditions of several types. Conditional control statements and iteration statements are the components of the Fortran language which are used for decision making. The IF statement is a conditional control statement and permits control to be directed to one of three statements depending upon the value of an arithmetic expression. The general form of an IF statement is IF (E) s_1, s_2, s_3. It directs the computer to evaluate the expression (E). If its value is less than zero, go to statement s_1; if the value is equal to zero, go to statement s_2; or if the value is greater than zero, go to statement s_3.

The majority of the procedures concerned with numerical computatuios and information processing involve repetition. The repetition process is controled by the iteration or DO statement. The DO statement has the form DO s j = i, l, k,. s is numerical and designates the last step in the repetition. j is a variable whose value is to be changed during the repetitive process. j may be any nonsubscripted integer variable and has the initial value of i prior to the execution of the first repetition. i, l, and k are integer constants or variables and cannot be negative or equal zero. After the first execution of the process the value of i is increased by increments

of k until it exceeds 1 which is the limit of the process. When i exceeds 1, control is transferred to the statement after s. Before each repetition, the value of i is tested. If i is less than 1 the procedure is repeated until i is greater than 1. The CONTINUE statement is used to terminate a DO loop when the loop contains an IF statement. The CONTINUE statement is merely a procedural step for the purpose of representing a labled junction point.

Unless otherwise instructed, the computer will execute statements in the order they are written. The unconditional transfer statement, GO TO, enables the programmer to arbitrarily alter the sequence in which the program statements are executed. The simplest form of a GO TO statement is GO TO s; s representing the number of a statement. It simply means to proceed to statement number s. The GO TO statement may also be used for branching to one of several places in a program depending upon the integer value of a test location i. The form of such a statement would be GO TO $(s_1, s_2, s_3, \ldots, s_n)$, i. $s_1, s_2, s_3, \ldots, s_n$ are statement numbers and i is a nonsubscripted interger variable which is between 1 and n. The statement GO TO (30, 42, 50, 9),K means that control is transferred to the statement labeled 30, 42, 50, or 9 depending upon whether the value of K is 1, 2, 3, or 4 respectively.

Data for operations to be performed must be read into the computer and the result obtained from the computer. READ and PRINT statements are used for this purpose. Both statements specify the items to be read or printed and refer to a FORMAT statement which tells how the information to be read or printed is arranged. READ and PRINT statements have the form READ f,l or PRINT f,l which means read or print the list l having the arrangement described in the FORMAT declaration whose lable is f.

The information to be printed or read is known as a unit input or output record. Each record is made up of one or more fields. A field is a group of one or more columns whose contents must be described separately. The format code describes each successive field, specifying its form, size, and location from left to right within each record. The form of the number to be read or printed is designated by a code. I representing integer, E representing a floating point number with an exponent, and F representing a floating point number without an exponent. A field specification for an I- , E- , or F-field consists of one of the letters I, E, or F and an integer indicating the size of the field. In an E- or F-field an additional code digit is used to denote the placement of the decimal point. For example, I3 denotes a three-column integer field and F10.3 denotes a ten-column F-field with the decimal point three places to the left of the right end of the field. On E and F input fields, the decimal point does not have to be punched on the data cards but may be designated by the field specification. If the decimal point is punched, it overrides the placement of the decimal given by the field specifications. For the program:

```
   PRINT 60, A,B,C,J,K
60 FORMAT (3F12.4, 2I7)
```

The information A,B,C,J,K would be printed:

```
xxxx.xxxx    xxxx.xxxx    xxxx.xxxx    xxx    xxx
xxxx.xxxx    xxxx.xxxx    xxxx.xxxx    xxx    xxx
xxxx.xxxx    xxxx.xxxx    xxxx.xxxx    xxx    xxx
```

The format is repeated for each line until all the information included in A,B,C,J,K has been printed. Spacing between the numbers is allowed for by setting the field at a greater number than the digits to be printed in that field. Spacing may also be accomplished by the use of a skip field. A skip field is included in the format code by use of the specification mX, where

nn represents the number of columns to be skipped. Therefore, the preceding format declaration would be equalivant to: 60 FORMAT (4X, F8.4, 4X, F8.4 4X, F8.4, 4X, 2I3, 4X, 2I3).

