HEATHKIT
CONTINUING
EDUCATION

Individual Learning Program

MICROPROCESSORS

Unit 1
NUMBER SYSTEMS AND CODES

EE-3401

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022
595-2039-02

Copyright © 1977
Heath Company
All Rights Reserved
Printed in the United States of America

CONTENTS

Unit 1

NUMBER SYSTEMS AND CODES

INTRODUCTION

The purpose of this first unit on microprocessors is to give you a firm foundation in number systems and codes. Binary numbers and codes are the basic language of all microprocessors. Octal and hexadecimal numbers allow easy manipulation of binary numbers and data. Thus, a good foundation in numbers and codes is essential to understanding microprocessors.

This unit will reacquaint you with the decimal number system, then expand the basic concept of numbers to the binary, octal, and hexadecimal systems. Understanding these systems fully will help you understand the many digital codes used with microprocessors. Although this unit can only give you a working knowledge of numbers and codes, you will become more proficient with them as you proceed through the units that follow.

A listing of number system tables has been provided in Appendix B of this course. Appendix B is located at the back of the second binder.

Examine the Unit Objectives listed in the next section to see what you will learn in this unit. Then follow the instructions in the Unit Activity Guide to be sure you perform all of the steps necessary to complete this lesson successfully. Check off each step as you complete it and, in the spaces provided, keep track of the time you spend on each activity.

UNIT OBJECTIVES

When you complete this unit you will have the following knowledge and capabilities:

1. Given any decimal number, you will be able to convert it into its binary, octal, hexadecimal, and BCD equivalent.

2. Given any binary number, you will be able to convert it into its decimal, octal, hexadecimal, and BCD equivalent.

3. Given any octal number, you will be able to convert it into its decimal and binary equivalent.

4. Given any hexadecimal number, you will be able to convert it into its decimal and binary equivalent.

5. Given any BCD code, you will be able to convert it into its decimal and binary equivalent.

6. Given a list of popular digital codes, you will be able to read and identify them including pure binary, natural 8421 BCD, Gray, ASCII, and BAUDOT.

7. You will be able to convert a letter or number into its ASCII binary code, and convert an ASCII binary code into its letter or number equivalent.

8. You will be able to define the following terms:

Radix	BCD
Integer	Gray Code
Decimal	ASCII
Binary	BAUDOT
Octal	Most Significant Bit (MSB)
Hexadecimal	Least Significant Bit (LSB)
Bit	Most Significant Digit (MSD)
Parity	Least Significant Digit (LSD)

UNIT ACTIVITY GUIDE

	Completion Time
☐ Read section on Decimal Number System.	_____
☐ Answer Self-Test Review questions 1 — 6.	_____
☐ Read section on Binary Number System.	_____
☐ Answer Self-Test Review questions 7 — 13.	_____
☐ Read section on Octal Number System.	_____
☐ Answer Self-Test Review questions 14 — 19.	_____
☐ Read section on Hexadecimal Number System.	_____
☐ Answer Self-Test Review questions 20 — 25.	_____
☐ Read section on Binary Codes.	_____
☐ Answer Self-Test Review questions 26 — 36.	_____
☐ Perform Experiments 1 and 2.	_____
☐ Complete the Unit Examination.	_____
☐ Check Examination Answers.	_____

DECIMAL NUMBER SYSTEM

The number system we are all familiar with is the decimal number system. This system was originally devised by Hindu mathematicians in India about 400 A.D. The Arabs began to use the system about 800 A.D., where it became known as the Arabic Number System. After it was introduced to the European community about 1200 A.D., the system soon acquired the title "decimal number system."

A basic distinguishing feature of a number system is its **base** or **radix**. The base indicates the number of characters or digits used to represent quantities in that number system. The decimal number system has a base or radix of 10 because we use the ten digits 0 through 9 to represent quantities. When a number system is used where the base is not known, a subscript is used to show the base. For example, the number 4603_{10} is derived from a number system with a base of 10.

Positional Notation The decimal number system is positional or weighted. This means each digit position in a number carries a particular weight which determines the magnitude of that number. Each position has a weight determined by some power of the number system base, in this case 10. The positional weights are 10^0 (units)*, 10^1 (tens), 10^2 (hundreds), etc. Refer to Figure 1-1 for a condensed listing of powers of 10.

$10^0 = 1$
$10^1 = 10$
$10^2 = 100$
$10^3 = 1{,}000$
$10^4 = 10{,}000$
$10^5 = 100{,}000$
$10^6 = 1{,}000{,}000$
$10^7 = 10{,}000{,}000$
$10^8 = 100{,}000{,}000$
$10^9 = 1{,}000{,}000{,}000$

Figure 1-1
Condensed listing of powers of 10.

*Any number with an exponent of zero is equal to one.

We evaluate the total quantity of a number by considering the specific digits and the weights of their positions. For example, the decimal number 4603 is written in the shorthand notation with which we are all familiar. This number can also be expressed with positional notation.

$$(4 \times 10^3) + (6 \times 10^2) + (0 \times 10^1) + (3 \times 10^0) =$$
$$(4 \times 1000) + (6 \times 100) + (0 \times 10) + (3 \times 1) =$$
$$4000 + 600 + 0 + 3 = 4603_{10}$$

To determine the value of a number, multiply each digit by the weight of its position and add the results.

