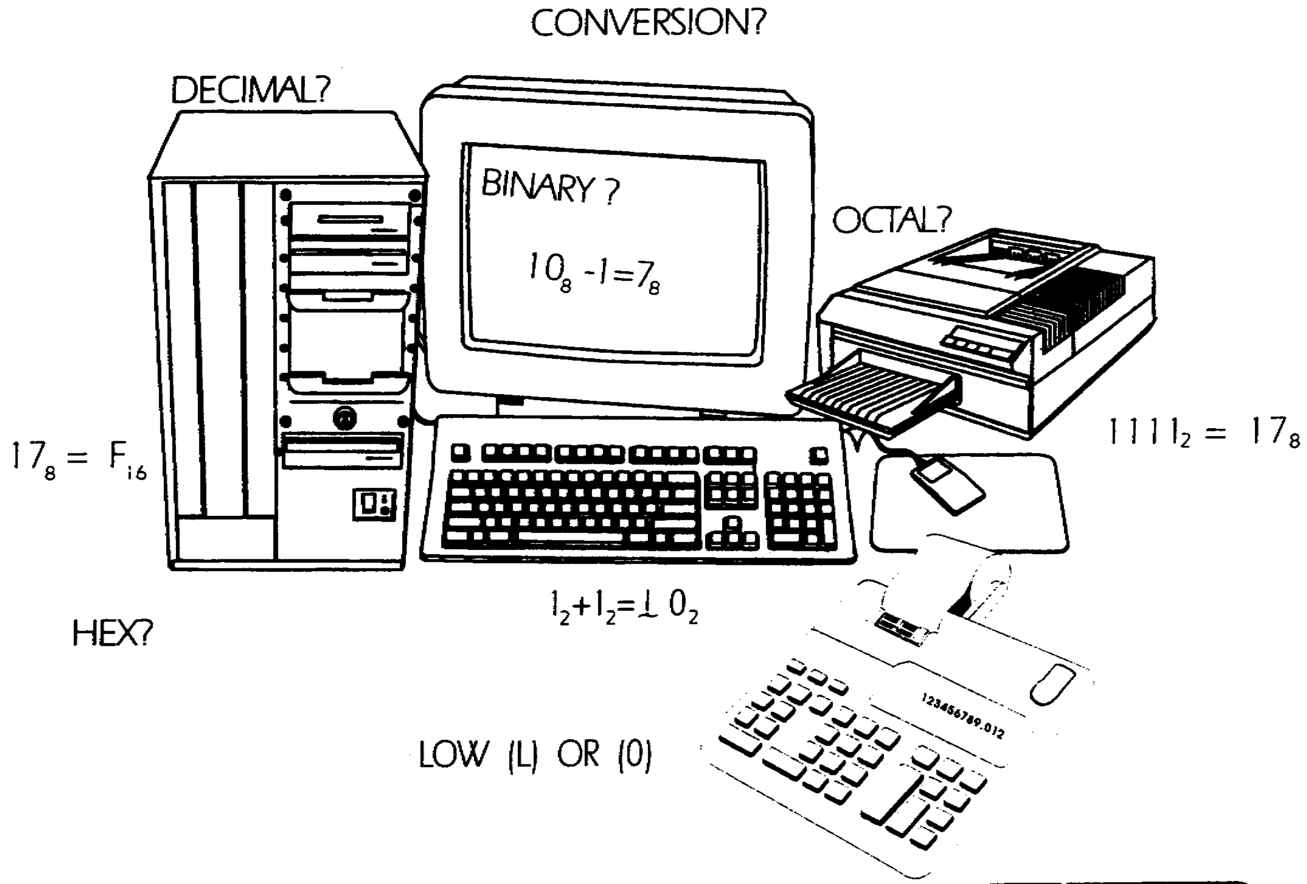
SUBCOURSE
IT0339

EDITION
B

US ARMY INTELLIGENCE CENTER

DIGITAL NUMBERING SYSTEM



DIGITAL NUMBERING SYSTEMS

Subcourse Number IT0339

EDITION B

US ARMY INTELLIGENCE CENTER
Fort Huachuca, AZ 85613-6000

2 Credit Hours

Edition Date: August 1996

SUBCOURSE OVERVIEW

This subcourse will enable you to identify and utilize different numbering systems used in the troubleshooting of computer equipment.

There are no prerequisites for this subcourse.

This lesson replaces SA0709.

TERMINAL LEARNING OBJECTIVE

ACTION:	Correctly count in each system, convert from one system to another, express systems with signed numbers in binary sign magnitude and carry out several arithmetic operations, including 1's and 2's complementing.
CONDITION:	Given various numbering systems including the decimal system.
STANDARD:	To demonstrate competency of this task, you must achieve a minimum of 70% on the subcourse examination.

TABLE OF CONTENTS

LESSON

DIGITAL NUMBERING SYSTEM

CRITICAL TASK: NONE

OVERVIEW

LESSON DESCRIPTION:

Upon completion of this lesson, you will be able to identify and utilize different numbering systems used in the troubleshooting of computer equipment. Computers perform operations using numbering systems other than the decimal system. It is important that you learn to convert the decimal system into a system the computer can understand. A mastery of this subcourse will form a basis for understanding and troubleshooting these circuits.

