UNCLASSIFIED

EXTERNAL TRANSMITTAL AUTHORIZED

~~FOR INTERNAL USE ONLY~~

ORNL

Central Files Number
57-6-68 (2d Rev.)

MASTER

ORACLE
MANUAL FOR PROGRAMMERS

Susie E. Atta

NOTICE

This document contains information of a preliminary nature and was prepared primarily for internal use at the Oak Ridge National Laboratory. It is subject to revision or correction and therefore does not represent a final report.

OAK RIDGE NATIONAL LABORATORY

OPERATED BY

UNION CARBIDE NUCLEAR COMPANY

Division of Union Carbide Corporation

POST OFFICE BOX X • OAK RIDGE, TENNESSEE

UNCLASSIFIED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UNCLASSIFIED

For Internal Use Only

ORNL

Central Files Number

57-6-68

Second Revision

Copy No. _____

ORACLE

Manual for Programmers

Susie E. Atta

January 22, 1958

Distribution

1-155. A. S. Householder
156-159. Laboratory Records
160. ORNL-RC

OAK RIDGE NATIONAL LABORATORY
Operated by
UNION CARBIDE NUCLEAR COMPANY
A Division of Union Carbide and Carbon Corporation
Post Office Box X
Oak Ridge, Tennessee

UNCLASSIFIED

UNCLASSIFIED

1

ACKNOWLEDGMENTS

I am grateful to Nancy M. Dismuke, who suggested the need for the writing of such material and offered many invaluable comments and suggestions. The engineers, W. J. Gerhard, R. J. Klein, J. W. Reynolds, C. H. Schalbe, and J. W. Woody, provided me with drawings, information, and many helpful suggestions concerning the engineering aspects of the ORACLE. Finally, I wish to acknowledge Mary Cline, Wilma Campbell and Helen Daggett for their secretarial help in the preparation of this manuscript.

UNCLASSIFIED

TABLE OF CONTENTS

<u>Chapter</u>	<u>Page</u>
I. INTRODUCTION	1
II. A DESCRIPTION OF THE ORACLE	3
A. General Characteristics	3
B. Description of the Seven Units	5
1. Internal Storage	5
2. External or Auxiliary Storage	7
3. The Arithmetic Unit	8
4. The Control Unit	8
5. Input-Output	9
6. Console	10
7. Power Supply	11
III. THE NUMBER SYSTEMS	14
A. Binary Numbers	14
B. Conversion of Decimal Numbers to Binary Numbers	16
1. Integers	16
2. Fractions	16
C. Conversion of Binary Numbers to Decimal Numbers	17
D. Binary Number Representation in the ORACLE	18
1. Word	18
2. Address	18
3. Digital	18
4. Fixed Point	19
E. Negative Numbers	19
F. Hexadecimal Numbers	21

<u>Chapter</u>	<u>Page</u>
IV. BASIC OPERATIONS OF THE ORACLE	24
A. Instruction Word	24
B. Arithmetical Operations of the ORACLE	26
1. Summation	26
a. Overflow	26
b. Addition	27
c. Subtraction	31
2. Shifting	34
3. Multiplication	37
4. Division	41
C. Logical Operations of the ORACLE	48
1. Transfer Operations	48
2. Substitutions	51
3. Stop Operations	54
a. Stop Instruction	54
b. Breakpoint Stop	54
D. Input-Output Operations	56
1. Magnetic Tapes	57
a. Auxiliary Storage	57
(1) Read or Write Instructions	59
(2) Hunt Instruction	60
(3) Rewind Instruction	61
b. Primary Output	63

<u>Chapter</u>	<u>Page</u>
2. Paper Tapes	63
a. ORACLE General Code	64
b. Primary Input	70
c. Primary Output	70
3. Computer Console Keyboard	72
a. Manual Instructions to the Computer	72
b. Secondary Input	73
4. Computer Console Typewriter	77
a. Secondary Output	77
b. Console Monitor	77
5. Input-Output Instructions	79
a. Hexadecimal Information	79
b. Alphameric Information	79
c. Instructions	81
6. The Photographic Curve Plotter for Output	84
a. Plotting Curves	84
(1) Translating the Coordinate Axes	87
(2) Scaling the Coordinates	88
b. Plotting Digits and Characters	89
c. Advancing the Film	95
E. Approximate Speeds of the Operations	97
F. Summary of Operations	98
BIBLIOGRAPHY	102

I. INTRODUCTION

Progress in arithmetical computations has been achieved through the years by man's desire to economize time, to eliminate human error and to promote scientific advancement. The abacus, the slide rule, the desk calculator, and finally the differential analyzer are all forerunners of the present day automatic, electronic computing machines.

History of these computing aids as well as the present day computing giant reveals that progress has been made chiefly by the interchange of ideas and many years of experimentation. The first recorded idea of an automatic computing device is credited to J. H. Müller [5], a Hessian military engineer, in 1786. His idea of a "difference engine" remained a theory until the beginning of the nineteenth century when Charles Babbage [4], an English mathematician, attempted to build the first automatic computing machine. His machine was never completed nor were his ideas understood and appreciated until the twentieth century.

The twentieth century brought an evolution of advanced mechanical and electrical engineering and advanced mathematical numerical methods as well as two world wars. It was not until 1942 during World War II that the limitations of the analyzer type machine were realized. The magnitude of computations needed for ballistic work and various other problems caused the Army and Navy to look for additional computing facilities. About this time the first automatic sequence controlled computer, "Mark I" [3,13], had been completed at Harvard. The Navy contracted with Harvard for the use of Mark I. The Army contracted with the Moore School of Engineering at the

University of Pennsylvania for the first high-speed electronic computer, "ENIAC", Electronic Numerical Integrator and Calculator [2]. This machine was designed and built by several different people including von Neumann Eckert, and Mauchly.

Mark I and ENIAC were the beginning of many strides taken in the field of research and development on high-speed electronic computers during the past ten years. In 1947, the United States Navy and Harvard University [16] sponsored together the first big conference on large-scale digital calculating machinery. At this conference many interested people, including scientists from Oak Ridge, interchanged their ideas in the design, construction, and application of computing machinery.

In 1949 [15] the U. S. Atomic Energy Commission with the assistance of Oak Ridge Mathematicians began to make plans for acquiring a high-speed computer in Oak Ridge.

In 1951 [19] Argonne National Laboratory contracted with Oak Ridge National Laboratory for the construction of the "ORACLE", Oak Ridge Automatic Computer and Logical Engine. Four Oak Ridge National Laboratory engineers, R. J. Klein, W. J. Gerhard, J. W. Woody and E. W. Burdette, assisted in the design and construction of the ORACLE which was carried on at the Argonne National Laboratory.

The ORACLE was installed at Oak Ridge National Laboratory in 1953 [7] and has been in operation since that time.

PHOTO 42113
UNCLASSIFIED

II. A DESCRIPTION OF THE ORACLE

A. GENERAL CHARACTERISTICS

Some of the general features and characteristics⁽¹⁾ of the ORACLE can be described by such adjectives as general purpose, digital, high-speed and automatic.

The ORACLE is a "general purpose" rather than a "special purpose" computer. This means that practically any problem which is capable of a mathematical formulation can be successfully solved on the ORACLE. The "differential analyzer" [8] is an example of a special purpose machine. It was designed to evaluate the solutions of differential equations by the use of scale readings or measurements which may vary with the temperature and humidity. Such solutions can be depended upon for only a limited amount of accuracy.

A digital machine such as the ORACLE treats numbers as discrete quantities. A number, N , is treated as a set of digits which satisfies the inequalities

$$-1 \leq N < 1.$$

The number, N , is called a "digital" number. The precision is limited only by the number of digits used in the set to represent the number N . As an example, let N equal one-third; then

$$N = .3333333333$$

(1) Oral lectures presented by N. M. Dismuke to a class of beginning programmers provided some ideas and information for this work.

is more precise than

$$N = .3333.$$

The high speeds of operation gives electronic computers an economical advantage over other computing devices. The ORACLE performs an addition or subtraction on a twelve-decimal digit number in seventy microseconds. A multiplication or division is executed in four hundred forty microseconds. These high speeds of operation would be useless if a human being were needed at the computer to give instructions by pushing buttons from one operation to the next.

A computer is automatic in the sense that it can carry out a long sequence of instructions without human intervention. The automatic control system enables the modern computer to perform arithmetical and logical operations, and to read, write, compare, select, and copy information according to a plan of instructions. The entire program or plan of instructions for solving a problem can be written into a language or code understood by the computer before going into the computer room. When the program has been read into the machine and the "go" or "start" button is pushed, the answers are given out as soon as they are computed. Such features have caused a widespread curiosity concerning the "machine that thinks" or the "electronic brain". It should be emphasized that the machine must be told what to do and how to do it. In other words, the ORACLE can do only what it is told to do in its own language.

B. DESCRIPTION OF THE SEVEN UNITS

The ORACLE, like most automatic digital computers, is composed of several units. A description of the ORACLE is given by a brief discussion of its seven units.

1. Internal Storage

An automatic computer must be able to store the program and initial conditions for solving a problem. It must also be able to remember or store intermediate results of a calculation until needed at some later stage. This part of the machine is called the "storage" or "memory". Most high-speed digital computers (through 1956) use one of three types of internal storages [1, 16, 20] :

1. magnetic core
2. mercury delay line
3. electrostatic or cathode-ray tube.

The type of storage used is an important factor in determining the speed, reliability, storage capacity, and construction cost of a computer.

The ORACLE has an electrostatic internal storage. A direct connection between this storage and the arithmetic unit permits parallel, and hence rapid, transfer of information between these two units. Random access to ORACLE internal storage is possible, that is, access to any storage location is independent of its relative position.

There are many disadvantages in the use of the electrostatic storage. Frequent regeneration is necessary to preserve the information stored on the cathode-ray tube. Reference to neighboring locations can change or distort

the information in a particular location, a problem known as "read around effect". Screen blemishes can also effect the accurate storage of information and finally, a power failure will destroy the information in storage.

Designers of computing machines found that a digit could be represented best by two states or conditions.. Some common examples are: an electric circuit is either on or off; a particular location on paper tape either has a hole or does not; a particular location on a magnetic tape is either magnetized or not. Many designers, therefore, turned to the number system which has only two digits "0" and "1". This is the binary or base two system.

The cathode-ray tube is used in the electrostatic storage of the ORACLE. Spots on the faces of these tubes record electrical signals in the form of a pattern of positive and negative charges [12]. Different

Cathode-Ray Tube

charge patterns are used to represent a "1" and a "0"⁽²⁾. Each tube displays these charge patterns as faint and bright spots in the form of a 32 by 32 square array making possible the storage of 1024 binary digits. In the figure on preceding page let i and j represent the row and column indices, respectively, of a binary digit. The ORACLE allows a set of 40 binary digits to represent a number. Thus 40 cathode-ray tubes are needed to represent 1024 numbers or storage registers. The ORACLE uses 80 tubes to give a storage capacity of 2048 registers.

2. External or Auxiliary Storage

Any device capable of copying information both to and from an internal storage automatically and at a rapid speed may be regarded as an auxiliary storage. The ORACLE uses magnetic tapes for auxiliary storage. Information in the auxiliary storage is not directly available to the arithmetic unit. The information must be first located on the tape and then read into the electrostatic storage before it may be used by the arithmetic unit. For this reason, the electrostatic storage is sometimes called the fast storage and the magnetic tape the slow storage. Information stored on tapes may be used at a later stage in the calculation procedure or the tapes may be removed from the calculator and filed for further reference. The magnetic tape unit has four tape drives. Each drive contains a large reel of two-inch wide tape. The tape is capable of moving either backward or forward.

(2) Information received from W. J. Gerhard.

3. The Arithmetic Unit

The arithmetic unit is that part of the computer in which all arithmetical and logical operations are executed. The unit [7] is composed of:

1. three registers:
 - a. accumulator register, A
 - b. quotient register, Q
 - c. storage register, S
2. the complement gates, C. G.
3. the adder or plusser, P.

The three registers are used to hold or store temporarily the numbers brought from the electrostatic storage for arithmetical operations. The A and Q registers are each two parallel registers, designated by A^U , A_L and Q^U , Q_L . These are shifting registers. Information contained in A^U and Q^U can be shifted right or left as it is sent to A_L and Q_L . The functions of the registers during arithmetical operations are described in Chapter IV.

4. The Control Unit

The control unit [18] directs the automatic operations of the internal and external storages, the arithmetic unit, and the input-output units. It consists of a control section to control each of the above four units, together with a central agency, the MOSC, Memory Order Sequence Control, to keep all units properly synchronized. The storage register S_L , which is associated in physical construction with the arithmetic storage register S^U , is used to store the order or instruction currently being executed. The

control counter, C. C., keeps track of the storage location of the next instruction. The dynamic programmer, static programmer, and shift counter are all part of the arithmetic control. The dispatch counter, D. C., and the pulsers make up the internal storage control.

5. Input-Output

The input-output system enables us to communicate with the computer. The media used are paper tape (7/8" wide), two types of magnetic tape (2" wide and 5/8" wide), the console keyboard and typewriter, and the photographic curve plotter. This system is composed of units connected to the ORACLE and controlled by the ORACLE as well as peripheral equipment used for processing the tapes.

The input-output equipment controlled by the computer includes:

1. The Ferranti photoelectric paper tape reader
2. The sixty character per second paper tape punch
3. The magnetic tape recorder for 5/8" magnetic tape
4. Curve plotter
5. Computer console keyboard
6. Computer console typewriter
7. Four magnetic tape drives. The two-inch magnetic tape may be thought of as an input and output device. However it must be remembered that the only way to read or record information on this tape is through the computer.

The peripheral equipment used for tape preparation and printing includes:

1. Tape preparation units for preparing the paper tape as input
2. Printing units for printing the output on paper tape and magnetic tape (5/8" wide)
3. The magnetic tape processing unit used to detect and mark flaws and impurities on the 5/8" magnetic tape.

The instructions or program together with the initial numbers and parameters are punched on paper tape, by means of the preparation unit. The information on the paper tape is read into the electrostatic storage by means of the photoelectric reader. The results of the computation can be photographed in curve plots or digits; they can be stored on magnetic tape to be used later by the computer; or they can be recorded on magnetic tape or punched on paper tape to be printed at a later time on the printing unit.