Fractional Numbers So far, only **integer** or whole numbers have been discussed. An integer is any of the natural numbers, the negatives of these numbers, or zero (that is, 0, 1, 4, 7, etc.). Thus, an integer represents a whole or complete number. But, it is often necessary to express quantities in terms of fractional parts of a whole number.

Decimal fractions are numbers whose positions have weights that are **negative powers of ten** such as $10^{-1} = \frac{1}{10} = 0.1$, $10^{-2} = \frac{1}{100} = 0.01$, etc.

Figure 1-2 provides a condensed listing of negative powers of 10 (decimal fractions).

$$10^{-1} = \frac{1}{10} = 0.1$$

$$10^{-2} = \frac{1}{100} = 0.01$$

$$10^{-3} = \frac{1}{1000} = 0.001$$

$$10^{-4} = \frac{1}{10,000} = 0.0001$$

$$10^{-5} = \frac{1}{100,000} = 0.00001$$

$$10^{-6} = \frac{1}{1,000,000} = 0.000001$$

Figure 1-2
Condensed listing of negative powers of 10.

A radix point (decimal point for base 10 numbers) **separates** the **integer** and **fractional** parts of a number. The integer or whole portion is to the left of the decimal point and has positional weights of units, tens, hundreds, etc. The fractional part of the number is to the right of the decimal point and has positional weights of tenths, hundredths, thousandths, etc. To illustrate this, the decimal number 278.94 can be written with positional notation as shown below.

$$(2 \times 10^2) + (7 \times 10^1) + (8 \times 10^0) + (9 \times 10^{-1}) + (4 \times 10^{-2}) =$$
$$(2 \times 100) + (7 \times 10) + (8 \times 1) + (9 \times 1/10) + (4 \times 1/100) =$$
$$200 + 70 + 8 + 0.9 + 0.04 = 278.94_{10}$$

In this example, the left-most digit (2×10^2) is the **most significant digit** or MSD because it carries the greatest weight in determining the value of the number. The right-most digit, called the **least significant digit** or LSD, has the lowest weight in determining the value of the number. Therefore, as the term implies, the MSD is the digit that will affect the greatest change when its value is altered. The LSD has the smallest effect on the complete number value.

Self-Test Review

1. The _____ indicates the number of characters or digits in a number system.

2. In the decimal number system, the base or radix is _____.

3. Write the following numbers using positional notation.

   A. 4563_{10}
   B. 26.32_{10}
   C. 536.9_{10}

4. In the decimal number system, the radix point is called the _____ _____.

5. Convert the following positional notations into their shorthand decimal form.

   A. $(5 \times 10^{1}) + (2 \times 10^{0}) + (3 \times 10^{-1}) + (8 \times 10^{-2})$
   B. $(4 \times 10^{-1}) + (6 \times 10^{-2}) + (2 \times 10^{-3})$
   C. $(3 \times 10^{3}) + (7 \times 10^{2}) + (1 \times 10^{1}) + (0 \times 10^{0})$

6. The radix point separates the _____ and _____ parts of a number.

Answers

1. base or radix.

2. 10.

3. A. $4563_{10} = 4000 + 500 + 60 + 3.$
   $= (4 \times 10^3) + (5 \times 10^2) + (6 \times 10^1) + (3 \times 10^0)$
   B. $(2 \times 10^1) + (6 \times 10^0) + (3 \times 10^{-1}) + (2 \times 10^{-2})$
   C. $(5 \times 10^2) + (3 \times 10^1) + (6 \times 10^0) + (9 \times 10^{-1})$

4. decimal point.

5. A. $(5 \times 10^1) + (2 \times 10^0) + (3 \times 10^{-1}) + (8 \times 10^{-2}) =$
   $(5 \times 10) + (2 \times 1) + (3 \times 1/10) + (8 \times 1/100) =$
   $50 + 2 + 0.3 + 0.08 = 52.38_{10}$
   B. 0.462_{10}
   C. 3710_{10}

6. integer or whole, fractional.

BINARY NUMBER SYSTEM

The simplest number system that uses positional notation is the binary number system. As the name implies, a **binary** system contains only two elements or states. In a number system this is expressed as a base of 2, using the digits 0 and 1. These two digits have the same basic value as 0 and 1 in the decimal number system.

Because of its simplicity, microprocessors use the binary number system to manipulate data. Binary data is represented by binary digits called **bits**. The term bit is derived from the contraction of **b**inary dig**it**. Microprocessors operate on groups of bits which are referred to as words. The binary number 11101101 contains eight bits.