TERMINAL LEARNING OBJECTIVE

ACTION:	Counting each system, convert from one system correctly to another, express systems with signed numbers in binary sign magnitude and carry out several arithmetic operations, including 1's and 2's complementing.
CONDITION:	Given various numbering systems including the decimal system.
STANDARD:	To demonstrate competency of this task, you must achieve a minimum of 70% on the subcourse examination.
REFERENCES:	Digital Fundamentals, Fifth Edition, FLOYD, Merrill Publishing Co., New York, 1994. The TTL Data Book, Volume 2, Texas Instruments, 1988.

INTRODUCTION

India developed the method of expressing all numbers using 10 symbols, each symbol receiving a value of position as well as an absolute value. Since hands are the most convenient tools nature has provided, man has always tended to use them in his counting. It is both natural and fortunate that our number system is based on 10 digits. The decimal system has been so widely adopted throughout the world that we rarely consider that other number systems exist. A seldom used but very simple system, the binary number system has proved to be the most natural and efficient system for computer use. The decimal system is positional, each numbers position indicating its relative value. Not all systems are positional. The Roman Numeral System is an example of a nonpositional system. Since only positional number systems are used in computers, they are the only ones discussed in this subcourse.

Personnel who work with computers must understand numbering systems because the computer operates only with numbers. Therefore, this subcourse describes several numbering systems and explains how to work with them. By definition, a numbering system is a way of expressing quantity with symbols. The same symbols might express different quantities in any one of several numbering systems, and how to express any given quantity in any given numbering system. Numbers are not expressed as decimal numbers within the computer because other systems are more suitable for machine processes. To understand fully how the computer calculates, you must be able to calculate in the same manner with pencil and paper.

Digital computers use a simple numbering system; it is so simple that it only has two digits: 1 and 0. Digital computers are designed to operate on a YES/NO basis. A digital number is either 1 or 0; a function is there, or it is not there; a diode is conducting or it is not conducting; a voltage pulse is present or not present or it is high state or low state. The numbering systems you will learn to use in this subcourse -binary, octal, and hexadecimal - can all be easily expressed with just 1's and 0's.

PART A

TERMINOLOGY

(1) Radix. The term radix means the number of admissible symbols in a given number system. The admissible symbols are all the characters (Arabic numerals, letters of the alphabet, or other recognizable symbols) used to represent the magnitude of a numerical quantity. In several numbering systems, some or all of the Arabic numerals are the only admissible symbols. The terms numerals, digit, and admissible symbols can be used to mean the same in most cases. There are 10 Arabic numerals. Thus, these symbols will work if the system radix (base) is not greater than 10. In the decimal system, all the numerals are used; the system radix, therefore, is 10 and the radix point is known as the decimal point. The term decimal implies 10 and no other numbering system uses a decimal point.

(2) Symbols and digits. There is no reason all the Arabic numerals must be used to make up the admissible symbols in a numbering system. Less than all of them may be used, all of them plus others may be used, or they may be discarded entirely and other symbols devised. In a positional numbering system, the admissible symbols must have an order of value. This means that the difference in value between adjacent, correctly-ordered symbols must be integral and uniform. One of the symbols must represent zero.

(3) Counting. The digits or symbols in any numbering system must have an order of value. Counting, therefore, is the process whereby the digits are advanced so that the value of an expression increases to the next higher order; that is, advances integrally.

(a) Digit advance. Advancing a digit means replacing it with the digit of the next higher value. In the decimal system, advancing the digit 0 means replacing it with 1; advancing 3 means replacing it with 4; and advancing 9 means replacing it with digit 0 and carrying over a 1, thus advancing the digit to its left. Counting may be more simply defined as the forming and expressing of successive whole numbers. It is the same in all number systems.

(b) Significant digit. In figure 1-1 the right-most digit in each number is the least significant digit (LSD) and the left-most digit is the most significant digit (MSD). When the number of digits in a number increases by one during the process of counting, a breakpoint has been reached. In figure 1-1, breakpoints are shown at the numbers 10 and 100 and 1,000. A breakpoint occurs whenever the value of the number is an integral power of the system's radix. In the decimal system, the breakpoints are 10, 100, 1000, etc., because these are expressions of the integral powers of 10.

	0	16	
	1	17	
	2	etc.	
	3	96	
	4	97	
	5	98	
	6	99	
	7	100	SECOND
		---	BREAK
	8	101	POINT
	9	102	
FIRST	10	etc.	
BREAK ---			
POINT	11	998	
	12	999	
	13	1000	THIRD
		---	BREAK
	14	1001	POINT
	15	1002	

Figure 1-1

Breakpoint in the Decimal System.