6. Console

The console is centrally located in front of the other computer units. It is surrounded by the Ferranti reader, the punch, photographic curve plotter, magnetic tape recorder, console keyboard and typewriter.

The console contains all of the necessary keys for directing and monitoring all computer operations. The console keyboard consists of a control key panel and an information key panel. A console typewriter acts as a monitor to the console keyboard. A symbol or character which is associated with each key is printed each time that key is depressed. There are various indicator lights to facilitate visual monitoring of the ORACLE.

The contents of the registers are displayed directly on the console by small tubes. A tube in each bit position represents a 1 by a light on, and represents a 0 by a light off. The instruction currently being executed by the computer is displayed in the S_L register, while the location of the next instruction is displayed in the control counter. Two of the cathode-ray tubes used in the electrostatic storage are also displayed on the console. Any tube or digit position may be displayed. A Memotron is provided in the console to act as a monitor to the photographic curve plotter. Such a display of information provides not only an attraction to the console but also assists the programmer in following the flow of computations in his problem as it is being solved on the computer.

7. Power Supply

The electrical power needed by the ORACLE is provided and controlled by this unit.

ORACLE System Block Diagram.

FIGURE 1

ORACLE Block Diagram Information Flow.

FIGURE 2

III. THE NUMBER SYSTEMS

A. BINARY NUMBERS

In most cases the binary or base two number system has been found more adaptable to automatic computations. This is due to the common two-state conditions characteristic of most electronic equipment as well as the simplicity of binary arithmetic. The binary system has only two digits 0, and 1, whereas the decimal system has ten digits, 0 through 9. The decimal digits are ordered in weighted sequence with powers of 10. The integers are represented in positive powers of ten and the fractions in negative powers of ten. That is,

$$2.625 = 2 \cdot 10^0 + 6 \cdot 10^{-1} + 2 \cdot 10^{-2} + 5 \cdot 10^{-3}.$$

Now write

$$2.625 = 2 + .5 + .125;$$

or write in powers of two

$$2.625 = 2 + 1/2 + 1/8.$$

A binary number is written with the two digits 0 and 1 ordered in weighted sequence by powers of two. The integers are represented in positive powers of two and fractions are represented in negative powers of two.

Write the decimal number 2.625 as a binary number.

$$2.625 = 2 + 1/2 + 1/8 = 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

$$2.625 \text{ (decimal)} = 10.101 \text{ (binary)}.$$

The radix point, decimal for the left hand side and binary for the right hand side, separates the positive and negative powers of the base.

The base may be indicated by enclosing the number in parentheses and writing the base as a subscript. Example:

$$(2.625)_{10} = (10.101)_2$$

The first few binary integers, together with their decimal equivalent are given in the table below.

<u>Binary</u>	<u>Decimal</u>
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

An addition or multiplication in the binary system is very simple.

The rules of addition and multiplication are summarized in the tables below.

<u>Addition</u>	<u>Multiplication</u>
$0 + 0 = 0$	$0 \cdot 0 = 0$
$0 + 1 = 1$	$0 \cdot 1 = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$
$1 + 1 = 0 \text{ and } 1 \text{ to carry}$	$1 \cdot 1 = 1$

B. CONVERSION OF DECIMAL NUMBERS TO BINARY NUMBERS

There are several methods for converting a number from one base to another. The methods described below are for converting integers and fractions separately.

1. Integers. A simple way to convert an integer from one base to another is by the use of the remainder after repeated divisions by the desired base. That is, if one desires to convert a decimal integer to a binary integer, then proceed as in the example given.

Example: Convert 185 (decimal) to binary

Divisions	Quotient	Remainder	Binary Digits
$185 \div 2$	92	1	1.
$92 \div 2$	46	0	01.
$46 \div 2$	23	0	001.
$23 \div 2$	11	1	1001.
$11 \div 2$	5	1	11001.
$5 \div 2$	2	1	111001.
$2 \div 2$	1	0	0111001.
$1 \div 2$	0	1	10111001.

$$(185)_{10} = (10111001)_2$$

2. Fractions. The conversion of a fraction from one base to another may be handled similarly by the use of repeated multiplications by the desired base, instead of divisions.

Example: Convert .827 (decimal) to binary.

Multiplication	Product		Binary Digits
	Integer	Fraction	
.827 x 2	1	+ .654	.1
.654 x 2	1	+ .308	.11
.308 x 2	0	+ .616	.110
.616 x 2	1	+ .232	.1101
.232 x 2	0	+ .464	.11010
.464 x 2	0	+ .928	.110100...
.			
.			
.			

Then .827 (decimal) equals .110100... (binary). It should be observed that a fraction expressed as a finite number of decimal digits need not have a finite representation in the binary system. A three decimal digit number requires about eleven binary digits ($3 \log_2 10$) to give approximately the same precision.

C. CONVERSION OF BINARY NUMBERS TO DECIMAL NUMBERS

The same methods as described above may be used to convert a number from base 2 to base 10. All arithmetic must now be done in binary and the binary representation of decimal ten must be used. Therefore the decimal digits will be developed in the binary form which must be converted to decimal.

Example. Convert $(10111001)_2$ to decimal.

$$(10)_{10} = (1010)_2$$

Divisions	Quotient	Remainder	Decimal Digits
$10111001 \div 1010$	10010	$(101)_2 = (5)_{10}$	5.
$10010 \div 1010$	1	$(1001)_2 = (8)_{10}$	85.
$1 \div 1010$	0	$(1)_2 = (1)_{10}$	185.

$$(10111001)_2 = (185)_{10}.$$

D. BINARY NUMBER REPRESENTATION IN THE ORACLE

A binary number is represented in the storage cell of the ORACLE by a set of 40 cathode-ray tubes, numbered from 0 to 39. Each tube is able to represent one digit for 1024 numbers. Tube 0 represents the most significant digit, tube 1 the next most significant digit, ..., finally tube 39 represents the least significant digit. Let a_0 be the digit on tube 0, a_1 on tube 1, ..., a_{39} on tube 39; then the 40 binary digits are $a_0 a_1 a_2 \dots a_{39}$. Such a set of 40 binary digits of information is called a word. Each of the 40 tubes stores one of the 40 digits in a corresponding position. The number which specifies the location of the cell or position of the digits on the tubes is called the address. There are 1024 locations per set of 40 tubes. It has already been stated that there are two sets of 40 tubes, which give 2048 locations or addresses in the storage of the ORACLE.

The binary number a , represented by the digits $(a_0 a_1 a_2 \dots a_{39})$, must be digital; that is, it must satisfy the inequalities

$$-1 \leq a < 1.$$

Hence -1 is digital, but +1 is not. Define the digital number as

$$a = -a_0 2^0 + a_1 2^{-1} + \dots + a_{39} 2^{-39}.$$

If a_0 equals 1 and $a_1, a_2, \dots, a_{39}$ equals 0, then a equals -1. If a_0 equals 0 and $a_1, a_2, \dots, a_{39}$ equals 1, then a equals $1 - 2^{-39}$, which is the largest positive digital number. A digital number satisfies the inequalities

$$-1 \leq a \leq 1 - 2^{-39}.$$

If a_0 is 1, then a is a negative number; if a_0 is 0, then a is positive or zero. Hence a_0 is called the sign digit. The binary point lies between positions 0 and 1 in all numbers a ; thus, the ORACLE is a fixed point machine.

$$a_0 . a_1 a_2 a_3 \dots a_{39}$$

$$\text{Example: } 3/4 = 0.1100 \dots 0$$

$$-1 = 1.0000 \dots 0$$

E. NEGATIVE NUMBERS

Negative numbers are represented in the ORACLE as complements with respect to two. This method is sometimes called a two's complement. The two's complement works on the same principle as the ten's complement. When two positive quantities are added together and produce a power of ten, one is said to be the ten's complement of the other. For example, the ten's complement of 613 is 387, since

$$613 + 387 = 1000 = 10^3.$$

The ten's complement of 54 is 46, since

$$54 + 46 = 100 = 10^2.$$

The ten's complement of a number can be obtained by subtracting the number from a next higher power of ten. The complement of 613 is obtained by

$$1000 - 613 = 387;$$

or the complement of 54 is obtained by

$$100 - 54 = 46.$$

When two positive quantities of the base two system are added together and produce a power of two, one is said to be the two's complement of the other. Of course, a binary digit has a value of either 0 or 1. The two's complement of 1001 is 0111, since

$$1001 + 0111 = 10000 = 2^4.$$

The two's complement of 11 is 01, since

$$11 + 01 = 100 = 2^2.$$

The two's complement of a number can be obtained by subtracting the number from a next higher power of two. The complement of 1001 is obtained by

$$10000 - 1001 = 0111;$$

or the complement of 11 is obtained by

$$100 - 11 = 01.$$

A simpler way to obtain a two's complement of a number is to replace all 0 digits with 1, all 1 digits with 0, and then add a unit in the least significant digit position. The two's complement of 1001 is

$$\begin{array}{r} 0110 \\ + 0001 \\ \hline 0111 \end{array}$$

As another example, represent $-7/8$ in the A-register.

$$\begin{array}{r} a_0 \cdot a_1 a_2 a_3 a_4 \dots a_{39} \\ +7/8 = 0.1110 \dots 0 \end{array}$$

Ones replaced by zeros, zeros replaced by ones and then addition of 2^{-39} gives

$$\begin{array}{r} a_0 \cdot a_1 a_2 a_3 a_4 \dots a_{39} \\ 1.00001 \dots 1 \\ \hline 0.00000 \dots 01 \\ -7/8 = 1.00010 \dots 0 \end{array}$$

F. HEXADECIMAL NUMBERS

Although there are many good reasons for using the binary number system in high-speed electronic machines, there are some disadvantages to the handling of forty-binary digits for representing a number. So many digits are not only cumbersome but lead to errors. The input-output units are not easily adaptable to the binary system.

These difficulties are overcome by treating binary digits in groups of 3 or 4. A number for the ORACLE is represented by groups of four binary digits. This number system which is of base 2^4 or 16 is commonly called hexadecimal or hex. The input-output equipment is designed to accept groups of 4 binary digits. Hence all binary information including the instructions is written in the hexadecimal notation for insertion into the computer. For the sixteen digits required in this system, the decimal digits 0 through 9 plus A, B, C, D, E, and F, are used. The table below shows the correspondence between the hexadecimal digit and the group of four binary digits it represents.

<u>Four Binary Digits</u>	<u>Hex Digits</u>	<u>Decimal Digits</u>
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

A forty-digit binary word can now be written as a ten-digit hex word. The conversion from decimal to hex can be made by the method similar to the one used to convert from decimal to binary. However, it must be remembered that the first binary digit of a word is used as the sign digit. Therefore, the first group of four binary digits of a word contains the sign digit plus a group of three binary digits. But a group of three binary digits represents base 2^3 or 8. For this reason the first digit in a hex word is really a digit to the base 8. It can be seen from the above table that if the first digit of a hex word is larger than 7, the number is negative.

$$a_0 a_1 a_2 a_3 \dots a_{39}$$

$$-1 = 1.0 \ 0 \ 0 \ \dots \ 0 \quad (\text{binary})$$

$$-1 = 8000000000 \quad (\text{hex})$$

Example 1: Represent $-7/8$ as a hex word.

$$+7/8 = 70 \ 000 \ 00 \ 000$$

$$-7/8 = 90 \ 000 \ 00 \ 000$$

Example 2: Convert the decimal .3125 to hexadecimal.

$$.3125 \times 8 = 2 + .5000$$

$$.5000 \times 16 = 8 + .000000 \ 000 \ (\text{hex})$$

$$.3125 \ (\text{decimal}) = 28 \ 000 \ 00 \ 000 \ (\text{hex})$$

IV. BASIC OPERATIONS OF THE ORACLE [14]⁽³⁾

A. INSTRUCTION WORD

A word has been defined as the forty digits of information stored in a cell. The ORACLE interprets the information in a word either as a number to be involved in an arithmetical or logical operation or as an instruction. An instruction then is represented by a numerical quantity. The digits in an instruction word are divided into several groups to provide the necessary information, such as the addresses of the numbers to be involved in an operation and the code number corresponding to the particular operation to be performed. In the ORACLE, most of the arithmetical operations bring a word from a specified storage cell and do something with it in the arithmetic unit, or else they take a word from some part of the arithmetic unit and place it in a specified storage cell. Hence an instruction consists of the operation to be performed and the address of the operand.

Since there are 2048 storage cells or addresses in the internal storage of the ORACLE, the addresses are numbered from 0 to 2047 inclusive. The largest address number 2047 converted to hexadecimal is 7FF or represented in binary notation is 0111 1111 1111. Three hex digits are then required

(3) A large portion of this chapter has been rewritten from "Basic Operations of The ORACLE" by A. S. Householder, reference 14. A more thorough discussion of some parts, such as the arithmetic algorithms, may be found in reference 14.

to represent an address. Two hex digits are required to designate the operation code. Since only five hex digits are required for an instruction, a word is divided into two half-words. A word is then composed of a left-hand instruction and a right-hand instruction.

Instruction Word

2 hex digits or 8 binary digits	3 hex digits or 12 binary digits	2 hex digits or 8 binary digits	3 hex digits or 12 binary digits
Operation	Address	Operation	Address
Left-hand Instruction		Right-hand Instruction	

A number and an instruction are indistinguishable to the ORACLE. Normally, an instruction is sent by the control counter to S_L to be executed. A number is sent to the arithmetic unit by the instruction being executed. However, if an instruction is sent to the arithmetic unit, it can be operated on just like a number. This means that instructions may be changed by the machine during the course of computation. Addresses are commonly modified by addition or subtraction.

Since each instruction contains one address, it is called a single address code. A three address code means that each instruction contains three addresses with one operation symbol. For example, an addition instruction would contain the addresses of the two numbers to be added as well as the address of the resulting sum.