Positional Notation

As with the decimal number system, each bit (digit) position of a binary number carries a particular weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example 2). To evaluate the total quantity of a number, consider the specific bits and the weights of their positions. (Refer to Figure 1-3 for a condensed listing of powers of 2.) For example, the binary number 110101 can be written with positional notation as follows:

$$(1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

To determine the decimal value of the binary number 110101, multiply each bit by its positional weight and add the results.

$$(1 \times 32) + (1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) =$$
$$32 + 16 + 0 + 4 + 0 + 1 = 53_{10}$$

$2^0 = 1_{10}$	$2^6 = 64_{10}$
$2^1 = 2_{10}$	$2^7 = 128_{10}$
$2^2 = 4_{10}$	$2^8 = 256_{10}$
$2^3 = 8_{10}$	$2^9 = 512_{10}$
$2^4 = 16_{10}$	$2^{10} = 1024_{10}$
$2^5 = 32_{10}$	$2^{11} = 2048_{10}$

Figure 1-3
Condensed listing of powers of 2.

Fractional binary numbers are expressed as negative powers of 2. Figure 1-4 provides a condensed listing of negative powers of 2. In positional notation, the binary number 0.1101 can be expressed as follows:

$$(1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$$

To determine the decimal value of the binary number 0.1101, multiply each bit by its positional weight and add the results.

$$(1 \times 1/2) + (1 \times 1/4) + (0 \times 1/8) + (1 \times 1/16) =$$
$$0.5 + 0.25 + 0 + 0.0625 = 0.8125_{10}$$

In the binary number system, the radix point is called the binary point.

$$2^{-1} = \frac{1}{2} = 0.5_{10}$$

$$2^{-2} = \frac{1}{4} = 0.25_{10}$$

$$2^{-3} = \frac{1}{8} = 0.125_{10}$$

$$2^{-4} = \frac{1}{16} = 0.0625_{10}$$

$$2^{-5} = \frac{1}{32} = 0.03125_{10}$$

$$2^{-6} = \frac{1}{64} = 0.015625_{10}$$

$$2^{-7} = \frac{1}{128} = 0.0078125_{10}$$

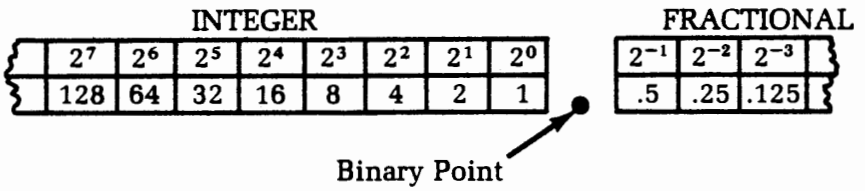
$$2^{-8} = \frac{1}{256} = 0.00390625_{10}$$

Figure 1-4
Condensed listing of negative powers of 2.

Converting Between the Binary and Decimal Number Systems

In working with microprocessors, you will often need to determine the decimal value of binary numbers. In addition, you will find it necessary to convert a specific decimal number into its binary equivalent. The following information shows how such conversions are accomplished.

Binary to Decimal To convert a binary number into its decimal equivalent, add together the weights of the positions in the number where binary 1's occur. The weights of the integer and fractional positions are indicated below.



As an example, convert the binary number 1010 into its decimal equivalent. Since no binary point is shown, the number is assumed to be an integer number, where the binary point is to the right of the number. The right-most bit, called the **least significant bit** or LSB, has the lowest integer weight of $2^0 = 1$. The left-most bit is the **most significant bit** (MSB) because it carries the greatest weight in determining the value of the number. In this example, it has a weight of $2^3 = 8$. To evaluate the number, add together the weights of the positions where binary 1's appear. In this example, 1's occur in the 2^3 and 2^1 positions. The decimal equivalent is ten.

Binary Number	1	0	1	0
Position Weights	2^3	2^2	2^1	2^0
Decimal Equivalent				

$$8 + 0 + 2 + 0 = 10_{10}$$

To further illustrate this process, convert the binary number 101101.11 into its decimal equivalent.

Binary Number	1	0	1	1	0	1	.1	1
Position Weights	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
Decimal Equivalent								

$$32 + 0 + 8 + 4 + 0 + 1 + .5 + .25 = 45.75_{10}$$

Decimal to Binary A decimal integer number can be converted to a different base or radix through successive divisions by the desired base. To convert a decimal integer number to its binary equivalent, successively divide the number by 2 and note the remainders. When you divide by 2, the remainder will always be 1 or 0.

The remainders form the equivalent binary number.

As an example, the decimal number 25 is converted into its binary equivalent.

25 ÷ 2 = 12	with remainder 1	← LSB
12 ÷ 2 = 6	0	
6 ÷ 2 = 3	0	
3 ÷ 2 = 1	1	
1 ÷ 2 = 0	1	← MSB

Divide the decimal number by 2 and note the remainder. Then divide the quotient by 2 and again note the remainder. Then divide the quotient by 2 and again note the remainder. Continue this division process until 0 results. Then collect remainders beginning with the last or most significant bit (MSB) and proceed to the first or least significant bit (LSB). The number $11001_2 = 25_{10}$. Notice that the remainders are collected in the reverse order. That is, the first remainder becomes the least significant bit, while the last remainder becomes the most significant bit.

NOTE: Do not attempt to use a calculator to perform this conversion. It would only supply you with confusing results.

To further illustrate this, the decimal number 175 is converted into its binary equivalent.