PART B

Numbering Systems, Operations and Codes

1. The Decimal System.

a. General. In digital technology there are many numbering systems in use. The most common systems are decimal, binary, octal, and hexadecimal. The decimal system is clearly the most familiar to us since we use it every day. Reviewing some of its characteristics will help us to better understand the other numbering systems.

b. Characteristics. The decimal system, also called the base 10 system, is composed of ten numerals or digits. They are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and can be used to express any quantity. Decimal numbers greater than nine are formed by placing numbers in columns to the left of the first digit. Each column has 10 times the value of the column immediately to its right. This method of representing decimal numbers is called positional notation. The value of a digit depends on its position. For example, consider the decimal number 357. This number can be written as 300 + 50 + 7 or in scientific notation:

$$(3x10^2) + (5x10^1) + (7x10^0).$$

In essence the three carries the most weight and is considered the most significant digit (MSD). The seven carries the least weight and is called the least significant digit (LSD). Consider as another example the number 29.35. This number is actually equal to 20 + 9 + .3 + .05 or in scientific notation:

$$(2x10^1) + (9x10^0) + (3x10^{-1}) + (5x10^{-2}).$$

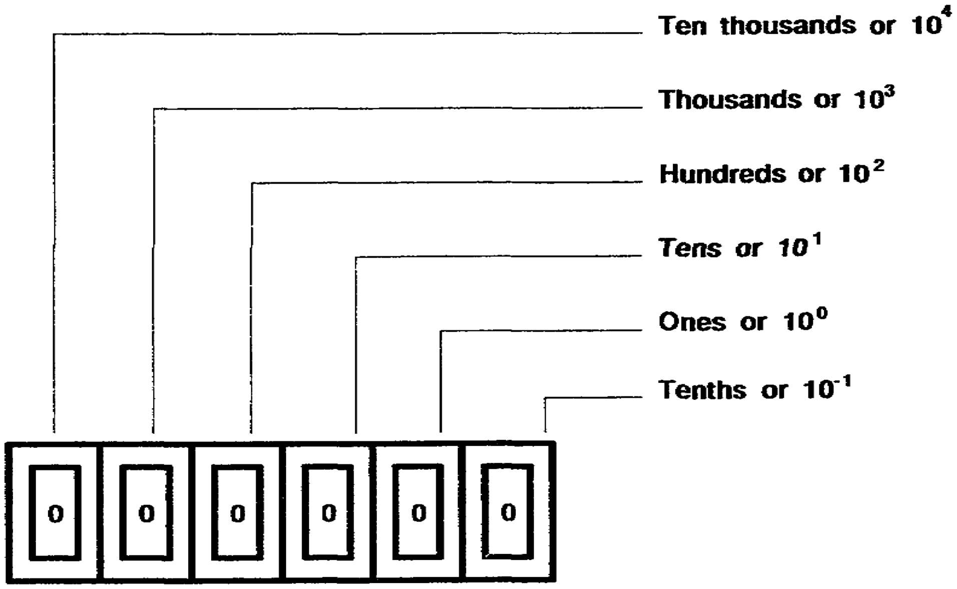
Consider another example, the digit 3 has a different value in each of the following base 10 numbers:

463 731 302

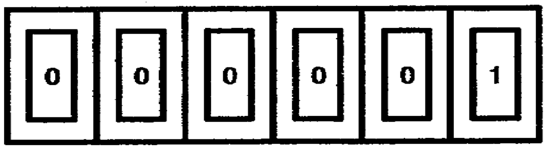
In the right hand position, 3 has the value of 3. In the middle position, 3 has the value 30. In the left position, 3 has the value 300. 3, 30, and 300 can all be represented by multiplying 3 by the base 10 raised to a power:

$$3 = 3 \times 1 = 3 \times 10^0$$
$$30 = 3 \times 10 = 3 \times 10^1$$
$$300 = 3 \times 100 = 3 \times 10^2$$

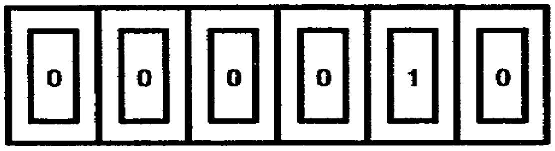
Consider as another example, the odometer of a car:



As the automobile travels one tenth of one mile the tenth digit advances one tenth:



Two Tenths:
and so on up to nine tenths. Once the automobile has traveled one mile, the digit in the tenths (or 10^{-1}) position returns to zero and the digit in the ones (or 10^0) position becomes a one:



Thus the automobile has traveled one mile and the odometer has been incremented by a power of ten. This process will continue as the automobile travels farther and farther. The 10^0 position would be increased to nine and then zero and the tens (or 10^1) position would be increased to a one.

(c) Sequence of count Figure 1-2 shows the sequence of a count to the first breakpoint in several number systems. Some of these numbering systems, such as the binary and octal systems, are commonly used in computer systems; hence, they are discussed in detail in this subcourse. The first breakpoint occurs at 10 in all the number systems shown, but the value of 10 is entirely ambiguous unless the radix is designated. This is clarified by using the radix as a subscript number. For example 10_{10} must be read as one zero, not ten. In all computer work, the agreed-upon practice is to write and read all subscripts in the decimal system. This prevents confusion, for if the radix subscripts were written in the same system as the number itself, all subscripts would be 10. For example, a radix value of 2 written in the binary system is 10, a radix value of 8 written in the octal system is 10, and so on.