B. ARITHMETICAL OPERATIONS OF THE ORACLE

1. Summation

a. Overflow. Although two numbers a and b may be digital numbers, their sum may not be digital. For example, obtain the sum of two digital numbers a and b. Let

$$\begin{array}{r} a_0 \cdot a_1 a_2 a_3 \dots a_{39} \\ a = 3/4 = 0. 1 1 0 \dots 0 \\ b = 1/2 = 0. 1 0 0 \dots 0 \\ a + b = c = 5/4 \neq 1. 0 1 0 \dots 0 \end{array}$$

The sum gives a "1" in the sign position. Then 1.010 is a representation of

$$-1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3},$$

or

$$-1 + 1/4 = -3/4.$$

The true sum, $5/4$, is not a digital number and has therefore exceeded the capacity of the register or overflowed. This presents a problem since addition is basic to all operations. An overflow in addition must either be avoided or else detected by the machine. This could be avoided by always being assured in advance that the sum is digital; however, at times this may be very difficult. It is best to design the machine in such a way that it will detect a nondigital sum and take the appropriate steps. For this purpose, the A and P registers are provided with an additional element, placed at the left, which makes 41 instead of the 40 elements.

The extra element is sometimes called the x-toggle. The state of the x-toggle is designated by either a_x or a_{-1} . If $a_0, \dots, a_{39}$ are used for the states of the other elements, the numerical interpretation of the state of the entire register will be

$$a = -a_{-1}2 + a_02^0 + \dots + a_{39}2^{39}.$$

A number, a , representable in this form satisfies the inequalities

$$-2 \leq a \leq 2 - 2^{-39},$$

and will be called bidigital. If $a_{-1} = a_0$, then

$$-a_{-1}2 + a_0 = -a_0,$$

and a is digital. Hence a bidigital number is digital if and only if the first two digits are alike. The machine tests on the a_{-1} element in the A-register for the sign of the number.

b. Addition. The algorithm for addition is designed so that the sum of two digital numbers will be correctly formed modulo 4 in the A-register, whether the sum is digital or not.

Example 1:

$$\begin{aligned} 1 \pmod{4} &= 1 \\ &\left[\begin{array}{l} \text{because } 1 = 0(4) + 1 \\ \text{or } 1 - 0(4) = 1 \end{array} \right] \end{aligned}$$

Example 2:

$$\begin{aligned} 9 \pmod{4} &= 1 \\ &\left[\begin{array}{l} \text{because } 9 = 2(4) + 1 \\ \text{or } 9 - 2(4) = 1 \end{array} \right] \end{aligned}$$

But when a nondigital number is formed in A, that is, one for which $a_{-1} \neq a_0$, a special toggle called the "O.F.", or overflow toggle, is set in the machine. One form of the conditional transfer is to test whether or not the overflow toggle is set. The transfer instructions and the resetting of the overflow toggle will be discussed later. As an illustration, obtain the sum of two digital numbers a and b. Let

$$\begin{array}{r} a_{-1}, a_0, a_1, a_2, a_3, \dots, a_{39} \\ a = 3/4 = 0, 0.1 \ 1 \ 0 \ \dots \ 0 \\ b = 1/2 = 0, 0.1 \ 0 \ 0 \ \dots \ 0 \\ a + b = c = 5/4 = 0, 1.0 \ 1 \ 0 \ \dots \ 0. \end{array}$$

Since $a_{-1} \neq a_0$, the O.F. toggle is set. This means that the sum is not digital, but it is correctly formed mod 4. If $5/4$ is multiplied by $1/2$ which is equivalent to shifting all elements in the A-register one place to the right, a digital number, $5/8$, is obtained.

The above can be shown by the algorithm for addition. Let the two digital numbers,

$$a = -\alpha_{-1}2 + \alpha_02^0 + \alpha_12^{-1} + \dots + \alpha_{39}2^{-39}$$

and

$$b = -\beta_{-1}2 + \beta_02^0 + \beta_12^{-1} + \dots + \beta_{39}2^{-39},$$

be added to give the sum,

$$c = -\gamma_{-1}2 + \gamma_02^0 + \gamma_12^{-1} + \dots + \gamma_{39}2^{-39}.$$

The algorithm [14] performed by the plusser is the following:

$$\begin{array}{rcl}
 \alpha_{39} 2^{-39} + \beta_{39} 2^{-39} & = & \gamma_{39} 2^{-39} + \cancel{\gamma'_{38} 2^{-38}} \\
 \alpha_{38} 2^{-38} + \beta_{38} 2^{-38} + \cancel{\gamma'_{38} 2^{-38}} & = & \gamma_{38} 2^{-38} + \cancel{\gamma'_{37} 2^{-37}} \\
 & \dots & \\
 \alpha_0 2^0 + \beta_0 2^0 + \cancel{\gamma'_{-1} 2^{-1}} & = & \gamma_0 2^0 + \cancel{\gamma'_{-1} 2^{-1}} \\
 \hline
 -\alpha_{-1} 2^{-1} - \beta_{-1} 2^{-1} + \cancel{\gamma'_{-1} 2^{-1}} & = & -\gamma_{-1} 2^{-1} + \gamma'_{-2} 2^{-2} \\
 \hline
 a + b & = & c + 4\gamma'_{-2}
 \end{array}$$

where the digits γ' are the "carries" digits and do not appear in the final results. When the equations are summed, the "carries" on one side of the equations cancel the "carries" on the other side of the equations with the exception of the last "carry". The sum of the above equations gives

$$a + b = c + 4\gamma'_{-2}.$$

Hence, if c is not the true sum of a and b , it can differ from the true sum only by ± 4 . When a and b are digital, they satisfy the inequalities

$$-1 \leq a < 1$$

$$-1 \leq b < 1$$

and the sum satisfies the inequalities

$$-2 \leq a + b < 2.$$

Hence, c is the true sum of $a + b$ whether or not the O.F. toggle is set.

Example: Let $a = -1$

$b = -1$

$a_{-1}, a_0, a_1 \dots a_{39}$

$a = 1, 1, 0 \dots 0$

$b = 1, 1, 0 \dots 0$

$a + b = c = 1, 0, 0 \dots 0 = -2$

The O.F. toggle is set. The sum, -2, is not digital but it is the true representation of c.

Now suppose that a is bidigital,

$$-2 \leq a < 2.$$

However, b is certainly digital since it must come from the storage.

Suppose the O.F. toggle has been cleared prior to the operation. If the operation $a + b$ does not set the O.F. toggle, then c is a digital true sum.

If the operation $a + b$ sets the O.F. toggle, and $\alpha_{-1} = \gamma_{-1}$ (sign $a = \text{sign } c$), then c is still the true sum of a and b. If the O.F. toggle is set, and $\alpha_{-1} \neq \gamma_{-1}$ (sign $a \neq \text{sign } c$), then

$$a + b = c + 4(\alpha_0 - \alpha_{-1}).$$

Example: Let $a = -2$

$b = -1$

$a_{-1}, a_0, a_1 \dots a_{39}$

$a = -2 = 1, 0, 0 \dots 0$

$b = -1 = 1, 1, 0 \dots 0$

$a + b = c = 0, 1, 0 \dots 0 = +1.$

Since the O.F. toggle is set and $\alpha_{-1} \neq \gamma_{-1}$

$$\begin{aligned} a + b &= c + 4(0 - 1) \\ &= c - 4 = 1 - 4 = -3. \end{aligned}$$

In normal operations, nondigital numbers should be avoided. When there is doubt about the digital character of any arithmetical result that is to appear in A, an overflow test should be introduced and a correction routine provided. However, cases may arise when it would be a great advantage to use the full flexibility of the A-register.

c. Subtraction. The ORACLE performs a subtraction by first complementing the subtrahend, and then adding this complement to the minuend. The carry resulting from the sum of the most significant digits is ignored. For example, perform

$$\begin{aligned} 3/4 - 1/2 &= 1/4. \\ 3/4 &= 0, 0. 110 0 \dots 0 \\ -1/2 &= 1, 1. 100 0 \dots 0 \\ 3/4 - 1/2 &= 1/4 = 0, 0. 010 0 \dots 0. \end{aligned}$$

An example of subtraction in the decimal system by using complements is as follows:

$$589 - 267 = 322.$$

The complement of 267 is 733, then

$$589 + 733 = 322.$$

Complementation is executed in the ORACLE by a circuit called a complement gate, C.G. The complement gate is actuated by a signal from the control unit and is capable of supplying either the number unchanged, or its digit-wise complement. Addition is carried out by the plusser.

The functions of the A, Q, and S registers in an addition or a subtraction are indicated in the table below.

Operation	Registers		
	A	Q	S
Addition	Augend sum		Addend
Subtraction	Minuend difference		Subtrahend

Before an addition or subtraction instruction is given, the augend or minuend must be placed in the A register. During execution of the addition or subtraction instruction, the addend or subtrahend is transferred from a specified storage location to the storage register. The number is then sent to the plusser by way of the complement gates where it is either left unchanged or complemented. The plusser forms the sum or difference of the A and S registers. The sum or difference is transferred to A, replacing the augend or minuend.

There are altogether, eight summation operations. The hex code, the symbol and the explanations of the operations are given in the table below. Assign the number, b, as the number to be operated upon, and M(b) as the address of b.

SUMMATION INSTRUCTIONS

Code	Symbol	Operation
20 M(b)	H M(b)	Hold the contents of the A-register and add b.
21 M(b)	Hm M(b)	" " " " " " " " the magnitude of b.
22 M(b)	H- M(b)	Hold the contents of the A-register and add the negative of b.
23 M(b)	H-m M(b)	Hold the contents of the A-register and add the negative of the magnitude of b.
24 M(b)	C M(b)	Clear the A-register to zeros and copy b into the A-register.
25 M(b)	Cm M(b)	Clear the A-register to zeros and copy the magnitude of b into the A-register.
26 M(b)	C- M(b)	Clear the A-register to zeros and copy the negative of b into the A-register.
27 M(b)	C-m M(b)	Clear the A-register to zeros and copy the negative of the magnitude of b into the A-register.

As an illustration, write an instruction word to subtract the number, b, located in cell 024 from the number, a, located in cell 12B.

Instruction Word				Remarks	
Operation	Address	Operation	Address	Left Instruction	Right Instruction
24	12B	22	024	Clear the A-register and add a	Hold a in the A-register and add -b
or					
26	024	20	12B	Clear the A-register and add -b	Hold -b in the A-register and add a

2. Shifting

Algorithms for both multiplication and division make use of the arithmetic shifts. The A and Q registers in the arithmetic unit are used for shifting a word a specified number of digits or positions to the right or left. A shift to the left is equivalent to multiplying by two; a shift to the right is equivalent to dividing by two or multiplying by one-half.

$$a_{-1}, a_0, a_1, a_2, a_3, a_4, \dots, a_{39}$$

$$a = 5/8 = 0, 0.1\ 0\ 1\ 0\ \dots\ 0$$

$$2^{-1}a = 5/16 = 0, 0.0\ 1\ 0\ 1\ 0\ \dots\ 0 \quad (\text{shifted one place to right})$$

$$2a = 5/4 = 0, 1.0\ 1\ 0\ 0\ \dots\ 0 \quad (\text{shifted one place to left})$$

(The overflow toggle will be set on the left shift.)

In order for the A and Q registers to move the digits to the right or left, the contents of A_L and Q_L are copied into A^U and Q^U . Each digit is then shifted one place according to the stored instructions as it is recopied from A^U and Q^U into A_L and Q_L .

A left shift of " ρ " places will set the overflow toggle if and only if $2^\rho a$ is non-digital. The logic for setting the O.F. toggle requires that the test, $a_{-1} \neq a_0$, be made before and after each shift. Therefore, if the number in A is non-digital but the O.F. toggle has been cleared by a transfer instruction, then a shift of " ρ " places, whether " ρ " is zero or not, will again set the O.F. toggle.

Example:

Let the A-register contain the following:

$$a_{-1}, a_0, a_1, a_2, \dots, a_{39}$$
$$0, 1, 1, 0, \dots, 0.$$

Then give instructions to clear the O.F. toggle and left shift one place. A now contains

$$a_{-1}, a_0, a_1, a_2, \dots, a_{39}$$
$$1, 1, 0, 0, \dots, 0.$$

The O.F. toggle will again be set, since $a_{-1} \neq a_0$ before the left shift was made.

A right shift will not set the overflow toggle except in the special cases given below.

1. If the number in A is non-digital but the O.F. toggle has been cleared by a transfer instruction, then a shift of " ρ " places, whether " ρ " is zero or not, will again set the O.F. toggle.
2. Any "Ro, ρ " shifts, whether " ρ " is zero or not, replaces the a_{-1} toggle with a zero. "Ro, 0" will set the O.F. toggle if $a_0 = 1$.

The number of places for a number to be shifted is designated in the address portion of a word. However, it is very important to remember that the number of places shifted is always modulo 64. The reason for this is that the shift counter has six binary positions and, hence, runs from 0 to $2^6 - 1$ or 63, inclusive. Normally, a shift of 40 or more places would not be ordered in A alone. Although, if the number of places is computed by the routine, the shift might turn out to be greater than 40 or 64.

There are eight different instructions each for shifting right and shifting left, making a total of sixteen shift instructions. These

instructions are demonstrated in the table below and the symbol code letters are defined.

SYMBOL		CODE		RESULT (Illustrated for $\rho = 2$)														
SHIFTS				A_{-1}	A_0	A_1	A_2	$\dots$	A_{37}	A_{38}	A_{39}	Q_0	Q_1	Q_2	Q_3	$\dots$	Q_{38}	Q_{39}
Ro	ρ	10	$\rho \uparrow$	0	0	0	α_0	$\dots$	α_{35}	α_{36}	α_{37}	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
Ror	ρ	11	$\rho \uparrow^{(4)}$	0	0	0	α_0	$\dots$	α_{35}	α_{36}	1	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
L	ρ	12	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
Ly	ρ	13	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	β_0	β_0	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
R	ρ	14	ρ	α_{-1}	α_{-1}	α_{-1}	α_0	$\dots$	α_{35}	α_{36}	α_{37}	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
Rr	ρ	15	ρ	α_{-1}	α_{-1}	α_{-1}	α_0	$\dots$	α_{35}	α_{36}	1	β_0	β_1	β_2	β_3	$\dots$	β_{38}	β_{39}
Lc	ρ	16	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	0	0	0	0	$\dots$	0	0
Lcy	ρ	17	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	0	0	0	0	$\dots$	0	0
Roq	ρ	18	$\rho \uparrow$	0	0	0	α_0	$\dots$	α_{35}	α_{36}	α_{37}	α_{38}	α_{39}	β_0	β_1	$\dots$	β_{36}	β_{37}
Roqr	ρ	19	$\rho \uparrow^{(4)}$	0	0	0	α_0	$\dots$	α_{35}	α_{36}	1	1	α_{39}	β_0	β_1	$\dots$	β_{36}	β_{37}
Lq	ρ	1A	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	β_2	β_3	β_4	β_5	$\dots$	α_0	α_1
Lqy	ρ	1B	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	β_0	β_1	β_2	β_3	β_4	β_5	$\dots$	α_0	α_1
Rq	ρ	1C	ρ	α_{-1}	α_{-1}	α_{-1}	α_0	$\dots$	α_{35}	α_{36}	α_{37}	α_{38}	α_{39}	β_0	β_1	$\dots$	β_{36}	β_{37}
Rqr	ρ	1D	$\rho^{(4)}$	α_{-1}	α_{-1}	α_{-1}	α_0	$\dots$	α_{35}	α_{36}	1	1	α_{39}	β_0	β_1	$\dots$	β_{36}	β_{37}
Lcq	ρ	1E	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	0	0	0	0	$\dots$	α_0	α_1
Lcqy	ρ	1F	ρ	α_1	α_2	α_3	α_4	$\dots$	α_{39}	0	0	0	0	0	0	$\dots$	α_0	α_1

$\uparrow_0 \rightarrow A_{-1}$ even when $\rho = 0$.