175 ÷ 2 = 87	with remainder 1	← LSB
87 ÷ 2 = 43	1	
43 ÷ 2 = 21	1	
21 ÷ 2 = 10	1	
10 ÷ 2 = 5	0	
5 ÷ 2 = 2	1	
2 ÷ 2 = 1	0	
1 ÷ 2 = 0	1	← MSB

The division process continues until 0 results. The remainders are collected to produce the number $10101111_2 = 175_{10}$.

To convert a decimal fraction to a different base or radix, multiply the fraction successively by the desired base and record any integers produced by the multiplication as an overflow. For example, to convert the decimal fraction 0.3125 into its binary equivalent, multiply repeatedly by 2.

$0.3125 \times 2 = 0.625 = 0.625$	with overflow	0	← MSB
$0.6250 \times 2 = 1.250 = 0.250$		1	
$0.2500 \times 2 = 0.500 = 0.500$		0	
$0.5000 \times 2 = 1.000 = 0$		1	← LSB

These multiplications will result in numbers with a 1 or 0 in the units position (the position to the left of the decimal point). By recording the value of the units position, you can construct the equivalent binary fraction. This units position value is called the "overflow." Therefore, when 0.3125 is multiplied by 2, the overflow is 0. This becomes the most significant bit (MSB) of the binary equivalent fraction. Then 0.625 is multiplied by 2. Since the product is 1.25, the overflow is 1. When there is an overflow of 1, it is effectively subtracted from the product when the value is recorded. Therefore, only 0.25 is multiplied by 2 in the next multiplication process. This method continues until an overflow with no fraction results. It is important to note that you can not always obtain 0 when you multiply by 2. Therefore, you should only continue the conversion process to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix (binary) point with the MSB and proceed to the LSB. This is the same order in which the overflows were produced. The number $0.0101_2 = 0.3125_{10}$.

To further illustrate this process, the decimal fraction 0.90625 is converted into its binary equivalent.

$0.90625 \times 2 = 1.8125 = 0.8125$	with overflow	1	← MSB
$0.81250 \times 2 = 1.6250 = 0.6250$		1	
$0.62500 \times 2 = 1.2500 = 0.2500$		1	
$0.25000 \times 2 = 0.5000 = 0.5000$		0	
$0.50000 \times 2 = 1.0000 = 0$		1	← LSB

The multiplication process continues until either 0 or the desired precision is obtained. The overflows are then collected beginning with the MSB at the binary (radix) point and proceeding to the LSB. The number $0.11101_2 = 0.90625_{10}$.

If the decimal number contains both an integer and fraction, you must separate the integer and fraction using the decimal point as the break point. Then perform the appropriate conversion process on each number portion. After you convert the binary integer and binary fraction, recombine them. For example, the decimal number 14.375 is converted into its binary equivalent.

$14.375_{10} = 14_{10} + 0.375_{10}$

$14 \div 2 = 7$	with remainder	0	← LSB
$7 \div 2 = 3$		1	
$3 \div 2 = 1$		1	
$1 \div 2 = 0$		1	← MSB

$\boxed{14_{10} = 1110_2}$

$0.375 \times 2 = 0.75 = 0.75$	with overflow	0	← MSB
$0.750 \times 2 = 1.50 = 0.50$		1	
$0.500 \times 2 = 1.00 = 0$		1	← LSB

$\boxed{0.375_{10} = 0.011_2}$

$14.375_{10} = 14_{10} + 0.375_{10} = 1110_2 + 0.011_2 = 1110.011_2.$

Self-Test Review

7. The base or radix of the binary number system is _____.

8. A binary digit is called a _____.

9. Convert the following binary integers to decimal.

   A. 101101
   B. 1001
   C. 1101100

10. Convert the following binary fractions to decimal.

   A. 0.011
   B. 0.01101
   C. 0.1001

11. Convert the following decimal integers to binary.

   A. 63
   B. 12
   C. 132

12. Convert the following decimal fractions to binary.

   A. 0.4375
   B. 0.96875
   C. 0.625

13. Convert 13.125_{10} to binary.

Answers

7. 2

8. bit

9. A. $101101_2 =$
$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) =$
$32 + 0 + 8 + 4 + 0 + 1 = 45_{10}$
B. $1001_2 = 9_{10}$
C. $1101100_2 = 108_{10}$

10. A. $0.011_2 = (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) =$
$0 + \frac{1}{4} + \frac{1}{8} = 0 + 0.25 + 0.125 = 0.375_{10}$
B. $0.01101_2 = 0.40625_{10}$
C. $0.1001_2 = 0.5625_{10}$

11. A.

$63 \div 2 = 31$	with remainder	1 ←	LSB
$31 \div 2 = 15$		1	
$15 \div 2 = 7$		1	
$7 \div 2 = 3$		1	
$3 \div 2 = 1$		1	
$1 \div 2 = 0$		1 ←	MSB

$\boxed{63_{10} = 111111_2}$

B. $12_{10} = 1100_2$
C. $132_{10} = 10000100_2$

12. A. $0.4375 \times 2 = 0.875 = 0.875$ with overflow 0 ← MSB
$0.8750 \times 2 = 1.750 = 0.750$ 1
$0.7500 \times 2 = 1.500 = 0.500$ 1
$0.5000 \times 2 = 1.000 = 0$ 1 ← LSB