SYSTEM	RADIX	COUNT
Binary	2	0, 1, 10
Ternary	3	0, 1, 2, 10
Quaternary	4	0, 1, 2, 3, 10
Quinary	5	0, 1, 2, 3, 4, 10
Sexternary	6	0, 1, 2, 3, 4, 5, 10
Septenary	7	0, 1, 2, 3, 4, 5, 6, 10
Octal	8	0, 1, 2, 3, 4, 5, 6, 7, 10
Nodal	9	0, 1, 2, 3, 4, 5, 6, 7, 8, 10
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Unidecimal	11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, 10
Duodecimal	12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, 10
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10

Figure 1-2

Sequence of a Count to the First Breakpoint

(Several Numbering Systems)

2. The Binary System.

a. General. Unfortunately the decimal system does not lend itself to convenient implementation in digital systems. It is very difficult to design equipment so it works with ten different voltage levels, each one representing one of the decimal characters 0 through 9. On the other hand, it is very easy to design electronic circuits that operate with only two different voltage levels. For this reason the binary (base 2) numbering system was developed. It is the basic system for operation of almost all digital systems, although other systems are often used in conjunction with it.

b. Characteristics. The binary number system has two binary digits called bits which are assigned the symbols 0 and 1. The binary system is also a positional value system. Each binary digit has its own value or weight expressed as a power of two. An example of a binary number is 1011 which is read as ONE-ZERO-ONE-ONE. The number 1301 is not a binary number, since 3 is not a binary digit. The right-most digit is the least significant digit (LSD) and carries the least weight while the left-most digit is the most significant digit (MSD) and carries the most weight.

c. Conversion of Binary.

(1) General. In computer work, it frequently happens that a value stated in one number system has to be converted to that same value in another number system. Unless conversions are made accurately, computer operations and output become meaningless.

(2) Positional Notation. By conversion agreement, a radix point is not written except when necessary to separate the integral and fractional parts of an expression. In writing 100.00, only 100 is written. The .00 portion is ordinarily discarded because it is meaningless and unnecessarily uses up computer capacity. On the other hand, the radix point is needed in writing numbers such 11.1 because the number contains both integral and fractional parts. If the radix point were omitted, the expression would be understood as 111, which is entirely different from the original number. In fact it equals 11.1 multiplied by the radix. Further, moving the radix point one position to the left in 11.1 yields 1.11 and this quantity is equal to 11.1 divided by the radix. This property of the radix point position is the basis of the powers-of-ten method of simplifying the multiplication and division of decimal numbers. This concept will be put to use later in this text to aid in converting from decimal to binary and decimal to octal. Most digital computers are built to handle their internal operations in the binary system, while reading in and out in the decimal system. This requires the computer to convert, with correct positional notation, all of its inputs and output.

(a) The table in Figure 1-3 covers conversion of binary numbers to decimal numbers and conversion of decimal numbers to binary numbers.

Position of digit	5	4	3	2	1
Power of two	2^4	2^3	2^2	2^1	2^0
Decimal value 16	8	4	2	1	
Binary Coefficients	1	1	0	0	1

Figure 1-3

DECIMAL	BINARY	DECIMAL	BINARY
1	1	11	1011
2	10	12	1100
3	11	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10010
9	1001	19	10011
10	1010	20	10100

Figure 1-4

Binary Counting.

(b) Binary counting. Figure 1-4 lists the first 20 binary numbers. While the same positional notation system is used, the decimal system uses powers of 10, and the binary system uses powers of 2. The number 125 actually means $(1x10^2) + (2x10^1) + (5x10^0)$. In the binary system, the same number (125) is represented as 1111101, meaning $(1x2^6) + (1x2^5) + (1x2^4) + (1x2^3) + (1x2^2) + (0x2^1) + (1x2^0)$.

(3) Conversion by Table. A table must have the required range of numbers that are to be converted. Such a table can be constructed to any desired length. When you make a binary table you must use the power of 2 (Figure 1-4). Using Figure 1-4 you can convert decimal numbers from 0 to 20. To represent the decimal value of 0, all positions of the binary coefficients would be 0. To represent the decimal value of 1,023, all positions of the binary coefficients would be 1. In Figure 1-3 the example of the binary coefficients equals a 25_{10}. The process of conversion is shown below:

$$11001_2 = (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

$$16 \quad + \quad 8 \quad + \quad 0 \quad + \quad 0 \quad + \quad 1$$

$$11001_2 = 25_{10}$$

(4) Conversion of Binary to Decimal. The converted answer can be read directly from Figure 1-4 by looking for the most significant digit (MSD) of the binary coefficients. Thus, binary 1 (the MSD) is found in digit position 5 which equals 16 in the decimal system. The next binary 1 is found in digit position 4 and is worth 8 in the decimal system. The remaining binary 1 is found in position 1, and is equal to 1. When the decimal values are added, the total is 25_{10}.

(5) Table Construction. To construct a conversion table, the values and radix of the system must be known. The value range determines the number of digit positions. The radix determines the base of the number and to what power the number must be raised. In the construction of the binary table (Figure 1-4), the power of 2 is used because 2 is the radix of the binary system. The conversions you will need to make are nearly always to the decimal system. Figure 1-4 is a conversion table that can be used for binary-to-decimal or decimal-to-binary conversion.

(6) Conversion by Division. Using tables can be a tedious and laborious method for converting numbers. A better method, especially for larger numbers, is division. The decimal number is repeatedly divided by 2, and the remainder after each division is used to indicate the coefficients of the binary number to be formed. Note that the binary number derived is written from the bottom up. Here are some examples that show how to use the division method in converting from decimal to binary.

(a) Example 1:

(1) Convert 125_{10} to the binary system.