L: left

R: right

y: allows β_0 to shift into α_{39}

q: shifts A and Q; on a right shift α_{39} goes into β_0 ; on a left shift α_0 (not α_{-1}) goes into β_{39}

r: round. The digits shifting from A_{38} to A_{39} is made to be a "1".

o: on a right shift, o's are fed into the left end of A

c: clear Q to zero prior to the execution of a shift

(4) If ρ is zero for any of the round shifts such as "Ror, 0", the round is not made.

3. Multiplication

There are two multiplication instructions:

Symbol Code

M M(a) 38 M(a) a straight multiplication with no round off.

Mr M(a) 39 M(a) " " " " round off.

The functions of the three registers are as indicated below.

Operation	Registers		
	A	Q	S
multiplication	most significant product	multiplier least significant product	multiplicand

When a multiply operation is given, it is assumed that the multiplier is already stored in the Q-register either as a result of a preceding instruction or as a result of an earlier arithmetical operation. The address associated with a multiply operation refers to the storage location containing the multiplicand.

To illustrate the steps of multiplication as they are performed by the machine, let

$$b = -b_0 + \sum_{i=1}^{39} b_i 2^{-i}$$

be stored in the Q-register, and let M(a) represent the address of the multiplicand a. The algorithm, which consists of a series of additions

and right shifts, yields the product

$$ab = a \left[-b_0 + \sum_{i=1}^{39} b_i 2^{-i} \right]$$

The instruction 38 M(a) directs the arithmetic unit to perform the sequence of steps as follows:

1. The multiplicand, a, is transferred from the storage to the S-register.
2. The A-register is cleared to zeros.
3. The plusser forms the sum of the contents of the A and S-registers.
4. b_{39} located in Q_{39} is examined; if $b_{39} = 1$, the contents of P are gated into A; if $b_{39} = 0$, the contents of A remain unchanged.
5. The A and Q registers are shifted one place to the right, as in the instruction "Rq, 1"; b_{39} is shifted off of Q; Q_{39} now contains b_{38} , etc.

The last three of the above steps are repeated 38 times. After thirty-nine right shifts have occurred, the extended register positions A_{-1} , ..., A_{39} , and Q_0 , ..., Q_{38} contain,

$$\sum_{i=1}^{39} ab_i 2^{-i},$$

while Q_{39} contains the sign digit, b_0 , of the multiplier. If the sign is positive, that is $b_0 = 0$, then $-b_0 a$ equals zero and the multiplication is complete. If the sign is negative, that is $b_0 = 1$, then the final step is to subtract $b_0 a$ without shifting. A double precision product consisting of the sign and seventy-eight digits lies in the A and Q registers; Q_{39} still retains the sign digit of the multiplier.

The instruction 39 M(a) directs the computer to form the rounded product of a and b. The final result

$$ab + 2^{-40}$$

is found in the A-register. At the time of the preliminary clearing of A, a "1" is inserted in the a_1 position. This has the effect of adding 2^{-1} at the start, and after 39 right shifts this 2^{-1} becomes 2^{-40} .

The O.F. toggle will be set by a multiplication only in the special case $a = b = -1$, which gives a nondigital product, +1.

Example 1: Find the product ab without round off. Let

$$b = 5/8 = 0.101,$$

$$a = 3/8 = 0.011.$$

For convenience, use the short four digit registers.

Steps	Registers			Plusser
	S	A	Q	
			0.101	
1	0.011			
2		0,0.000		
3				0,0.011
4		0,0.011		
<u>5</u>		0,0.001	10.10	
3				0,0.100
4		0,0.001		
<u>5</u>		0,0.000	110.1	
3				0,0.011
4		0,0.011		
<u>5</u>		0,0.001	1110.	

Since the sign digit is positive, the multiplication is finished.

The product found in the A and Q registers is

$$\begin{aligned}
 & \begin{array}{c} A \quad Q \\ ab = 0.001 \ 111 \end{array} \text{ (binary)} \\
 &= -0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1 \cdot 2^{-6} \\
 &= 1/8 + 1/16 + 1/32 + 1/64 \\
 &= 15/64.
 \end{aligned}$$

Example 2: Find the product ab with round off. Let

$$b = -5/8 = 1.011,$$

$$a = 3/8 = 0.011.$$

Use the short four digit registers.

Steps	Registers			Plusser
	S	A	Q	
			1.011	
1	0.011			
2		0,0.100		
3				0,0.111
4		0,0.111		
<u>5</u>		0,0.011	11.01	
3				0,0.110
4		0,0.110		
<u>5</u>		0,0.011	011.0	
3				0,0.110
4		0,0.011		
<u>5</u>		0,0.001	1011.	
sign digit = 1		1,1.110	1011.	1,1.110

The product found in the A-register is

$$\begin{aligned}ab &= 1.110 \\&= -1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} \\&= -1 + 1/2 + 1/4 \\&= -1/4.\end{aligned}$$

In computations which require only the most significant part of the product, the effect of using multiplication with round off is to reduce the error and to eliminate the bias. There are many computations, such as forming multiple precision products and multiplying by scaled integers, which require the least significant part of the product. In such cases the "38" operation must be chosen. However, it is important to remember that the Q-register contains the least significant digits but not the sign of the product.

Exercise: Find the product of $6 \cdot 2^{-39}$ and $-5 \cdot 2^{-39}$. What are the contents of the A and Q registers after the multiplication has been completed?

4. Division

There are four division instructions:

	<u>Symbol</u>	<u>Code</u>	<u>Result</u>
1.	D M(b)	3A M(b)	The dividend is stored in A; the quotient is not rounded.

<u>Symbol</u>	<u>Code</u>	<u>Result</u>
2. Dq M(b)	3B M(b)	The dividend is stored in A and Q and the least significant digits of the dividend in Q are gradually shifted into A during the division process; the quotient is not rounded.
3. Dr M(b)	3E M(b)	The dividend is stored in A; the last digit of the quotient is forced to be 1.
4. Dqr M(b)	3F M(b)	The dividend is stored in A and Q and the least significant digits of the dividend in Q are gradually shifted into A during the division process; the last digit of the quotient is forced to be 1.

The functions of the three registers in division are as indicated below:

Operation	Registers		
	A	Q	S
Division	Dividend (most significant digits) Remainder times 2^{39}	Dividend (least significant digits for Dq and Dqr) Quotient	Divisor

When a divide operation is given, it is assumed that the dividend is already stored either as a single precision number in A or as a double precision number in A and Q. The dividend is stored either as a result of a preceding instruction or as a result of an earlier arithmetical operation. The address associated with a division operation refers to the storage location containing the divisor.

To illustrate the steps of division as they are performed by the machine, let a represent the dividend and let b represent the divisor. The division algorithm consists of a series of subtractions and left shifts. Since the O.F. toggle is not always set when the quotient is nondigital, a test for a digital quotient must be made before the division instruction is given. A digital quotient is always assured if $|a| < |b|$. This test automatically excludes division by zero. However, a digital quotient can also be obtained when $|a| = |b|$ provided b is negative or provided a and b have different signs. A nondigital quotient is obtained when $|a| > |b|$ or $a = b \geq 0$.

The digits of the quotient are formed consecutively in Q_{39} , beginning with the sign digit and the current remainder is formed in A . The digits of the quotient and the current remainder are then shifted to the left. On the left shift, digits are shifted from Q into A ($A_{39} \leftarrow Q_0$) if the operation is "Dq" or "Dqr"; if the operation is "D" or "Dr", a left shift drops the digit shifted from Q_0 and sends a zero to A_{39} ($A_{39} \leftarrow 0$).

The division instruction directs the arithmetic unit to perform the sequence of steps as follows:

1. (a) -39 goes to the shift counter to count the steps.
(b) The divisor, b , is copied from the storage into the S^U -register.

2. The signs of a and b are compared. (after this step⁽⁵⁾, the current remainder in A always has the same sign as b .)
 - (a) If a and b have like signs, that is $A_X = S^0$, the algorithm proceeds to step 3.
 - (b) If a and b have unlike signs, that is $A_X \neq S^0$, then
 - (1) $a + b$ is formed in the plusser and copied into A^U . (Since the signs are different, this is really the sign of b times $(|b| - |a|)$.⁽⁵⁾)
 - (2) Q^U is cleared; $Q^{40(6)}$ is set to 1.
 - (3) Proceed to step 4.
3. (a) The difference between the contents of A_L and S^U is formed in the plusser.
 - (b) P_X and S^0 are compared.
 - (1) If $P_X = S^0$, Q^{40} is set to 1; the contents of the plusser are copied into A^U .
 - (2) If $P_X \neq S^0$, Q^{40} is set to 0; the contents of A_L are copied into A^U . However, if this is the 40th step and "Dr" or "Dqr" has been given, then (1) is carried out.
4. (a) Q_L is copied into Q^U (digits 0-39).
 - (b) The shift counter is increased by 1.
 - (c) If this is the 40th step, the algorithm proceeds to step 6.

(5) One exception to this statement occurs when $b = -a < 0$.

(6) Q^U has an extra digit on the right labeled Q^{40} .

5. (a) The contents of A^U and Q^U are shifted one place to the left as the digits are copied into A_L and Q_L .
- (1) If "Dq" or "Dqr" is being executed Q^0 goes into A_{39} .
- (2) If "D" or "Dr" is being executed Q^0 is shifted off of the register and 0 goes into A_{39} .
- (b) The algorithm proceeds to step 3.
6. (a) A^U is copied straight down to A_L .
- (b) Q^U is shifted one place to left as the digits are copied down into Q_L .

Example 1: Find the quotient of a/b without round off. Let

$$a = 1/8 = 0.0010 \dots 0$$

$$b = 1/2 = 0.100 \dots 0$$

Example 2: Find the quotient of a/b with round off. Let

$$a = -1/8 = 1.1110 \dots 0$$

$$b = 1/2 = 0.100 \dots 0$$

Example 1.

Steps	Registers					S.C.
	S^U	A_L	Q_L	Q^{40}	Plusser	
	$S_0, S_1, S_2, S_3 \dots S_{39}$	$A_{-1}, A_0, A_1, A_2, A_3 \dots A_{39}$	$Q_0, Q_1, Q_2, Q_3 \dots Q_{39}$		$P_{-1}, P_0, P_1, P_2, P_3 \dots P_{39}$	
1	0.1 0 0 ... 0	0,0 .0 0 10 ... 0				-39
2					0,0 .0 0 1 = 3	
3					$\frac{1,1 .1 0 0}{1,1 .1 0 1} = -b$	
4				0		-38
5		0,0 .0 1 0 ... 0	0		0,0 .0 1 0 = remainder in A	
3					$\frac{1,1 .1 0 0}{1,1 .1 1 0} = -b$	
4				0		-37
5		0,0 .1 0 ... 0	00		0,0 .1 0 0 = r	
3					$\frac{1,1 .1 0 0}{0,0 .0 0 0} = -b$	
		0,0 .0 0 0 ... 0		1		
4						-36
5		0,0 .0 0 0 ... 0	001			
...	...	...	...	...	...	...
3				0	1,1 .1 0 0 ... 0	
4						0
5		0,0 .0 0 0 ... 0	0.0 0 1 ... 0			
3				0	1,1 .1 0 0 ... 0	
4						1
6		0,0 .0 ... 0	0.0 1 0 ... 0			

The quotient found in the Q-register is

$$a/b = 0.010 \dots 0$$

$$= -0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} \dots 0 \cdot 2^{-39}$$

$$= 1/4 + 0 \cdot 2^{-40}$$

The remainder times 2^{39} , in the A-register is zero.

Steps	Registers					S.C.
	S^U	A_L	Q_L	Q_{40}	Plusser	
	$S_0, S_1, S_2, S_3 \dots S_{39}$	$A_{-1}, A_0, A_1, A_2, A_3 \dots A_{39}$	$Q_0, Q_1, Q_2, Q_3 \dots Q_{39}$		$P_{-1}, P_0, P_1, P_2, P_3 \dots P_{39}$	
1	0.1 0 0 ... 0	1,1 .1 1 1 0 ... 0				-39
2		(A^U) 0,0 .0 1 1 0 ... 0		1	1,1 .1 1 1 0 = a 0,0 .1 0 0 0 = b <u>0,0 .0 1 1 0 = a + b</u>	-38
4		0,0 .1 1 0 ... 0	1			
5		(A^U) 0,0 .0 1 0 0 ... 0		1	0,0 .1 1 0 0 = r 1,1 .1 0 0 0 = -b <u>0,0 .0 1 0 0</u>	-37
4		0,0 .1 0 0 ... 0	11			
5		(A^U) 0,0 .0 0 0 ... 0		1	0,0 .1 0 0 = r 1,1 .1 0 0 = -b <u>0,0 .0 0 0</u>	-36
4		0,0 .0 0 ... 0	111			
...	...	...	...	...	...	...
3				0	1,1 .1 0 ... 0 = -b	0
4		0,0 .0 0 ... 0	1 1 1 0 ... 0			
5		(A^U) 1,1 .1 0 ... 0		1	1,1 .1 0 ... 0 = -b	1 1
3		1,1 .1 0 ... 0	1.1 1 0 ... 0	1		
4						
6						

$$a/b = 1.110 \dots 01 + (R/b)2^{-39}$$

$$= -1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} \dots + 0 \cdot 2^{-38} + 1 \cdot 2^{-39} + (R/b)2^{-39}$$

$$= -1 + 1/2 + 1/4 + 1 \cdot 2^{-39} + (R/b)2^{-39}$$

$$= -1/4 + 1 \cdot 2^{-39} + (R/b)2^{-39}$$

The remainder (R) times 2^{39} , in the A-register, is $-1/2$.