$\boxed{0.4375_{10} = 0.0111_2}$

B. $0.96875_{10} = 0.11111_2$

C. $0.625_{10} = 0.101_2$

13. $13.125_{10} = 13_{10} + 0.125_{10}$

$13 \div 2 = 6$ with remainder 1 ← LSB
$6 \div 2 = 3$ 0
$3 \div 2 = 1$ 1
$1 \div 2 = 0$ 1 ← MSB

$\boxed{13_{10} = 1101_2}$

$0.125 \times 2 = 0.25 = 0.25$ with overflow 0 ← MSB
$0.250 \times 2 = 0.50 = 0.50$ 0
$0.500 \times 2 = 1.00 = 0$ 1 ← LSB

$\boxed{0.125_{10} = 0.001_2}$

$13.125_{10} = 13_{10} + 0.125_{10} = 1101_2 + 0.001_2 = 1101.001_2$

OCTAL NUMBER SYSTEM

Octal is another number system that is often used with microprocessors. It has a base (radix) of 8, and uses the digits 0 through 7. These eight digits have the same basic value as the digits 0—7 in the decimal number system.

As with the binary number system, each digit position of an octal number carries a positional weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example, 8). To evaluate the total quantity of a number, consider the specific digits and the weights of their positions. Refer to Figure 1-5 for a condensed listing of powers of 8. For example, the octal number 372.01 can be written with positional notation as follows:

$$(3 \times 8^2) + (7 \times 8^1) + (2 \times 8^0) + (0 \times 8^{-1}) + (1 \times 8^{-2})$$

The decimal value of the octal number 372.01 is determined by multiplying each digit by its positional weight and adding the results. As with decimal and binary numbers, the radix (octal) point separates the integer from the fractional part of the number.

$$(3 \times 64) + (7 \times 8) + (2 \times 1) + (0 \times 0.125) + (1 \times 0.015625) =$$
$$192 + 56 + 2 + 0 + 0.015625 = 250.015625_{10}$$

$$8^{-4} = \frac{1}{4096} = 0.000244140625_{10}$$

$$8^{-3} = \frac{1}{512} = 0.001953125_{10}$$

$$8^{-2} = \frac{1}{64} = 0.015625_{10}$$

$$8^{-1} = \frac{1}{8} = 0.125_{10}$$

$$1_{10} = 8^0$$
$$8_{10} = 8^1$$
$$64_{10} = 8^2$$
$$512_{10} = 8^3$$
$$4096_{10} = 8^4$$
$$32768_{10} = 8^5$$
$$262144_{10} = 8^6$$

Figure 1-5
Condensed listing of powers of 8.

Conversion From Decimal to Octal

Decimal to octal conversion is accomplished in the same manner as decimal to binary, with one exception; the base number is now 8 rather than 2. As an example, the decimal number 194 is converted into its octal equivalent.

194 ÷ 8 = 24 with remainder	2	← LSD
24 ÷ 8 = 3	0	
3 ÷ 8 = 0	3	← MSD

Divide the decimal number by 8 and note the remainder. (The remainder can be any number from 0 to 7.)

Then divide the quotient by 8 and again note the remainder. Continue dividing until 0 results. Finally, collect the remainders beginning with the last or most significant digit (MSD) and proceed to the first or least significant digit (LSD). The number $302_8 = 194_{10}$. Figure 1-6 illustrates the relationship between the first several decimal, octal, and binary integers.

DECIMAL	OCTAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111
16	20	10000
17	21	10001
18	22	10010
19	23	10011
20	24	10100

Figure 1-6
Sample comparison of decimal, octal, and binary integers.

To further illustrate this process, the decimal number 175 is converted into its octal equivalent.

175 ÷ 8 = 21 with remainder	7	←	LSD
21 ÷ 8 = 2	5		
2 ÷ 8 = 0	2	←	MSD

The division process continues until a quotient of 0 results. The remainders are collected, producing the number $257_8 = 175_{10}$.

To convert a decimal fraction to an octal fraction, multiply the fraction successively by 8 (octal base). As an example, the decimal fraction 0.46875 is converted into its octal equivalent.

0.46875 × 8 = 3.75 = 0.75 with overflow	3	←	MSD
0.75000 × 8 = 6.00 = 0	6	←	LSD

Multiply the decimal number by 8. If the product exceeds one, subtract the integer (overflow) from the product. Then multiply the product fraction by 8 and again note any "overflow." Continue multiplying until an overflow, with 0 for a fraction, results. Remember, you can not always obtain 0 when you multiply by 8. Therefore, you should only continue this conversion process to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix (octal point) with the MSD and proceed to the LSD. The number $0.36_8 = 0.46875_{10}$. Figure 1-7 illustrates the relationship between decimal, octal, and binary fractions.

Now, the decimal fraction 0.136 will be converted into its octal equivalent with four-place precision.