125 / 2 = 62 + remainder of 1

62 / 2 = 31 + remainder of 0

31 /2 = 15 + remainder of 1

15 /2 =7 + remainder of 1

7 /2 =3 + remainder of 1

3 / 2 = 1 + remainder of 1

1 / 2 = 1 + remainder of 1

(2) Solution: 125_{10}=1111101_2

(3) The common way of solving this problem is as follows:

```
2) 125 (1
2)  62 (0
2)  31 (1
2)  15 (1
2)   7 (1
2)   3 (1
2)   1 (1
     0
```

Read the binary number from bottom up: 1111101

(b) Example 2:

(1) Convert 70_{10} to the binary system.

```
2) 70 (0
2) 35 (1
2) 17 (1
2)  8 (0
2)  4 (0
2)  2 (0
2)  1 (1
    0
```

(2) Solution: $70_{10} = 1000110_2$

(7) Checking the Division Method. This checking method is also the way of converting from binary to decimal using the appropriate conversion table.

Checking the results of the two examples give the following:

$1 \times 2^6 = 64$	$1 \times 2^6 = 64$
$1 \times 2^5 = 32$	$0 \times 2^5 = 32$
$1 \times 2^4 = 16$	$0 \times 2^4 = 16$
$1 \times 2^3 = 8$	$0 \times 2^3 = 8$
$1 \times 2^2 = 4$	$1 \times 2^2 = 4$
$0 \times 2^1 = 0$	$1 \times 2^1 = 2$
$1 \times 2^0 = \underline{1}$	$0 \times 2^0 = \underline{0}$
125_{10}	70

3. The Octal Numbering System.

a. Unfortunately binary numbers tend to be long and cumbersome. Binary digits can be grouped into sets of three or four to form octal (base 8) or hexadecimal (base 16) numbers respectively. The octal numbering system uses only the digits zero through seven (base 8). The table in Figure 1-5 lists the binary numbers zero through seven and their octal equivalents using the group of three method.

THREE BINARY DIGITS	OCTAL DIGIT
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Figure 1-5

Group of Three Method.

b. The primary use of octal is in recording values stored in binary registers. There is a simple trick for converting a binary number to an octal number. Simply group binary digits into groups of three starting at the binary point, and read each set of three binary digits according to figure 1-5. This group of three method of conversion is simple. It merely requires you to memorize the binary to octal conversions from 0 to 7 to cover the octal radix of 8. For example, to convert the binary number 011101, we first break it into threes (011 101). Then we convert each group of three binary digits, getting 35 in octal as a result. Therefore, $011101_2 = 35_8$.

(1) Example 1: $111'110'111_2 = 767_8$

7 6 7

(2) Example 2: $110'110'101_2 = 665_8$

6 6 5

(3) Example 3: $11'011_2 = 33_8$

3 3

c. There are several other shortcuts for converting octal to decimal and decimal to octal, manually and using tables. Once you have converted the number to binary you can then convert to decimal or octal from depending on what base you are converting from. Tables are as convenient as any other method. Octal-to-decimal-to-octal tables are readily available in the military or commercial manuals for specific computers. An important use for octal is in listing of programs and for memory dumps for binary machines, thus making printouts more compact. In keeping track of the contents of registers and in orally conveying the contents of a register to someone, it is very useful to use octal characters rather than binary.

(1) Example 1: $476_8 = 4 \quad 7 \quad 6_8 = 100111110_2 = 318_{10}$

$100\ 111\ 110_2$

(2) Example 2: $010_8 = 0 \quad 1 \quad 0_8 = 10002 = 8_{10}$

$000\ 001\ 000_2$

4. The Hexadecimal Numbering System.

a. General. The hexadecimal system, often referred to as hex, is a base 16 number system. The symbol used in the hexadecimal system are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

b. Hexadecimal Conversion. All the IBM 360 series of machines have their memories organized into sets of bytes (a byte consists of eight binary digits). Each byte either is used as a single entity to represent a single alphanumeric character (a combination of letter and numbers) or is broken into two four-bi pieces. The group of four method is another shortcut for converting binary to hexadecimal, and the reverse. This method is based on the equivalency of any four-bit binary group to a particular hexadecimal digit as shown in figure 14.

(1) For binary-to-hexadecimal conversions, the binary number is divided into four-bit groups, beginning at the radix point; the hexadecimal equivalent of each group is then written.

1010101010110001_2

$1010'1010'1011'0001_2$

A A B 1_{16}

$1101'1110'1010'1111_2$

D E A F_{16}

BINARY	HEXADECIMAL	DECIMAL
0000	0	0
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

Figure 1-6

Binary-to-Hexadecimal-to-Decimal Conversion.

(2) Hexadecimal-to-binary conversions are an exact reverse, wherein the equivalent four-bit group is written for each hexadecimal digit.

2_{16}	A_{16}	B_{16}	E_{16}
0010_2	1010_2	1011_2	1110_2

5. Arithmetic Operations.

a. General. Now that you know how to count and convert, the next step is to perform addition and subtraction in nondecimal numbering systems. To understand how the computer computes, you must understand these mathematical operations and how to perform them. The first task is to learn the machine methods of performing arithmetic computations. Knowledge of how the computer is designed is not required to use these methods. Although this chapter introduces some computer terminology and theory, the primary concern is with pencil and paper solutions that result when you use the same methods that computers use.

b. Rules for Addition. The rules taught in elementary school for adding decimal numbers can be used in adding numbers in any system. The only differences are the breakpoints in the nondecimal systems. The counting and breakpoint for carrying is the important operation. In the following descriptions of addition, we will emphasize the binary system.