C. Logical Operations of the ORACLE

1. Transfer Operations

The ORACLE carries out instructions in sequence from consecutively numbered storage cells, executing the left-hand instruction and then the right-hand instruction, until a transfer instruction of the type below is encountered. There are two kinds of transfer operations:

1. Unconditional
 - (a) do not transfer
 - (b) transfer
2. Conditional
 - (a) overflow conditional
 - (b) sign conditional.

In a conditional transfer, if the condition is not met, the sequence of instructions is not interrupted. Otherwise, the transfer instruction will send the controls either to the right or to the left half of the word whose address is given. The address of the instruction "Ntr, -" is irrelevant. This instruction is sometimes called a stall. However, it is not a pure stall since any transfer instruction will clear the O.F. toggle. When the O.F. toggle is set, it is cleared if and only if a transfer instruction is given, whether or not a transfer actually takes place.

The A_{-1} toggle is used to test the sign of a number. If $A_{-1} = 0$, the number in A is ≥ 0 ; if $A_{-1} = 1$, the number in A is < 0 . It is important to remember that, if a test on the sign of a number immediately

follows a left shift, the left shift must be sufficient to place the sign digit in the A_{-1} position⁽⁷⁾.

The transfer instructions are given in the table below.

<u>Symbol</u>	<u>Code</u>	<u>Result</u>
Ntr -	40 -	Stall
Tr m	41 m	Transfer control to right of m
Tl m	43 m	Transfer control to left of m
T + r m	48 m	Transfer control to right of m when $a_{-1} = 0$
T - r m	49 m	Transfer control to right of m when $a_{-1} = 1$
T + l m	4A m	Transfer control to left of m when $a_{-1} = 0$
T - l m	4B m	Transfer control to left of m when $a_{-1} = 1$
Tnr m	4C m	Transfer control to right of m when O.F. toggle is clear
Tyr m	4D m	Transfer control to right of m when O.F. toggle is set
Tnl m	4E m	Transfer control to left of m when O.F. toggle is clear
Tyl m	4F m	Transfer control to left of m when O.F. toggle is set

The symbol code letters are defined:

m - address

r - right

l - left

n - no

y - yes

(7) See instructions in locations 101 and 102 of the example on page 50.

Example: Write a code for the problem given below.

Problem: If a/b is positive, compute $a/b - c$; if a/b is negative,

compute $a/b + c$. Given: (1) $a = 3/8$

(2) $b = 1/4$

(3) $c = 3/4$

(4) a is stored in cell 012

(5) b is stored in cell 013

(6) c is stored in cell 014.

The digital answer should be left in the A-register.

		CODE		REMARKS	
<u>Storage Location</u>	<u>Operation</u>	<u>Address</u>	<u>Operation</u>	<u>Address</u>	
100	24	012	1C	001	$a \rightarrow rA; 2^{-1}a$
101	3B	013	1B	029	$(2^{-1}a)/b \rightarrow rQ; 2^{41}(a/b) \rightarrow rA,$ $q_0 \rightarrow a_{-1}, \text{ etc.}$
102	4B	120	22	014	if a/b is negative, transfer left 120; $a/b - c \rightarrow rA$
⋮					
120	20	014	43	103	$a/b + c$; transfer to the left of 103

Exercise: Write the contents of the A and Q registers at the end of each instruction in the example above

(a) if a and b are negative numbers,

(b) if a is negative and b is positive.

2. Substitutions

Information from a storage cell may be copied to either the A or the Q registers; and conversely, information from either the A or the Q registers may be copied to a storage cell. The only entry to A from the storage is the indirect path through S^U , plusser, etc., by means of the summation instructions. An entry to Q from the storage is effected directly by the instruction "Q, M(d)", which causes the number d in the storage cell whose address is M(d) to be copied into Q.

Whole words or certain parts of words from A or Q may replace information already stored in a storage cell. A partial substitution leaves the contents of the other positions unaffected. A forty-digit word is composed of four parts which may be copied, as illustrated.

WORD

8 binary digits	12 binary digits	8 binary digits	12 binary digits
--------------------	---------------------	--------------------	---------------------

Since there are sixteen possible combinations of the four parts, thirty-two substitution orders are provided for copying the digits of any part or combination of parts of a word from A or Q into a specified storage cell. The possible combinations include the null combination, which is another stall operation.

The mnemonic symbol for a substitution from A consists of four characters corresponding to the four parts of a word. If the part is to be copied, the character is an "a"; if the part is not to be copied, the character is a "o". The second hex digit of the code may be derived from the mnemonic symbol by representing the four parts of a word as a group of four binary digits. The "a" is replaced by a "1". For substitutions from Q, "a" is replaced by "q". The substitution instructions are indicated in the table on the following page.

Substitution Instructions

Symbol	Code	Result
0000 -	50 -	Stall
000a M(d)	51 M(d)	$a_{28} \dots a_{39}$ replaces $d_{28} \dots d_{39}$
00a0 M(d)	52 M(d)	$a_{20} \dots a_{27}$ replaces $d_{20} \dots d_{27}$
00aa M(d)	53 M(d)	$a_{20} \dots a_{39}$ replaces $d_{20} \dots d_{39}$
0a00 M(d)	54 M(d)	$a_8 \dots a_{19}$ replaces $d_8 \dots d_{19}$
0a0a M(d)	55 M(d)	$a_8 \dots a_{19}, a_{28} \dots a_{39}$ replaces $d_8 \dots d_{19}, d_{28} \dots d_{39}$
0aa0 M(d)	56 M(d)	$a_8 \dots a_{27}$ replaces $d_8 \dots d_{27}$
0aaa M(d)	57 M(d)	$a_8 \dots a_{39}$ replaces $d_8 \dots d_{39}$
a000 M(d)	58 M(d)	$a_0 \dots a_7$ replaces $d_0 \dots d_7$
a00a M(d)	59 M(d)	$a_0 \dots a_7, a_{28} \dots a_{39}$ replaces $d_0 \dots d_7, d_{28} \dots d_{39}$
a0a0 M(d)	5A M(d)	$a_0 \dots a_7, a_{20} \dots a_{27}$ replaces $d_0 \dots d_7, d_{20} \dots d_{27}$
a0aa M(d)	5B M(d)	$a_0 \dots a_7, a_{20} \dots a_{39}$ replaces $d_0 \dots d_7, d_{20} \dots d_{39}$
aa00 M(d)	5C M(d)	$a_0 \dots a_{19}$ replaces $d_0 \dots d_{19}$
aa0a M(d)	5D M(d)	$a_0 \dots a_{19}, a_{28} \dots a_{39}$ replaces $d_0 \dots d_{19}, d_{28} \dots d_{39}$
aaa0 M(d)	5E M(d)	$a_0 \dots a_{27}$ replaces $d_0 \dots d_{27}$
aaaa M(d)	5F M(d)	$a_0 \dots a_{39}$ replaces $d_0 \dots d_{39}$
0000 -	70 -	Stall
000q M(d)	71 M(d)	$q_{28} \dots q_{39}$ replaces $d_{28} \dots d_{39}$
	...	
qqqq M(d)	7F M(d)	$q_0 \dots q_{39}$ replaces $d_0 \dots d_{39}$
Q M(d)	60 M(d)	$d_0 \dots d_{39}$ replaces $q_0 \dots q_{39}$

3. Stop Operations

Two kinds of stop instructions are available in the ORACLE.

a. Stop Instruction

The code for a stop is "00 -", where the address portion is irrelevant. This stop instruction is used to terminate a computation.

b. Breakpoint Stop

A breakpoint stop can be ordered in either a right-hand instruction or a left-hand instruction by meeting the two requirements as follows:

- (1) A "1" must be inserted in the extreme left-hand binary digit position of the address portion of an instruction. It has already been stated that the largest address 2047 is written as 7FF in hex notation. In binary notation this is 0111 1111 1111. The address of an instruction is designated in positions a_8 to a_{19} and in positions a_{28} to a_{39} . The digit positioned in toggle a_8 or a_{28} is available for indicating a breakpoint stop by inserting a "1" in this position. This is equivalent to adding 800 to the hex address portion of an instruction.

Example: Add a breakpoint stop to the instruction.

3A 369

In binary notation, the address 369 is 0011 0110 1001. In binary notation, $369 + 800 = B69$ is 1011 0110 1001. Note that a "1" has been inserted in the extreme left-hand binary digit position.

- (2) A certain switch on the console, which is called the breakpoint, B.P., switch must be set in the up position.

The computer will stop prior to the execution of the instruction which carries the breakpoint stop.

The breakpoint stop is used largely to check a program the first time it is run. It is sometimes convenient to have the machine stop at

certain critical points in the computation so that the contents of the registers may be examined. After the routine is well tested, the breakpoint stops may be removed or they may be ignored by setting the B.P. switch in the down position.

D. Input-Output Operations

An input unit is capable of copying information into the electrostatic storage. An output unit is capable of receiving information from the electrostatic storage [9]. The ORACLE has five media for input and output as listed.

1. Magnetic Tapes for
 - a. Auxiliary Storage (tape 2" wide)
 - b. Primary Output (tape 5/8" wide)
2. Paper Tapes for
 - a. Primary Input
 - b. Primary Output
3. Computer Console Keyboard for
 - a. Manual Instructions to the Computer
 - b. Secondary Input
4. Computer Console Typewriter for
 - a. Secondary Output
 - b. Monitoring Operations at the Console
5. Curve Plotter for Output

Automatic control over all input-output units is obtained by the computer through the instructions in the stored program. All input-output instructions, except the one word and one character instructions, use the control counter. Therefore, they must be left-hand instructions followed by a transfer of control. Also, all input-output instructions are

storage-type instructions; information either to or from the ORACLE comes directly from the storage. However, the direct path for words to or from the electrostatic storage is through the Q-register.

A parity checking system, which permits more efficient operation of the ORACLE, is used on all input-output equipment except the curve plotter. This is an odd-even type of check consisting of counting the number of binary ones in a character and determining if their sum is odd or even. This system is completely automatic. If an error exists during an input or output operation, the operation is stopped and an error light indicator on the console will be turned on.

Each of the above media is discussed separately below.

1. Magnetic Tapes

a. Auxiliary Storage. A magnetic tape which is two inches in width is used by the ORACLE as an auxiliary or external storage. There are 42 binary positions across the width of the tape. For checking purposes, each half-word, 20 bits of information plus 1 parity check bit, is recorded twice in the 42 positions. Information is recorded on the tape in blocks of words. The size of a block is 128 words. A block length, 2.41 inches of tape, includes the distance required to stop and start the tape. The packing density is 193 characters per inch. The unit is composed of four tape drives, numbered 0, 1, 2, and 3. Each drive is capable of holding 1000 feet of tape or 5,000 blocks.

BLOCK X IN POSITION TO BE READ FORWARD
BLOCK Y IN POSITION TO BE READ BACKWARD
BLOCK Z HAS BEEN DELETED BY PUNCHING
HOLES 1 AND 2 (0.100-in. D)

SYSTEM IS SYMMETRICAL WITH RESPECT TO
FORWARD OR REVERSE

VIEW OF OXIDE COATED SIDE
ORACLE Magnetic Tape Data.

FIGURE 3

Information stored on this tape may be processed only through the ORACLE. The method by which information may be copied to or received from this magnetic tape is through the read and write instructions in the stored program. Information stored on magnetic tape is not directly accessible to the arithmetic unit. The information must be read from the tape into the electrostatic storage before it is used by the arithmetic unit.

The tape may be instructed to move in either the forward or backward direction. There are then six basic magnetic tape instructions for reading, writing and hunting blocks of information on the tape. In addition, there is a basic rewind instruction. To obtain the specific instruction from the basic instruction, one simply adds the tape drive number to the second hex digit of the basic instruction. This is illustrated in the table on page 62.

(1) Read or Write Instructions. Since read and write instructions use the control counter, they must be left-hand instructions followed by a transfer of control instruction. On reading or writing n words, $n^{(8)}$ must be inserted in Q on a previous instruction. Since the counter for words or blocks refers to only the last twelve bits or digits in Q, the upper limit of n is 4095. If a read or write instruction of zero words

(8) A * on the right of a number means that the digits are located in the extreme right end of a word.

is given, nothing is read or written and the tape does not move. If $n \neq 0 \pmod{128}$, that is, if n is not a multiple of 128, then the unused portion of the last block written will be automatically filled out with zeros; and on a read instruction, the uncalled for words in the last block will not be read, but the tape will continue to the end of that block.

The address portion of a read or write instruction specifies the first word in the storage and all succeeding words are read or written consecutively thereafter.

Information passing between the electrostatic storage and the auxiliary storage must pass through the Q-register. On a read instruction, the last word read will be left in Q_L and the control counter will contain the address following the last word written. After a write instruction, the Q-register will be cleared to zeros and the control counter may not contain the address following the last word written. However, A is not altered by a read or write instruction.

(2) Hunt Instruction. A hunt or search instruction is used to position the tape before and after reading or writing. This instruction may be either a right or a left-hand instruction. The address portion of a hunt instruction is irrelevant. On hunting n blocks, n^* must be inserted in Q on a previous instruction. If a hunt order of zero blocks is given, no hunting takes place and the tape does not move. Neither A nor Q is altered by a hunt instruction.

(3) Rewind Instruction. A rewind instruction is used to position the tape back to the beginning of the first block. This instruction may be a right or a left-hand instruction. The address portion of a rewind instruction is irrelevant. The contents of Q are irrelevant. Neither A nor Q is altered by a rewind instruction.

Example: Copy the 5th block of information from drive 0 to the 1st block of drive 2. Position drives 0 and 2 to block 0. Assume that locations 200-280 are available for receiving information from tapes. Note that information transferred from one tape to another must first be read into the electrostatic storage.