0.136 × 8 = 1.088 = 0.088 with overflow	1	←	MSD
0.088 × 8 = 0.704 = 0.704	0		
0.704 × 8 = 5.632 = 0.632	5		
0.632 × 8 = 5.056 = 0.056	5	←	LSD

$0.136_{10} \cong 0.1055_8$

The number 0.1055_8 approximately equals 0.136_{10}. If you convert 0.1055_8 back to decimal (using positional notation), you will find $0.1055_8 = 0.135986328125_{10}$. This example shows that extending the precision of your conversion is of little value unless extreme accuracy is required.

DECIMAL	OCTAL	BINARY
0.015625	0.01	0.000001
0.03125	0.02	0.00001
0.046875	0.03	0.000011
0.0625	0.04	0.0001
0.078125	0.05	0.000101
0.09375	0.06	0.00011
0.109375	0.07	0.000111
0.125	0.1	0.001
0.140625	0.11	0.001001
0.15625	0.12	0.00101
0.171875	0.13	0.001011
0.1875	0.14	0.0011
0.203125	0.15	0.001101
0.21875	0.16	0.00111
0.234375	0.17	0.001111
0.25	0.2	0.01
0.265625	0.21	0.010001
0.28125	0.22	0.01001
0.296875	0.23	0.010011
0.3125	0.24	0.0101

Figure 1-7
Sample comparison of decimal, octal, and binary fractions.

As with decimal to binary conversion of a number that contains both an integer and fraction, decimal to octal conversion requires two operations. You must separate the integer from the fraction, then perform the appropriate conversion on each number. After you convert them, you must recombine the octal integer and octal fraction. For example, convert the decimal number 124.78125 into its octal equivalent.

$124.78125_{10} = 124_{10} + 0.78125_{10}$

$124 \div 8 = 15$ with remainder 4 ← LSD
$15 \div 8 = 1$ 7
$1 \div 8 = 0$ 1 ← MSD

$\boxed{124_{10} = 174_8}$

$0.78125 \times 8 = 6.25 = 0.25$ with overflow 6 ← MSD
$0.25000 \times 8 = 2.00 = 0$ 2 ← LSD

$\boxed{0.78125_{10} = 0.62_8}$

$124.78125_{10} = 124_{10} + 0.78125_{10} = 174_8 + 0.62_8 = 174.62_8$

Converting Between the Octal and Binary Number Systems

Microprocessors manipulate data using the binary number system. However, when larger quantities are involved, the binary number system can become cumbersome. Therefore, other number systems are frequently used as a form of binary shorthand to speed-up and simplify data entry and display. The octal number system is one of the systems that is used in this manner. It is similar to the decimal number system, which makes it easier to understand numerical values. In addition, conversion between binary and octal is readily accomplished because of the value structure of octal. Figures 1-6 and 1-7 illustrate the relationship between octal and binary integers and fractions.

As you know, three bits of a binary number exactly equal eight value combinations. Therefore, you can represent a 3-bit binary number with a 1-digit octal number.

$$101_2 = (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 4 + 0 + 1 = 5_8$$

Because of this relationship, converting binary to octal is simple and straight forward. For example, binary number 101001 is converted into its octal equivalent.

101001_2
REWRITE AS
MSB LSB
101 001
YIELDS
51_8

To convert a binary number to octal, first separate the number into groups containing three bits, beginning with the least significant bit. Then convert each 3-bit group into its octal equivalent. This gives you an octal number equal in value to the binary number.

Binary fractions can also be converted to their octal equivalents using the same process, with one exception. The binary bits must be separated into groups of three beginning with the most significant bit. For example, the binary fraction 0.011101 is converted into its octal equivalent.

0.011101_2
REWRITE AS
MSB LSB
0.011 101
YIELDS
0.35_8

Again, you must first separate the binary number into groups of three beginning at the radix (binary) point. Then convert each 3-bit group into its octal equivalent.

To separate binary numbers into 3-bit groups when the number does not contain the necessary bits, add zeros to the number until the number can be separated into 3-bit groups. For example, binary number 10010101.1011 is converted into its octal equivalent.

10010101.1011_2
REWRITE AS
MSB LSB
010 010 101.101 100
YIELDS
225.54_8

As before, the integer part of the number is separated into 3-bit groups, beginning at the radix (binary) point. Note that the third group contains only two bits. However, a zero can be added to the group without changing the value of the binary number. Next, the fractional part of the number is separated into 3-bit groups, beginning at the radix (binary) point. Note that the second group contains only one bit. By adding two zeros to the group, the group is complete with no change in the value of the binary number.

NOTE: Whenever you add zeros to a **binary integer**, always place them to the **left** of the most significant bit. When you add zeros to a **binary fraction**, always place them to the **right** of the least significant bit.

After you have formed the 3-bit groups, convert each group into its octal equivalent. This gives you an octal number equal in value to the binary number. Now convert binary number 1101110.01 into its octal equivalent.

1101110.01_2
REWRITE AS
MSB LSB
001 101 110.010
YIELDS
156.2_8

Separate the integer and fraction into 3-bit groups, adding zeros as necessary. Then convert each 3-bit group to octal. **Never** shift the radix (binary) point in order to form 3-bit groups.

Converting octal to binary is just the opposite of the previous process. You simply convert each octal number into its 3-bit binary equivalent. For example, convert the octal number 75.3 into its binary equivalent.