(1) The rules of binary addition are simple because of the small radix. The rules differ from regular decimal addition only when the radix or breakpoint is reached.

(a) Adding 0 and 0 yields 0:

$$\begin{array}{r} 0 \\ \underline{+0} \\ 0 \end{array}$$

(b) Adding 0 and 1 or 1 and 0 yields 1:

$$\begin{array}{r} 0 \\ \underline{+1} \\ 1 \end{array} \qquad \begin{array}{r} 1 \\ \underline{+0} \\ 1 \end{array}$$

(c) Adding 1 and 1 yields 0 with a carry of 1. This rule shows the point at which the radix of the binary system is reached:

$$\begin{array}{r} 1 \\ \underline{+1} \\ 10 \end{array}$$

(2) Single addition. Using the rules for adding binary numbers, the following are examples of how to add and how to check the results in the decimal system.

Binary	Decimal
10	2
+01	+1
11	3

(a) In this example, adding 0 and 1 yields 1; adding 1 and 0 yields 1. This gives the correct answer of 11 (note that the least significant digit is to the right). The check in decimal numbers yields an answer of 3 which equals binary 11.

Binary	Decimal
11	3
+11	+3
110	6

(b) In the second example of the carry, adding 1 and 1 yields 0 with a carry of 1 to the next position to the left. In the third (most significant digit) position, 1 and 1 plus the carry of 1 results in a 1 with a carry of 1. In the examples below the carry of 1 is reemphasized:

Binary	Decimal
111	7
+101	+5
1100	12
1011	11
+0111	+7
10010	18

(3) Multiple addition. When several binary 1's appear in the same column, the rules still apply but the carries sometime become more complex. Actually, it is easier to use the following as an aid. When an odd number of 1's is added, the sum is 1; when an even number of 1's is added, the sum is 0. The total number of breakpoints of the radix are carried to the next position as 1's. The following examples show this procedure.

Binary	Decimal
101	5
011	3
+110	+6
1110	14

(a) In this first addition, there is an even number of 1's, causing a carry of a 1 with a 0 entered in the sum. In the second position, there is an addition of 1 and 1 plus the carry of 1, making the total odd which gives a 1 in the sum and a carry of 1. In the last position, there is another odd addition; therefore, a sum of 1 plus a carry of 1.

Binary	Decimal
1011	11
0101	5
0011	3
+0001	+1
10100	20

(b) In the first addition, the number of 1's is even, causing a 0 to be entered with two 1's being carried. The second position has two 1's and two 1's carried making the addition even, with a 0 entered as the sum and two 1's carried again. In the third position, there is one 1 and two 1's carried, making the addition odd, with a sum of 1 and a carry of 1. In the fourth position, there is a 1 and 1 carried making the addition even, with a 0 entered for the sum and the carry 1 entered as the MSD. Remember, in any addition of binary numbers, each time the radix (2) is reached, there is a carry of 1 to the higher order position. The following examples further illustrate this rule.

Example 1:

Binary	Decimal
1011	11
1001	9
0101	5
+0111	+7
100000	32

Example 2:

1101	13
1100	12
1011	11
+1001	+9
101101	45

(c) In Example 2, the sum in the rightmost binary column is a 1 with a 1 carry; in the next column to the left, the sum is 0 with a carry; the next column's sum is 1 with a 1 carry.

In the final column, a trick can be applied to keep the carry straight. If the column is divided in half, then a 1 is written as the sum and two 1's are carried as shown:

0 with 1 carry 1
1

0 with 1 carry 1
1
1 Carry from prior column

The two 1's carried now total 10 which are the final two digits.

c. Subtraction by Complementation. Most computers can only add and must use other shortcuts to accomplish other arithmetic operations. Subtraction is done by adding the complement. Certain adjustments are necessary.

(1) Decimal complement. To understand the complement and how it is used in computer subtraction, you must first fully understand the decimal complement. The 10's complement (true complement) of a number is that quantity which, when added to a number, will total the next breakpoint or power of 10.

Decimal number		10's complement	Next breakpoint
7_{10}	+	3_{10}	10_{10}
10_{10}	+	90_{10}	100_{10}
170_{10}	+	830_{10}	1000_{10}
540_{10}	+	460_{10}	1000_{10}
783_{10}	+	217_{10}	1000_{10}

(2) Decimal complement subtraction. This process involves the addition of a number and the complement of the second number being subtracted. This achieves subtraction by addition.

(a) Subtraction by the 10's complement. A number (subtrahend) may be subtracted from another number (minuend) by adding the 10's complement of the subtrahend to the minuend.

```
minuend       42     42
subtrahend   -39    +61   (10's complement of 39)
              3     103
```

The difference between 42 and 39 is not 103. With this method, an extra-order 1 is generated. This 1 signifies the answer is a positive number and is dropped (103 - 100 = +3). The answer then is +3 which is the correct difference between 42 and 39.