<u>Storage Location</u>	<u>Instructions</u>	<u>Remarks</u>
094	60 094 A8 004	4* → rQ; hunt forward on drive 0
095	60 095 40 080	128* → rQ; stall with 128* in address position
096	B0 200 43 097	read forward on drive 0; transfer control
097	60 097 40 080	128 → rQ; stall with 128* in address position
098	A2 200 43 099	write forward on drive 2; transfer control
099	BC 000 BE 000	rewind drive 0; rewind drive 2.

MAGNETIC TAPE AUXILIARY STORAGE

Symbol	Basic Hex Instruction Code	Q ₂₈ , Q ₂₉ , ..., Q ₃₉	Specific Hex Instruction Code	Drive No.
Write Forward (WF) (Uses control counter)	A0 m	Number of <u>words</u> to be written	A0 m A1 m A2 m A3 m	0 1 2 3
Write Backward (WB) (Uses control counter)	A4 m	Number of <u>words</u> to be written	A4 m A5 m A6 m A7 m	0 1 2 3
Hunt Forward (HF)	A8 -	Number of <u>blocks</u> to be hunted	A8 - A9 - AA - AB -	0 1 2 3
Hunt Backward (HB)	AC -	Number of <u>blocks</u> to be hunted	AC - AD - AE - AF -	0 1 2 3
Read Forward (RF) (Uses control counter)	B0 m	Number of <u>words</u> to be read	B0 m B1 m B2 m B3 m	0 1 2 3
Read Backward (RB) (Uses control counter)	B4 m	Number of <u>words</u> to be read	B4 m B5 m B6 m B7 m	0 1 2 3
Rewind (RW)	BC -	Irrelevant	BC - BD - BE - BF -	0 1 2 3

b. Primary Output. A magnetic tape five-eighths inch in width is one of the primary output media from the ORACLE. However, the tape cannot be used as an input medium. Information received directly from the ORACLE is recorded on the tape by a magnetic tape recorder. The instruction codes for directing information to the tape are included in the output operations for paper tape, magnetic tape and for the console typewriter.

The end of the reel of tape should be marked with a slit 0.25" long and 0.050" wide in the tape. When this slit is sensed by a photocell in the tape recorder, the machine stops and holds up the output tape instruction that is in progress until either the reel is rewound, or a new reel is loaded, or the leader switch is depressed. The machine then continues recording until the conclusion of the interrupted instruction.

The tape is processed in the peripheral units as listed below.

The tape printing unit serves as a printer for information from both paper and magnetic tapes. It is operated as a typewriter. It prints the information read from the magnetic tape using the magnetic tape reader.

The tape processing unit is used to discover and indicate impurities or flaws in the magnetic tape before it is used in the magnetic tape recorder and the printer units.

2. Paper Tapes.

Paper tape is the primary input medium as well as one of the primary output media. A seven-hole paper tape is used for the ORACLE. The paper tape can be punched in any of the seven positions across its width. A

hole in the tape represents a "1", while the absence of a hole represents a "0". The feed or sprocket holes, which are smaller than the other holes, lie between the third and fourth positions on the tape. Six of the seven binary positions are used to define up to sixty-four separate codes or characters. The seventh binary position is used as a parity check bit; that is, either a "0" or "1" is recorded in the seventh bit in order that the sum of the binary ones stored at positions 1 through 7 on the tape is odd.

Paper Tape

a. ORACLE General Code. The "ORACLE General Code" may be used in the ORACLE either as a hexadecimal code or as an alphameric⁽⁹⁾ code. It has been discussed previously how the hexadecimal code is used to represent

(9) Alphameric is a word coined from alphabetic and numerical. The alphameric code not only includes the decimal numbers but also includes the English alphabet as well as the most commonly used Greek characters.

numbers and instructions for the ORACLE. The alphameric code is used for printing and editing the ORACLE output data on the typewriter. Some of the code symbols are used as control characters, while the letters of the alphabet are used for identifying headings, etc. Although an alphameric character is defined by six binary digits on paper tape, it is allotted eight binary digits in the machine. Therefore, five of these characters will form a machine word. For this reason special input-output instructions are required for alphameric characters. These instructions are listed with the input-output instructions for paper tape, 5/8 inch magnetic tape, and for the console typewriter. See page 79.

ORACLE GENERAL CODE

The Pattern
of Holes

Check Bit	Sprocket Hole	<u>Keys on Typewriter</u>			
		<u>Bi Hex</u>	<u>Lower Case</u>	<u>Upper Case</u>	<u>Explanation Number</u>
7654S321					
0	o	00	0	0	1
	o o	01	1	Q	2
	o o	02	2	B	3
0	o oo	03	3	Z	4
	oo	04	4	S	5
0	oo o	05	5	e	6
0	ooo	06	6	λ	7
	oooo	07	7	φ	8
	oo	08	8	σ	9
0	oo o	09	9	π	10
0	oo o	0A	a	A	
	oo oo	0B	b	B	
0	oo0	0C	c	C	
	oo0 o	0D	d	D	
	oo00	0E	e	E	
0	oo000	0F	f	F	
	0 o	10	SPACE		11
0	o o o	11	g	G	
0	o o o	12	h	H	
	o o oo	13	i	I	
0	o oo	14	j	J	
	o oo o	15	k	K	
	o ooo	16	l	L	
0	o oooo	17	m	M	
0	ooo	18	n	N	
	ooo o	19	o	O	
	ooo o	1A	p	P	
0	ooo oo	1B	q	Q	
	ooo0	1C	r	R	
0	ooo0 o	1D	s	S	
0	ooo00	1E	t	T	
	ooo000	1F	TAPE FEED		12

ORACLE GENERAL CODE (Cont.)

The Pattern

of Holes

Check Bit	Sprocket Hole	Bi Hex	<u>Keys on Typewriter</u>		<u>Explanation</u> <u>Number</u>
			<u>Lower</u> <u>Case</u>	<u>Upper</u> <u>Case</u>	
7654S321					
0 0		20	/	*	
00 0 0		21	u	U	
00 0 0		22	v	V	
0 0 00		23	w	W	
00 00		24	x	X	
0 00 0		25	y	Y	
0 000		26	z	Z	
00 0000		27	,	?	
00 00		28	'	)	
0 00 0		29	:	(	
0 00 0		2A	=		
00 00 00		2B	+	Δ	14
0 000		2C	.	$\frac{\circ}{\circ}$	15
00 000 0		2D	-	"	16
00 0000		2E			
0 00000		2F	STOP CODE		17
000 0		30	Rs		18
00 0 0		31	TABULATE		
00 0 0		32	UPPER CASE		19
000 0 00		33	BACK SPACE		20
00 00		34	LOWER CASE		21
000 00 0		35	CARRIAGE RETURN		13
000 000		36			
00 0000		37			
0000		38	Ps		22
00000 0		39			
00000 0		3A			
0000 00		3B			
000000		3C			
00000 0		3D			
000000		3E	Bp		23
00000000		3F	CODE DELETE		24

CHARACTER NOTES

1. PARTIAL DIFFERENTIATION SYMBOL
2. ALPHA
3. BETA
4. SUMMATION SIGN
5. DOLLAR SIGN
6. EPSILON
7. LAMBDA
8. PHI
9. SIGMA
10. PI
11. SPACE: A space character is used to terminate a hexadecimal word. It is verified and reproduced.
12. TAPE FEED: A tape feed character is punched on the tape when the tape is being run out of the punch for a leader. It is not reproduced, verified, or read into the machine by the Ferranti reader.
13. CR: The carriage return character is used to terminate instructions for reading hexadecimal words. It returns the carriage on the typewriter and it is verified and reproduced.
14. DELTA
15. PERCENT SIGN
16. MINUS SIGN
17. STOP CODE: This character causes the readers on the printer to stop. In verification, this character causes the reader to stop, but it is verified in the normal manner and punched when the reader start button is punched. In reproduction, this character causes the reader to stop, but it is not reproduced.

18. **Rs:** This character (Reader Stop) is used to terminate instructions for reading alphameric words. It is reproduced and verified. The Rs symbol must be either the first character of a new word, or preceded by four zeros.
19. **UPPER CASE:** The upper-case character locks the typewriter in an upper case condition such that all characters following will be printed in upper case. The typewriter stays in this condition until the lower-case character changes it.
20. **BACKSPACE:** A backspace character causes the typewriter to backspace.
21. **LOWER CASE:** The lower case character locks the typewriter in a lower case condition such that all characters following will be printed in lower case. The typewriter remains in this condition until the upper-case character changes it.
22. **Ps:** This character (Punch Stop) is used to terminate the output instruction for alphameric data. It is ignored by the typewriter of the printing unit, but it is reproduced and verified. Characters following a Ps symbol in a word must be zeros.

Examples: Ps0000

REDPs0

BLUEPs

23. **Bp:** This character acts as a breakpoint-stop for the readers on the printer. It does not actuate the typewriters on the printers. It is reproduced and verified.
24. **CODE DELETE:** The code delete symbol is used to mark out unwanted characters on the tape. It acts as a tape feed to all readers. It is not reproduced, verified, or read into the machine by the Ferranti reader.

b. Primary Input. The primary input device connected to the ORACLE is the Ferranti photoelectric reader. The coded program and initial data punched on paper tape may be read into the electrostatic storage by placing the tape under the reader and giving a paper tape read instruction.

The input tape of the coded program and initial data is prepared by using the peripheral tape preparation units. There are two tape preparation units. The primary difference between the two units is the keyboard. One unit is provided with an alphameric keyboard, while the other unit is provided with a hexadecimal keyboard. In addition to the keyboard, each unit contains a paper tape punch, a reader, a parity check monitor and a verifier.

The operations in preparing tapes are listed as follows:

1. To punch a tape from a manuscript.
2. To verify an existing paper tape against a manuscript by punching a verified or second tape.
3. To reproduce or reperfurate a tape for the purpose of making corrections or joining short tapes. Every time a tape is reproduced it is wise to check the copy by comparing it with the original tape.

A hand punch is available for changing one or two characters. This sometimes avoids reproducing a long tape.

c. Primary Output. A sixty character per second punch, which is connected to the ORACLE, is the output device for paper tape. Output tapes are processed in one of two tape printing units. It has been stated previously that each printer will print information from both paper and 5/8" magnetic tapes.

Example: The results from a computation will be printed in two columns, with headings β and $\tan \beta$. (1) Prepare an input tape for the two headings. (2) By referring to the paper tape instructions, write codes (a) to read the tape into the electrostatic storage and (b) to punch the information from the electrostatic storage to paper tape.

(1) The tape preparation unit with the alphameric keyboard is used to punch the following words:

```

s s s s u
p p p p c

      s s s s
 $\beta$  p p p p

s s l
p p c t a

      s u c
n p c  $\beta$  r

Ps 0 0 0 0
00 0 0 0 Rs

```

where $\overset{s}{p}$ represents space; $\overset{u}{c}$ represents upper case; $\overset{l}{c}$ represents lower case; and $\overset{c}{r}$ represents carriage return.

(2) <u>Storage Location</u>		<u>Instruction Word</u>	<u>Explanation</u>
(a)	020	98 10A 43 021	read alphameric characters from paper tape into cells 10A through 10E; transfer of control
(b)	021	88 10A 43 022	punch alphameric characters onto paper tape from cells 10A to Ps symbol; transfer of control
	022	00 000 00 000	stop

3. Computer Console Keyboard

The console keyboard is composed of a control keyboard and an information keyboard. The control keyboard contains all of the keys and switches needed for manually operating the ORACLE. The information keyboard contains all of the hexadecimal characters. An interlock is provided to allow only one key to be depressed at a time. A diagram of the console keyboard is given on page 74.

a. Manual Instructions to the Computer. The console keyboard is used to give an instruction to the computer manually.

Example: Instruct the ORACLE from the console to punch on paper tape the hexadecimal words located in storage cells 000 through 157.

The instruction for punching hexadecimal words from storage beginning at cell 000 to paper tape is
80 000.

The following operations must be executed at the console.

1. Depress the key marked SU. This prepares the information keyboard for receiving the instruction. A light is turned on at KBH.
2. Type 80 000 on the information keyboard.
3. Depress the key marked SPACE. This kills the KBH light.
4. Depress the key marked SO. This will start the output on paper tape.

5. The control counter register, which is displayed on the console, will tell the location of the next word to be punched. When the register reads 158, the key marked OOS should be depressed. This will terminate the punch instruction.

b. Secondary Input. The console keyboard is also used to insert information into the electrostatic storage from the console. The information may be instruction words or numbers.

Example: From the console, insert the hexadecimal word

24 053 5F 1FB into the storage location 3AC.

Execute the following operations at the console.

1. Depress SU.
2. Type 963AC.
3. Depress the key marked SPACE.
4. Depress SO.
5. Type 240535F1FB.
6. Depress SPACE. (The space character indicates that the word is to be inserted in storage.)

CONSOLE KEYBOARD

PTF PAPER TAPE FEED
SU SET UP CR —
SO SPECIAL ORDER /
MS MEMORY SEARCH

OOS OUTPUT ORDER STOP
PCR PAPER TAPE CARRIAGE RETURN
SC SINGLE CYCLE X
OP OPERATE ON CONTINUOUS V

STOP—STOP EVERYTHING CR
BPS BREAKPOINT STOP
KBH KEYBOARD HOT

FIGURE 4

ARITHMETIC REGISTERS

FIGURE 5

CONSOLE TYPEWRITER KEYBOARD

FIGURE 6

4. Computer Console Typewriter

The typewriter associated with the ORACLE console is a standard electric typewriter. The keyboard contains all the alphameric characters, except tape feed and code delete. This typewriter serves as a secondary output, a console monitor, and as a typewriter.

a. Secondary Output. As a secondary output device, it is used to print information directly from the computer. Instructions for directing information to the typewriter may either be contained in the stored program or they may be given manually at the console. Characters such as Rs, Bp, etc., which do not cause the typewriter to function are to be avoided on console typewriter output instructions.

b. Console Monitor. As a console monitoring system, the console typewriter together with the console keyboard gives a permanent record of all manual operations at the console. When an information key at the console is depressed, it is automatically printed by the typewriter. Five of the control or operating keys automatically actuate the typewriter when they are depressed. These keys together with the action produced on the typewriter are given below.

<u>Operating Keys</u>	<u>Action on Typewriter</u>
SU (set up)	The carriage returns and a - is printed
SO (special order)	/ is printed
SC (single cycle)	x is printed
OP (operate on continuous)	v is printed
STOP (stop everything)	Carriage returns

Example: An operator at the console reads a hexadecimal program from paper tape into the electrostatic storage beginning at cell 03B; transfers control to the left of 12C; and then operates on continuous. What will be printed on the paper in the console typewriter?