In printing an output record, the printer must be told on which line of paper the record is to be printed. The printer receives the carriage-control information in the form of a character code placed in the first character position of the line. Therefore, the first character of a record is never printed but is assumed to be the carriage-code control. A blank in the first character position is the code for single-line spacing. The carriage-code for double spacing is zero. A diagonal (/) is used within a format declaration to denote the end of a record and means to advance the paper one line. Multiple diagonals may be used to advance the paper any number of desired lines.

Alphanumeric words and phrases in the form of comments, titles, and headings may be accomplished by using a Hollerith or H-field. The H-field is made a part of the format code for the given output record. An H-field consists of a space count, denoting the number of spaces needed for the message; the letter H, which identifies the field; and then the desired statement. The space count must include all blanks as well as characters. If the title "This is a trial program" is desired with two blank lines after it, the following would be used with (¤) denoting a blank space.

```
      READ 6
    6 FORMAT (25H¤THIS¤IS¤A¤TRIAL¤PROGRAM¤//)
```

The following is a program to determine if triangles with the lengths of sides stated are right triangles. The program allows for an error of .1 in the length of the sides.

```
      PRINT 4
    4 FORMAT (17H¤RIGHT¤TRIANGLES¤///)
    1 READ 6, A, B, C
      PRINT 7, A, B, C
```

```
    IF (ABSF (A * A + B * B - C * C) - .1) 20, 5, 5
  5 IF (ABSF (B * B + C * C - A * A) - .1) 20, 10, 10
 10 IF (ABSF (C * C + A * A - B * B) - .1) 20, 15, 15
 15 PRINT 8
    GO TO 1
 20 PRINT 9
    GO TO 1
  6 FORMAT (3F10.5)
  7 FORMAT (1X, 3F10.5)
  8 FORMAT (29H□THIS□IS□NOT□A□RIGHT□TRIANGLE//)
  9 FORMAT (25H□THIS□IS□A□RIGHT□TRIANGLE//)
    END
```

DATA

3.	4.	5.
4.	3.	5.
5.	3.	4.
5.1	3.1	3.9
5.1	3.03	4.1
8.9	4.25	1.4

This program instructs the computer to first print the heading "Right Triangles" and go down three lines. Read the length of the sides A, B, C. Print the lengths on the result sheet. Test to see if the absolute value of $A^2 + B^2 - C^2$ substracted from .1 is less than, equal to, or greater than one. If the value is less than 0, print "This is a right triangle," move the sheet up two lines and return to statement one. If the result is greater than or equal to 0, test to see if $B^2 + C^2 - A^2$ is less than .1 or $C^2 + A^2 - B^2$ is less than .1. If the result of the operations are less than zero, proceed as before. If the result cannot be found to be less than zero after trying all the possibilities, print "This is not a right triangle," move the paper up two lines, and proceed to statement one. When all values have been tested it is the end of the program.

The output of the preceding program would appear as:

```
RIGHT TRIANGLES

   3.00000   4.00000   5.00000
THIS IS A RIGHT TRIANGLE
```

```
  4.00000  3.00000  5.00000
THIS IS A RIGHT TRIANGLE

  5.00000  3.00000  4.00000
THIS IS A RIGHT TRIANGLE

  5.10000  3.10000  3.90000
THIS IS NOT A RIGHT TRIANGLE

  5.10000  3.03000  4.10000
THIS IS A RIGHT TRIANGLE

  8.90000  4.25000  1.40000
THIS IS NOT A RIGHT TRIANGLE
```

BIBLIOGRAPHY

Laurie, Edward J. Computers and How They Work. Burlingame, California: South-Western Publishing Company, 1963.

Organick, Elliott I. A Fortran Primer. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1963.

Saxon, James A., Herman S. Englander, and William R. Englander. System 360 Programming. Englewood Cliffs, New Jersey: Printice-Hall Inc., 1968.