Here are some other examples of this method.

```
56        56
-8       +92   (10's complement of 8)
         148 = +48

738       738
-46      +954   (10's complement of 46)
         1692 =  +692
```

(b) These two examples involved positive answer. Let's now consider the situation in which the subtrahend is larger than the minuend. The difference in such cases will be a negative quantity. This demonstrates a negative difference.

```
 150       150
-172      +828      (10's complement of 172)
           978
```

Note that in this example, an extra-order 1 was not generated; this means the answer is a negative number. To find the answer, the 10's complement of 978 must be found and written with a negative sign.

1000 - 978 = 22 (10's complement)

The answer is -22. As you can see from this example, when you are dealing with a negative answer, you must find two complements. The extra-order 1 must be generated before the end-around-carry can be performed. If there is no extra-order 1, the answer is a negative number.

d. Binary Complement. A digital computer that uses the binary number system will use the binary complement system. In the binary system, the two methods are: 2's complement (no end-around-carry) and 1's complement (end-around-carry). First consideration will be given to finding the 2's complement of any binary number. The 2's complement is performed by changing each digit to is opposite and adding 1 to the least significant digit (LSD).

Number	Opposite +1	2's Complement
1	0+1	1
10	01+1	10
101	010+1	011
111	000+1	001
11000	00111+1	01000

Finding the 2's complement of any binary number can be simplified even further by starting at the least significant bit (binary digit) position and inspecting each bit position for a 1. The least significant 1 and all 0s to its right are copied without change. All bits to the left of the least significant 1 are changed. Of course, identical results are obtained with the two methods.

Number	2's Complement
100100	011100
111110	000010
001001	110111
010000	110000

(1) Binary Complement Subtraction. Binary complement subtraction is performed by complementing the subtrahend and adding. The method of 2's complement for subtraction is illustrated below.

```
 11        11
-01       +11   (2's complement of 01)
          110 = +10₂ (changing extra-order 1 to a +)
```

(2) One's Complement Subtraction. The same basic steps for subtraction apply in one's complement subtraction that are applied in two's complement subtraction, with very little change in the procedure. To one's complement a number, just turn the ones into zeros and the zeros into ones.

01101	given
10010	one's complement

To subtract the following:

```
 10110    (1)  copy the top                 10110
-00110    (2)  one's complement the bottom  11001
          (3)  ADD                         101111
```

NOTE: IF THE CARRY BIT IS 1, THEN END AROUND CARRY AND ADD. THE ANSWER IS POSITIVE. IF THE CARRY BIT IS 0, THEN ONE'S COMPLEMENT THE ANSWER. THE ANSWER IS NEGATIVE.

```
  10110
 +11001
 101111
 +    1
  10000
```

Using these two numbers and subtracting with the ones complement method, we find NO CARRY BIT.

```
  10010      10010
 -11010     +00101
             10111
```

We must then one's complement the answer, and place a negative sign before the number.

10111 -1000: THE ANSWER

e. The last objective to learn in this text is to convert fractions expressed in decimals and their binary equivalent from one base to the other. Base 10 fractions can be expressed by numbers with a decimal point, for example, 3/4 is the same as 0.75. similarly, base 2 fractions can be expressed in numbers following a binary point, for example, $1/10_2$ is the same as 0.1_2.

In the base 10 system, numbers to the left of the decimal point are whole numbers, and numbers to the right of the decimal point are fractions. The same thing is true in the binary system, or for that matter for any other number base.

Look at the binary numbers to the left of the binary point first (Figure 1-7):

2^6	2^5	2^4	2^3	2^2	2^1	2^0	Power value
64	32	16	8	4	2	1	Base 10 value

Figure 1-7

Starting from the right side of the chart, the base 10 value of each binary digit is shown. A binary 1 in the righth and position next to the binary point will have a base 10 value of 1. A binary 1 in the next position has a base 10 value of 2. A binary 1 in the third position from the right has a base 10 value of 4, and each successive digit doubles in value.

Another way to calculate the base 10 value of a binary number is to add up the base 10 value of each of its digits. The binary number 1111111 has a base 10 value of 64 + 32 + 16 + 8 + 4 + 2 + 1 or 127_{10}.

Since each value to the left has a value twice the preceding one, then each value to the right has a value of half the preceding one. This relationship continues to the right of the binary point, too. Thus, the first value to the right of the binary point is 1/2, the next 1/4, and so on. This is shown on the chart in figure 1-8 below:

Power value Base 2	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Base 10 value	16	8	4	2	1	1/2 or .5	1/4 or .25	1/8 or .125	1/16 or .0625

Figure 1-8

To convert a binary fraction to its base 10 value, add the base 10 values for each binary 1 in the expression. For example, to convert 0.101_2 to its base 10 value, add the values .5 and .125. The result is .625. Convert the following binary fractions to their base 10 value and check your answers on the next page:

a. .0100

b. .1010

c. .0110

d. .0111

e. .1100

f. .1111

g. .00001 (HINT: This one requires you to figure out the next value in the series.)

ANSWERS to problems on page 1-22:

a. .25
b. .625
c. .375
d. .4375
e. .75
f. .9375
g. .03125 (This value is equal to 1/32 or can be determined by dividing the preceding value of .0625 by 2.)

To convert a base 10 fraction (decimal) to a binary fraction, multiply the fraction by 2. Remove the value to the left of the decimal (whether 0 or 1), and continue to multiply by 2 and remove the whole number, until no fraction remains, or until you have enough places for your purpose. The 0's and 1's you remove, written in order to the right of the binary point, become the resulting binary fraction.