Operations at Console

1. Depress SU on control keyboard
2. Type 9003B on information keyboard
3. Depress SPACE
4. Depress SO
5. Depress SU
6. Type 4312C
7. Depress SPACE
8. Depress SO
9. Depress OP

Paper in Typewriter

-9003B/ -4312C/V

The console typewriter is also used to type information not normally taken from the computer, such as the date on a monitor sheet in the typewriter.

The ability to use the console typewriter and the console keyboard as an input-output device during the processing of a problem is a great advantage to the programmer. However, it is well for one to remember that

the time required to execute these instructions is relatively great. Also, a type-in instruction is a human operation and an added source of error.

5. Input-Output Instructions.

a. Hexadecimal Information. The space (SPACE) character is used to terminate a hexadecimal word, as well as to initiate a storage action. The carriage return (CR) character is used to terminate an input instruction for one or more hexadecimal words, but it will not initiate a storage action.

On reading hexadecimal information, the Q-register is cleared at the beginning of each word, the information enters Q at the right end and it is shifted left until either a space character or a carriage return character occurs. If this character is a space, the contents of the Q-register will be inserted into the electrostatic storage cell. If this character is a carriage return, the word remains in Q and no storage action is initiated. If one desires that a hexadecimal word be read from the paper tape or the console keyboard into the Q-register with no storage action, he should terminate the word with only a carriage return character. Both long and short hexadecimal words are acceptable.

When hexadecimal information is received from the ORACLE via paper tape, 5/8" magnetic tape, or the console typewriter, the space character is automatically recorded or punched after each word.

b. Alphameric Information. An alphameric character is defined by six binary digits on paper tape. Five of these characters will form a machine word since

each alphameric character is allotted eight binary digits in the machine. The two extra digits allotted to the left of each alphameric character are automatically filled in with zeros.

On reading alphameric information, the Q-register is cleared at the beginning of each word, the information enters Q at the right end and it is shifted left until five alphameric characters have been read. The contents of the Q-register is automatically inserted into the electrostatic storage while the first character of the next word is being read into a buffer register. The Rs character is used to terminate the read instruction for alphameric information. Since no storage action is taken after the Rs character occurs, the Rs character should either be the first character of a new word, or be preceded by four zeros.

Alphameric information from the electrostatic storage is gated to the output unit via the Q-register. When each word enters Q, a parity check bit is recorded in the bit position to the right of Q. By shifting the Q-register 8 bits to the left, each character of the word is sent to the output unit via a short 8 digit register. A parity check bit is then recorded for each character. A sum or parity check bit of the combined characters shifted from Q is also recorded. This combined parity check bit for each word is compared with the parity check bit in Q at the termination of the word or upon the termination of the instruction. If the two parity check bits do not agree, an error light is displayed on the console and the machine stops. The Ps character is used to terminate the

output instruction for alphameric information. When the Ps character reaches the short 8 digit register, the instruction is terminated and the two parity check bits are compared. In order to assure an agreement between these two check bits if no error has occurred, the Engineers advise that either Ps be the last character of a word or that all characters following the Ps character and thus left in Q should be zeros.

c. Instructions. The input instructions are for both the paper tape and the console keyboard. The output instructions are for the paper tape, the 5/8" magnetic tape, and for the console typewriter. These instructions are for both hexadecimal and alphameric characters.

INPUT INSTRUCTIONS

SYMBOL	CODE	DESCRIPTION
IR m	90 m	Loads one or more hexadecimal words from paper tape to the electrostatic storage until a carriage return (CR) character is detected. The Q-register is cleared both at the beginning of the instruction and after each word has been inserted into the storage. The control counter is used.
IK m	92 m	Loads hexadecimal words from console keyboard to the carriage return (CR) character. The operation is the same as for the 90 instruction.
IR ₁ m	94 m	Loads <u>one hexadecimal word</u> from paper tape. The word will be sent to storage if it is terminated with a space character. Since Q is cleared only at the beginning of this instruction, the word remains in Q at the end of this instruction. The word may also be terminated with only a CR character but no storage action will be initiated. The control counter is not disturbed.
IK ₁ m	96 m	Loads <u>one hexadecimal word</u> from console keyboard. The operation of the instruction is the same as for the 94 instruction.
IR _α m	98 m	<p>Loads alphameric information from paper tape. The Rs character terminates the instruction. The Rs character should:</p> <ul style="list-style-type: none"> a. either be the first character of a new word, b. or be preceded by four zeros. <p>The control counter is used.</p>

OUTPUT INSTRUCTIONS

SYMBOL	CODE	DESCRIPTION
OP m	80 m	Punches hexadecimal words from storage to paper tape. A space after each word is automatically punched. The punch stop is controlled manually by a key on the console keyboard (OOS). Two paper tape keys are available on the console keyboard. One (PCR) will punch CR, the other (PTF) is a paper tape feed. The control counter is used.
OM m	81 m	Hexadecimal output via magnetic tape. Operation of the instruction is the same as in 80.
OT m	82 m	Hexadecimal output via typewriter. Operation of the instruction is the same as in 80.
OP ₁ m	84 m	Punch hexadecimal <u>one word</u> . Word will be punched from storage location whose address is specified in the instruction. Space is automatically inserted. The control counter is not disturbed.
OM ₁ m	85 m	Hexadecimal <u>one word</u> output via magnetic tape. Operation of the instruction is the same as in 84.
OT ₁ m	86 m	Hexadecimal <u>one word</u> output via typewriter. Operation of the instruction is the same as in 84.
OP _α m	88 m	Punch alphameric to Ps symbol. All characters in the same word with, but succeeding Ps must be zeros. Ps symbol is not punched. See discussion on page 80. The control counter is used.
OM _α m	89 m	Alphameric output via magnetic tape. Operation of the instruction is the same as in 88.
OT _α m	8A m	Alphameric output via typewriter. Operation of the instruction is the same as in 88.
OP _{α1} m	8C m	Punch alphameric <u>one character</u> . The character to be punched must be on the left and all other characters in the word must be zeros. Control counter is not disturbed.
OM _{α1} m	8D m	Alphameric <u>one character</u> output via magnetic tape. Operation of the instruction is the same as in 8C.
OT _{α1} m	8E m	Alphameric <u>one character</u> output via typewriter. Operation of the instruction is the same as in 8C.

6. The Photographic Curve Plotter for Output [17]

The photographic curve plotter is one of the primary output media for the ORACLE. The unit which is connected to the ORACLE consists of two cathode-ray tubes and a camera together with the control section. The 35 mm camera is mounted in front of one of the two tubes. Information from the internal storage can be displayed on both cathode-ray tubes simultaneously in the form of intensified or brightened spots. The brightened spots displayed on the face of the tubes may be used to form curves, decimal or hexadecimal digits, or alphameric characters. Photographs are made automatically of the brightened spots on the face of the tube to give a permanent record of the information. The face of the second tube, a Memotron, located in the ORACLE console acts as a monitor by providing a rapid visual display of the information. The display of information on the face of the tubes as well as the advancing of the film in the camera are all done under the control of the ORACLE through the stored program. There are four instructions for the curve plotter as indicated in the discussion to follow.

a. Plotting Curves. A display of successive points generates a curve. Each spot or point on the curve is represented by a pair of rectangular coordinates. The face of the cathode-ray tube represents a square array of points. There is a maximum of 1024 positions in the x and y directions. The square array on the face of the tube may be thought of as a representation of the first quadrant of the finite cartesian plane, where both

coordinates of the points are positive. The origin then is in the lower left-hand corner and the upper bound of the points plotted on the tube is 1023 (decimal) or 3FF (hex).

If 1023 is converted to its binary representation 11 1111 1111, one may see that only ten binary digits are needed to represent this upper bound. The ten digits are treated as an unsigned or positive number. Therefore, the "x" and "y" coordinate registers consist of ten toggles each. The "x" and "y" coordinates are stored in the first ten toggles (a_{10} to a_{19} or a_{20} to a_{29}) of the address portion of an instruction for plotting a curve. The point plotting procedure consists of the following instruction:

1. C1 [y], for clearing the "y" register; gating the "y" coordinate into its register; and directing the plotting beam to the correct position on the y-axis.
2. C2 [x], for clearing the "x" register; gating the "x" coordinate into its register; moving the plotting beam along the y line until the correct position is reached on the "x" direction; and after a five microsecond delay, the plotting beam is turned on.

If more than one point is plotted with the same "y" coordinate, it is not necessary to repeat the C1 [y] instruction until a change is made in the "y" coordinate.

Example: Plot the points $(x_1, y) = (12B, 340)$ and $(x_2, y) = (2C0, 340)$.

The coordinates are located in the storage cells as indicated.

The code for plotting the points begins in storage location 170.

<u>Storage Location</u>			<u>Word</u>			
157	12	B00	00	000		x_1
158	2C	000	00	000		x_2
159	34	000	00	000		y
<u>Code</u>						
170	24	157	14	008		
171	54	175	24	159		
172	14	01C	51	174		
173	24	158	14	01C		
174	51	175	C1	[y]		
175	C2	[x_1]	C2	[x_2]		
176	00	000	00	000

If one desires to plot successive distinct points in the x or y direction, an increment of five or more must be used since only approximately 200 spots in any one row or column are distinguishable. A smooth line may be plotted either in the horizontal or vertical direction by selecting an increment of one. Multiple-plotting and spacing of individual spots controls the spot intensity and trace width.

(1) Translating the Coordinate Axes. Since the ORACLE handles the coordinates of a point as unsigned numbers, the coordinate axes must be translated for plotting points in either or both negative directions. That is, the origin is transferred to a new point, with the new axes parallel to the old axes.

The origin (h, k) of the new axes may be selected by examining the negative coordinates in both the x' and y' directions. Set h and k greater than or equal respectively to the largest magnitude of the x' and y' coordinates in the negative direction. The formulas required for translating the new coordinate axes to the old coordinate axes are:

$$x = x' + h$$

$$y = y' + k.$$

Example: Plot the following points: $P_1'(-80, 100)$, $P_2'(-40, 50)$, $P_3'(32, -40)$.

The coordinates of the new points: $P_1:(0, 140)$, $P_2:(40, 90)$, $P_3:(112, 0)$

(2) Scaling the Coordinates. Sometimes the range between the largest and smallest coordinates of a set of points is not sufficient to give a distinct picture of the curve; or, the range between the coordinates of the points may be too great to represent all of the points on one frame. In such cases as the above, the picture of the curve may either be enlarged or reduced. This may be accomplished by multiplying the coordinates by an appropriate scale factor. The new coordinates may be determined by the formulas

$$x = x' \left[\frac{x_{\max} - x_{\min}}{x_{\max}' - x_{\min}'} \right]$$

$$y = y' \left[\frac{y_{\max} - y_{\min}}{y_{\max}' - y_{\min}'} \right]$$

where max and min represent the maximum and minimum of the desired range of the coordinates and max' and min' represent the present range of the coordinates. If one desires to use the whole frame to represent the curve, the above formulas reduce to

$$x = x' \left[\frac{1024}{x_{\max}' - x_{\min}'} \right]$$

$$y = y' \left[\frac{1024}{y_{\max}' - y_{\min}'} \right]$$

The two formulas for translating the coordinate axes and scaling the coordinates may be combined into one formula as follows:

$$x = \left[x' - x_{\min}' \right] \left[\frac{x_{\max} - x_{\min}}{x_{\max}' - x_{\min}'} \right]$$

$$y = \left[y' - y_{\min}' \right] \left[\frac{y_{\max} - y_{\min}}{y_{\max}' - y_{\min}'} \right]$$

If the coordinates of the points are floating point numbers, it is necessary to equalize the exponents so that the coordinates of the plotted points will have the same scale factor.

b. Plotting Digits and Characters. Bright-up spots are used as a tracer to form digits and characters. The control circuitry forces the plotting beam to sweep through a 5 x 8 rectangle for each digit or character. The bright-up spots occur in the 40 spot positions in accordance with a selected

code word previously inserted in the Q-register. This code word is shifted 39 times by giving a multiply instruction. After each shift the beam is turned on if the digit that went from Q_{38} to Q_{39} was a "1". Q_{39} in the code word is not plotted.

Example:

Code word in Q: 1000 0010 0100 0100 0010 1000 1111 1110 0000 0000

(Not Plotted)

The first column of the character is represented by the last group of 8 binary digits, the second column by the next to the last group of 8 binary digits and so on.

Each of the above 40 spot positions in the 5×8 rectangle represents the center of a 4×4 square. This means that the x and y-coordinates are stepped four divisions at a time. The x-coordinate may be started at any desired value. However, since the plotting beam steps 32 (8×4) divisions for each four locations in the x-direction, the carry between 2^4 and 2^5

positions in the y-coordinate register is coupled to the counting toggles of the x-register. Therefore, when the y-coordinate reaches a multiple of 32 (decimal) or 020 (hex), the plotting beam is automatically forced to its original y-coordinate with the beam shifted one step or four spot locations in the x-direction. For the above reason, y must be started at a multiple of 020 (hex). See example 2 below.

The procedure for plotting a character consists of the following instructions:

1. C1 [y] inserts y into the y-coordinate register; and directs the plotting beam to the y-coordinate on the y-axis.
2. 60 [M] inserts the code word [M] in the Q-register.
3. C3 [x] inserts x into the x-coordinate register; and sets the plotting beam on the x-axis at the lower left corner of the letter. The plotter control is now prepared for the multiply instruction to follow.
4. 38 ~ shifts the code word in the Q-register 39 places to the right. The plotting beam is shifted for each shift of Q; and if the digit that went from Q_{38} to Q_{39} was a "1", the beam is turned on.

Example 1: Plot "K" at location (152, 280). The code word to form K and the coordinates are stored in the cells indicated on following page. The code for plotting K begins in storage location 170.

<u>Storage Location</u>	<u>Word</u>				<u>Remarks</u>
157	82	442	8F	E00	code word to form K
158	15	200	00	000	x-coordinate
159	28	000	00	000	y-coordinate
<u>Code</u>					
170	24	158	14	008	
171	54	174	24	159	
172	14	008	54	173	
173	C1	[]	60	157	
174	C3	[]	38	000	
175	00	000	00	000

The following example is given to illustrate what happens when the y-coordinate is not a multiple of 020.