75.3_8
YIELDS
MSB LSB
111 101.011

REWRITE AS
111101.011_2

The above example is a simple conversion. Now a more complex octal number (1752.714) will be converted to a binary number.

1752.714_8
YIELDS
MSB LSB
001 111 101 010.111 001 100
REWRITE AS
1111101010.1110011_2

Again, each octal digit is converted into its 3-bit binary equivalent. However, in this example, there are two insignificant zeros in front of the MSB and after the LSB. Since these zeros have no value, they should be removed from the final result.

Self-Test Review

14. The base or radix of the octal number system is _____.

15. Convert the following decimal integers to octal.

    A. 156
    B. 32
    C. 1785

16. Convert the following decimal fractions to octal. Do not use greater than 4-place precision.

    A. 0.1432
    B. 0.8125
    C. 0.6832

17. Convert 735.984375_{10} to octal.

18. Convert the following binary numbers to octal.

    A. 10000111.01101
    B. 11101.0101
    C. 1001101.000001

19. Convert the following octal numbers to binary.

    A. 372.61
    B. 11.001
    C. 3251.034

Answers

14. 8

15. A. $156 \div 8 = 19$ with remainder 4 ← LSD
$19 \div 8 = 2$ 3
$2 \div 8 = 0$ 2 ← MSD

$\boxed{156_{10} = 234_8}$

B. $32_{10} = 40_8$
C. $1785_{10} = 3371_8$

16. A. $0.1432 \times 8 = 1.1456 = 0.1456$ overflow 1 ← MSD
$0.1456 \times 8 = 1.1648 = 0.1648$ 1
$0.1648 \times 8 = 1.3184 = 0.3184$ 1
$0.3184 \times 8 = 2.5472 = 0.5472$ 2 ← LSD

$\boxed{0.1432_{10} = 0.1112_8}$

B. $0.8125_{10} = 0.64_8$
C. $0.6832_{10} = 0.5356_8$

17. $735.984375_{10} = 735_{10} + 0.984375_{10}$
$= 735 \div 8 = 91$ with remainder 7 ← LSD
$91 \div 8 = 11$ 3
$11 \div 8 = 1$ 3
$1 \div 8 = 0$ 1 ← MSD

$\boxed{735_{10} = 1337_8}$

$0.984375 \times 8 = 7.875 = 0.875$ overflow 7 ← MSD
$0.875000 \times 8 = 7.00 = 0$ 7 ← LSD

$\boxed{0.984375_{10} = 0.77_8}$

$735.984375_{10} = 735_{10} + 0.984375_{10} = 1337_8 + 0.77_8 = 1337.77_8$

18. A. $10000111.01101_2 = 010\ 000\ 111.011\ 010_2$
$= 207.32_8$
B. $11101.0101_2 = 35.24_8$
C. $1001101.000001_2 = 115.01_8$

19. A. $372.61_8 = 011\ 111\ 010.110\ 001_2 = 11111010.110001_2$.
B. $11.001_8 = 1001.000000001_2$
C. $3251.034_8 = 11010101001.0000111_2$

HEXADECIMAL NUMBER SYSTEM

Hexadecimal is another number system that is often used with microprocessors. It is similar in value structure to the octal number system, and thus allows easy conversion with the binary number system. Because of this feature and the fact that hexadecimal simplifies data entry and display to a greater degree than octal, you will use hexadecimal more often than any other number system in this course. As the name implies, hexadecimal has a base (radix) of 16_{10}. It uses the digits 0 through 9 and the letters A through F.

The letters are used because it is necessary to represent 16_{10} different values with a single digit for each value. Therefore, the letters A through F are used to represent the number values 10_{10} through 15_{10}. The following discussion will compare the decimal number system with the hexadecimal number system.

All of the numbers are of equal value between systems ($0_{10} = 0_{16}$, $3_{10} = 3_{16}$, $9_{10} = 9_{16}$, etc.). For numbers greater than 9, this relationship exists: $10_{10} = A_{16}$, $11_{10} = B_{16}$, $12_{10} = C_{16}$, $13_{10} = D_{16}$, $14_{10} = E_{16}$, and $15_{10} = F_{16}$. Using letters in counting may appear awkward until you become familiar with the system. Figure 1-8 illustrates the relationship between decimal and hexadecimal integers, while Figure 1-9 illustrates the relationship between decimal and hexadecimal fractions.

DECIMAL	HEXADECIMAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010
19	13	10011
20	14	10100
21	15	10101
22	16	10110
23	17	10111
24	18	11000
25	19	11001
26	1A	11010
27	1B	11011
28	1C	11100
29	1D	11101
30	1E	11110
31	1F	11111
32	20	100000
33	21	100001
34	22	100010
35	23	100011

Figure 1-8
Sample comparison of decimal, hexadecimal, and binary integers.