For example, convert .625 to a binary fraction:

Multiply by 2: $\begin{array}{r} .625 \\ \text{X } \underline{\ \ 2} \\ 1.250 \end{array}$

Remove the whole number: 1

Multiply by 2: $\begin{array}{r} .250 \\ \text{X } \underline{\ \ 2} \\ 0.500 \end{array}$

Remove the whole number: 0

Multiply by 2: $\begin{array}{r} .500 \\ \text{X } \underline{\ \ 2} \\ 1.000 \end{array}$

Remove the whole number: 1

List the numbers removed to the right of the binary point: .101

Following the procedure shown above, convert .8125 to a binary fraction. Check your answer on the next page.

ANSWERS to problems on page 1-23: $.1101_2$

SOLUTION

```
.8125
  X 2
------
1.6250      >      .1

 .6250
   X 2
------
1.2500      >       1

 .2500
   X 2      >       0
------
0.5000

 .5000
   X 2      >       1
------
1.0000
```

$.8125_{10} = .1101_2$

Convert the following base 10 fractions to binary fractions. When you have finished, check your answers on the next page.

a. .625

b. .675 (just carry this problem and the next one to 6 places. Indicate that it is unfinished by putting a + after the last digit.)

c. .01

ANSWERS to problems on page 1-24:

a. .1010
b. .101011+
c. .000000+

LESSON

PRACTICE EXERCISE

1. How many different single-digit numbers are there in the octal system?

2. In the number $17{,}206_{10}$, the digit 7 is multiplied by ten raised to what power?

3. In the number 4381_{10}, the digit 8 is multiplied by 10 raised to what power?

4. In the number 4617_{10}, the digit 7 is multiplied by ten raised to what power?

5. If you count up from 0 in the octal system, how would you write the number after 7?

6. What is the highest single-digit value in the base $_{16}$ (or hexadecimal) numbering system? Answer with a base ten number, unless you know the hexadecimal notation already.

7. In the base ten system, if you add 1 to 99, the result is 100. Using the octal (base 8) system, how would you write the number after 77_8?

8. Convert the following base ten numbers to octal:

a. 372_{10} b. 746_{10}

9. Complete the table for the values 13 through 20.

Decimal	Binary	Hex	Decimal	Binary	Hex
0	0000	00	11	1011	0B
1	0001	01	12	1100	0C
2	0010	02	13		
3	0011	03	14		
4	0100	04	15		
5	0101	05	16		
6.	0110	06	17		
7.	0111	07	18		
8.	1000	08	19		
9.	1001	09	20		
10.	1010	0A			

10. a. Hex is the short form for hexadecimal. How many hex digits are needed to express a, 16-bit word in a computer?

    b. Convert the following binary numbers to hex.

binary:	1011	0101	1101	0011
hex:	?	?	?	?

11. Convert the following to binary:

    a. 9_{10} b. 23_{10}

12. Convert the following examples to base ten:

    a. 272_8 b. 164_8

13. Convert the following binary numbers to base 10 numbers.

    a. 1100101_2 b. 1111100_2 c. 0010011_2

14. Convert each of the numbers shown below to base ten.

    a. 11101_2

    b. 100011_2

    c. 77_8

    d. 21_8

15. Convert the following base 10 fractions to binary fractions.

    a. .625

    b. .675

    c. .01

LESSON

PRACTICE EXERCISE

ANSWER KEY AND FEEDBACK

Item	Correct Answer and Feedback
1.	8
2.	3, repres. by 7 x10^3
3.	one (8 x 10^1)
4.	zero (7 x 10^0)
5.	The number after 7 in the octal system is written as 10. It has the same value as the base ten value 8, but since 8 is not included in the octal system, it must proceed to the lowest 2-digit number, or 10.
6.	15_{10} is the value of the highest single-digit number in the base 16, or hexadecimal, numbering system. If you were familiar with the notation already, you could have answered "F', which is the usual hexadecimal notation for the value 15_{10}. You could also have written ">F" or "Fh", which are both common ways of showing that F is in the base 16 numbering system.
7.	100_8
8. a.	564_8
b.	1352_8

9.

Decimal	Binary	Hex	Decimal	Binary	Hex
0	0000	00	11	1011	0B
1	0001	01	12	1100	0C
2	0010	02	13	1101	0D
3	0011	03	14	1110	0E
4	0100	04	15	1111	0F
5	0101	05	16	10000	10
6.	0110	06	17	10001	11
7.	0111	07	18	10010	12
8.	1000	08	19	10011	13
9.	1001	09	20	10100	14
10.	1010	0A			

10.	a.	1 hex digit equals 16 bits
	b.	B 5 D 3
11.	a.	1001
	b.	10111
12.	a.	186
	b.	116
13.	a.	101_{10}
	b.	124_{10}
	c.	19_{10}
14.	a.	29
	b.	35
	c.	63
	d.	17
15.	a.	.1010
	b.	.101011+
	c.	.000000+

This is the end of the text. The examination follows. Review any material you are unsure of, and make sure you have actually worked the problems, not just looked up the answers. When you are ready, take the examination. You may refer back to the text any time during the examination.

THIS PAGE LEFT INTENTIONALLY BLANK