Example 2: Plot "K" at location (150, 290). The same code written for example 1, will be used for this example.

1000 0010 0100 0100 0010 1000 1111 1110 0000 0000 (code word)

When the first group of 4 binary digits has been shifted off the Q-register, the y-coordinate is ready to be increased from 29C to 2A0. However, y cannot be increased to 2A0 since there is no carry between its 2^4 and 2^5 positions. Since the y-coordinate register loses the carry, y is now equal to 280.

$$29C = 0010\ 1001\ 1100$$

$$4 = \quad\quad\quad 0100$$

$$29C + 4 = 2A0 \neq 0010\ 1000\ 0000 = 280$$

The carry is lost.

Some additional remarks and precautions in the use of the C3 instruction are listed.

1. The code word should be put in Q before the C3 instruction, certainly before the following 38 instructions.
2. The beam for plotting characters is stepped by T_1 pulses from the dynamic programmer. Hence, any Σ , Π , or shift operation following a C3 instruction will cause the beam to shift. However, the control of the plotter depends on the carry delay built into multiplication so a 14 $[\sim]$, say, would not be an acceptable substitute for 38 $[\sim]$.
3. The code word in Q should have $Q_{39} = 0$. The DA on the last step of the 38 operation causes the 78th digit of the product to be plotted in position $(x_0 + 014, y_0)$. If Q_{39} in the code word, which is not plotted anyway, is a 0, this will have no practical effect.
4. After plotting a character, the y-coordinate register is left unchanged. In case another character is to be plotted on the same line, it is not necessary to repeat the C1 instruction.

It is necessary to give the C3 instruction each time a character is to be plotted.

5. A character requires 24 positions on the x-axis, 20 positions for plotting plus 4 positions for spacing between characters. Therefore, 42 characters will fill one line.

c. Advancing the Film. Advancing the film exposes a new frame. The exposed frame contains everything that appears on the face of the cathode-ray tube between advance film instructions.

In order to insure a good reproduction of the photographed data, the following precautions should be observed.

1. Advance film one frame before starting to plot the first curve to insure that fresh film is used.
2. Advance film one frame after problem is complete or before leaving computer room to insure that the following program does not ruin your last plot.

A maximum of 1000 photographs can be made before it becomes necessary to replace the magazine of film.

The instruction for advancing the film is C8 ____, where the address portion is irrelevant.

A SUGGESTED KEY WORD LIST FOR CHARACTERS COMMONLY USED
ON THE ORACLE CURVE PLOTTER

0	7C82827C00	Q	7E868A827C	σ	20203C221C
1	0002FE4200	R	629498FE00	φ	3844FE4438
2	62928A4600	S	4C92926400	ω	1C220C221C
3	8CD2A28400	T	8080FE8080	SPACE	0000000000
4	087E08F800	U	FC0202FC00	+	10107C1010
5	8C9292F400	V	E0180618E0	-	1010101010
6	0C92927C00	W	FE083008FE	x	4428102844
7	COA09E8000	X	8244384482	÷	1010541010
8	6C92926C00	Y	C0201E20C0	=	2828282800
9	7C92926000	Z	C2B29A8600	≡	5454545454
				→	1038541010
A	7E88887E00	┐	C0808080FE	←	1010543810
B	6C9292FE00	△	0E32C2320E	<	0044281000
C	4482827C00	π	C692AA682	>	0010284400
D	7C8282FE00	ψ	7008FE0870	/	4020100804
E	829292FE00	∩	7A8680867A	[	008282FE00
F	809090FE00	Σ	C692AA682	]	00FE828200
G	5C92827C00	α	320A3C423C	(	0082827C00
H	FE1010FE00	β	58A4A47E01	)	007C828200
I	0082FE8200	γ	4020107886	}	106C828200
J	FC02020C00	δ	7C92920C00	{	82826C1000
K	824428FE00	δ	0C92B24C00		
L	020202FE00	ε	222A2A1C00	.	0000020200
M	FE201820FE	E	8292543800	,	00001C1200
N	FE1860FE00	θ	7C9292927C	:	0000121200
O	7C82827C00	λ	0618688482	!	0000FA0000
P	609090FE00	μ	F8301830FF	?	40A09A8040

E. Approximate Speeds of the Operations

The table below gives the approximate speed of the operations listed. The times include the access time for both executing the instructions and obtaining the operand if there is one.

Operation	Hex Symbols	Time
Transfers	All 4's	40-50 μ s
Curve Plotter	C1, C2, C3	40-50 μ s
Memory type such as add, store	2's, 5's, 6's, 7's	70 μ s
Shifts	1's	(50 + 8p) μ s
Multiply	38, 39	240-600 μ s (carry time 10 μ s)
Divide	3A, 3B, 3E, 3F	600 μ s
Read, write, hunt per block (2" magnetic tape)	A's, B's	<div> <div> 25 ms information 5 ms start 2 ms stop 18 ms between blocks </div> <div> Total 50 ms/block </div> </div>
Write on 5/8" magnetic tape	81, 85, 89, 8D	<div> 5-10 ms to start tape 1 ms/character </div>
Ferranti reader	90, 94, 98	200 characters/sec.
Film advance	C8	1 sec.
Punch	80, 84, 88, 8C	60 characters/sec.
Output via typewriter	82, 86, 8A, 8E	100 ms/character

F. SUMMARY OF OPERATIONS

Operation			Registers Used			Page Numbers
Timing	Symbol	Code	Accumulator	Quotient	Control Counter	
<u>Shifts</u>						
(50 + 8p) μs	Ro	10	0 →	not used	not used	36
	Ror	11	0 →; α ₃₉ =1	not used	not used	
	L	12	← 0	not used	not used	
	Ly	13	← β ₀	not changed	not used	
	R	14	α ₋₁ →	not used	not used	
	Rr	15	α ₋₁ →; α ₃₉ =1	not used	not used	
	Lc	16	← 0	set to zeros	not used	
	Lcy	17	← 0	set to zeros	not used	
	Roq	18	0 →	... α ₃₉ →	not used	
	Roqr	19	0 →; α ₃₉ =1	β ₀ =1, ... α ₃₉ →	not used	
	Lq	1A	← 0	← α ₀ ...	not used	
	Lqy	1B	← β ₀ ...	← α ₀ ...	not used	
	Rq	1C	α ₋₁ →	... α ₃₉ →	not used	
	Rqr	1D	α ₋₁ →; α ₃₉ =1	β ₀ =1, ... α ₃₉ →	not used	
	Lcq	1E	← 0	set to zeros ← α ₀ ...	not used	
	Lcqy	1F	← 0	set to zeros ← α ₀ ...	not used	
<u>Addition</u>						
70 μs	H	20	* Ac + b → Ac	not used	not used	33
	Hm	21	Ac + b → Ac	not used	not used	

*Ac = Accumulator; b = the number involved in the operation

SUMMARY OF OPERATIONS (Cont.)

Operation			Registers Used			Page Numbers
Timing	Symbol	Code	Accumulator	Quotient	Control Counter	
<u>Addition</u> 70 μ s	H-	22	Ac-b $\rightarrow$ Ac	not used	not used	33
	H-m	23	Ac- b $\rightarrow$ Ac	not used	not used	
	C	24	b $\rightarrow$ Ac	not used	not used	
	Cm	25	b $\rightarrow$ Ac	not used	not used	
	C-	26	-b $\rightarrow$ Ac	not used	not used	
	C-m	27	- b $\rightarrow$ Ac	not used	not used	
<u>Product</u> 240-600 μ s	M	38	product	multiplier product	not used	37
	Mr	39	rounded product	multiplier product	not used	
<u>Division</u> 600 μ s	D	3A	dividend remainder	quotient	not used	42
	Dq	3B	dividend remainder	dividend quotient	not used	
	Dr	3E	dividend remainder	quotient, $q_{39}=1$	not used	
	Dqr	3F	dividend remainder	dividend quotient, $q_{39}=1$	not used	
<u>Transfer</u> 40-50 μ s	Ntr	40	not used	not used	not used	49
	Tr	41	not used	not used	not used	
	Tl	43	not used	not used	not used	
	T+r	48	$\alpha_{-1}=0$	not used	not used	
	T-r	49	$\alpha_{-1}=1$	not used	not used	
	T+l	4A	$\alpha_{-1}=0$	not used	not used	
	T-l	4B	$\alpha_{-1}=1$	not used	not used	
	Tnr	4C	$\alpha_{-1}=\alpha_0$	not used	not used	

SUMMARY OF OPERATIONS (Cont.)

Operation			Registers Used			Page Numbers
Timing	Symbol	Code	Accumulator	Quotient	Control Counter	
<u>Transfer</u>	Tyr	4D	$\alpha_{-1} \neq \alpha_0$	not used	not used	49
40-50 μ s	Tnl	4E	$\alpha_{-1} = \alpha_0$	not used	not used	
	Tyl	4F	$\alpha_{-1} \neq \alpha_0$	not used	not used	
<u>Substitution</u>	α	5-	uses all or parts of Ac	not used	not used	53
	Q	60	not used	$b \longrightarrow Q$	not used	
	q	7-	not used	uses all or parts of Q	not used	
<u>Output</u>	OP	80	not used	used	used	83
	OM	81	not used	used	used	
	OT	82	not used	used	used	
	OP ₁	84	not used	used	not used	
	OM ₁	85	not used	used	not used	
	OT ₁	86	not used	used	not used	
	OP _{α}	88	not used	used	used	
	OM _{α}	89	not used	used	used	
	OT _{α}	8A	not used	used	used	
	OP _{α_1}	8C	not used	used	not used	
	OM _{α_1}	8D	not used	used	not used	
	OT _{α_1}	8E	not used	used	not used	
<u>Input</u>	IR	90	not used	used	used	82
	IK	92	not used	used	used	
	IR ₁	94	not used	used	not used	

SUMMARY OF OPERATIONS (Cont.)

Operation			Registers Used			Page Numbers
Timing	Symbol	Code	Accumulator	Quotient	Control Counter	
<u>Input</u>	IK ₁	96	not used	used	not used	82
	IR _α	98	not used	used	used	
<u>Auxiliary Storage</u>						62
<u>Write</u>	WF	A0	not used	(words) N [*]	used	
	WB	A4	not used	(words) N [*]	used	
<u>Hunt</u>	HF	A8	not used	(blocks) N [*]	not used	
	HB	AC	not used	(blocks) N [*]	not used	
<u>Read</u>	RF	B0	not used	(words) N [*]	used	
	RB	B4	not used	(words) N [*]	used	
<u>Rewind</u>	RW	BC	not used	not used	not used	
<u>Curve Plotter</u>	CY	C1	not used	not used	not used	85
40-50 μs	CX	C2	not used	not used	not used	85
	CC	C3	not used	not used	not used	91
1 Sec.	Fa	C8	not used	not used	not used	95

- 100 -

BIBLIOGRAPHY

1. Walter F. Bauer, An Introduction to High-Speed Digital Computation, Industrial Mathematics Society, 1952.
2. Edmund C. Berkeley, Giant Brains or Machines That Think, John Wiley and Sons, Inc., New York, 1949.
3. Richard M. Bloch, "Mark I Calculator," Proceedings of a Symposium on Large-Scale Digital Calculating Machinery, Harvard University Press, 1948.
4. Bertram Vivian Bowden, Faster Than Thought, Sir Isaac Pitman and Sons, Ltd., London, 1953.
5. Florian Cajori, A History of Mathematics, MacMillan and Co., London, 1919.
6. John W. Carr, III, "Course Notes in ML74," Methods in High-Speed Computation, University of Michigan, Ann Arbor, Michigan, 1952.
7. J. C. Chu, Design of a Computer - ORACLE, ANL-5368 Physics, Argonne National Laboratory, 1954.
8. J. Crank, The Differential Analyser, Longmans, Green and Company, Inc., New York, 1947.
9. First Glossary of Programming Terminology, Report to the Association for Computing Machinery, June 1954.
10. G. E. Forsythe, Selected References on Use of High-Speed Computers for Scientific Computation, MTAC-8, 25, 1956.
11. H. H. Goldstine and J. von Neumann, Planning and Coding of Problems for an Electronic Computing Instrument, Vol. III, Parts I and II, The Institute for Advanced Study, Princeton, New Jersey, 1947.
12. Andrew V. Haeff, "The Memory Tube and Its Application to Electronic Computation," Mathematical Tables and Other Aids to Computation, 3:281-286 (1948).
13. Joseph O. Harrison, Jr., "The Preparation of Problems for the Mark I Calculator," Proceedings of a Symposium on Large-Scale Digital Calculating Machinery, Harvard University Press, 1948.
14. A. S. Householder, "Basic Operations of the ORACLE," Manual for the ORACLE, Oak Ridge National Laboratory, 1953.
15. A. S. Householder, Mathematics and Computing Panel Quarterly Progress Report, ORNL-408, July 1949.

16. Jointly Sponsored by the Navy Department Bureau of Ordinance and Harvard University at the Computation Laboratory, Proceedings of a Symposium on Large-Scale Digital Calculating Machinery, Harvard University Press, 1948.
17. R. J. Klein and C. H. Schalbe, ORACLE Photographic Curve Plotter and Digital Output Device.
18. Mathematics Panel of Oak Ridge National Laboratory, The Oak Ridge Automatic Computer and Logical Engine.
19. Physics and Mathematics, Progress Report on the Argonne-Oak Ridge Digital Computer, ANL-4600, March 1951.
20. Staff of the Computation Laboratory, Description of a Magnetic Drum Calculator, Harvard University Press, Cambridge, Massachusetts, 1952.
21. M. V. Wilkes, D. J. Wheeler and S. Gill, The Preparation of Programs for an Electronic Digital Computer, Addison-Wesley Press, Inc., Cambridge, Massachusetts, 1951.
22. A. S. Householder, Principles of Numerical Analysis, McGraw-Hill, New York, 1953.
23. D. D. McCracken, Digital Computer Programming, John Wiley and Sons, Inc., New York, Chapman and Hall, Ltd., London, 1957.
24. G. R. Stibitz and J. A. Larrivee, Mathematics and Computers, McGraw-Hill, New York, 1953.

Periodicals

25. Journal of the Association for Computing Machinery
26. Mathematical Tables and Other Aids to Computation, MTAC.
27. Computers and Automation.

UNCLASSIFIED