DECIMAL	HEXADECIMAL	BINARY
0.00390625	0.01	0.00000001
0.0078125	0.02	0.0000001
0.01171875	0.03	0.00000011
0.015625	0.04	0.000001
0.01953125	0.05	0.00000101
0.0234375	0.06	0.0000011
0.02734375	0.07	0.00000111
0.03125	0.08	0.00001
0.03515625	0.09	0.00001001
0.0390625	0.0A	0.0000101
0.04296875	0.0B	0.00001011
0.046875	0.0C	0.00011
0.05078125	0.0D	0.00001101
0.0546875	0.0E	0.0000111
0.05859375	0.0F	0.00001111
0.0625	0.1	0.0001
0.06640625	0.11	0.00010001
0.0703125	0.12	0.0001001
0.07421875	0.13	0.00010011
0.078125	0.14	0.000101
0.08203125	0.15	0.00010101
0.0859375	0.16	0.0001011
0.08984375	0.17	0.00010111
0.09375	0.18	0.00011
0.09765625	0.19	0.00011001
0.1015625	0.1A	0.0001101
0.10546875	0.1B	0.00011011
0.109375	0.1C	0.000111
0.11328125	0.1D	0.00011101
0.1171875	0.1E	0.0001111
0.12109375	0.1F	0.00011111
0.125	0.2	0.001

Figure 1-9
Sample comparison of decimal, hexadecimal, and binary fractions.

As with the previous number systems, each digit position of a hexadecimal number carries a positional weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example, 16_{10}). The total quantity of a number can be evaluated by considering the specific digits and the weights of their positions. (Refer to Figure 1-10 for a condensed listing of powers of 16_{10}.) For example, the hexadecimal number E5D7.A3 can be written with positional notation as follows:

$$(E \times 16^3) + (5 \times 16^2) + (D \times 16^1) + (7 \times 16^0) + (A \times 16^{-1}) + (3 \times 16^{-2})$$

The decimal value of the hexadecimal number E5D7.A3 is determined by multiplying each digit by its positional weight and adding the results. As with the previous number systems, the radix (hexadecimal) point separates the integer from the fractional part of the number.

$$(14 \times 4096) + (5 \times 256) + (13 \times 16) + (7 \times 1) + (10 \times 1/16) + (3 \times 1/256) =$$
$$57344 + 1280 + 208 + 7 + 0.625 + 0.01171875 =$$
$$58839.63671875_{10}$$

$$16^{-4} = \frac{1}{65536} = 0.0000152587890625_{10}$$

$$16^{-3} = \frac{1}{4096} = 0.000244140625_{10}$$

$$16^{-2} = \frac{1}{256} = 0.00390625_{10}$$

$$16^{-1} = \frac{1}{16} = 0.0625_{10}$$

$$1_{10} = 16^0$$
$$16_{10} = 16^1$$
$$256_{10} = 16^2$$
$$4096_{10} = 16^3$$
$$65536_{10} = 16^4$$
$$1048576_{10} = 16^5$$
$$16777216_{10} = 16^6$$

Figure 1-10
Condensed listing of powers of 16.

Conversion From Decimal to Hexadecimal

Decimal to hexadecimal conversion is accomplished in the same manner as decimal to binary or octal, but with a base number of 16_{10}. As an example, the decimal number 156 is converted into its hexadecimal equivalent.

$156 \div 16 = 9$	with remainder	$12 = C$	← LSD
$9 \div 16 = 0$		$9 = 9$	← MSD

Divide the decimal number by 16_{10} and note the remainder. If the remainder exceeds 9, convert the 2-digit number to its hexadecimal equivalent (12_{10} = C in this example). Then divide the quotient by 16 and again note the remainder. Continue dividing until a quotient of 0 results. Then collect the remainders beginning with the last or most significant digit (MSD) and proceed to the first or least significant digit (LSD). The number $9C_{16} = 156_{10}$. NOTE: The letter H after a number is sometimes used to indicate hexadecimal. However, this course will always use the subscript 16.

To further illustrate this, the decimal number 47632 is converted into its hexadecimal equivalent.

$47632 \div 16 = 2977$	with remainder	$0 = 0$	← LSD
$2977 \div 16 = 186$		$1 = 1$	
$186 \div 16 = 11$		$10 = A$	
$11 \div 16 = 0$		$11 = B$	← MSD

The division process continues until a quotient of 0 results. The remainders are collected, producing the number $BA10_{16} = 47632_{10}$. Remember, any remainder that exceeds the digit 9 must be converted to its letter equivalent. (In this example, 10 = A, and 11 = B.)

To convert a decimal fraction to a hexadecimal fraction, multiply the fraction successively by 16_{10} (hexadecimal base). As an example the decimal fraction 0.78125 is converted into its hexadecimal equivalent.

$0.78125 \times 16 = 12.5 = 0.5$	with overflow	$12 = C$	← MSD
$0.50000 \times 16 = 8.0 = 0$		$8 = 8$	← LSD

Multiply the decimal by 16_{10}. If the product exceeds one, subtract the integer (overflow) from the product. If the "overflow" exceeds 9, convert the 2-digit number to its hexadecimal equivalent. Then multiply the product fraction by 16_{10} and again note any overflow. Continue multiplying until an overflow, with 0 for a fraction, results. Remember, you can not always obtain 0 when you multiply by 16. Therefore, you should only continue the conversion to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix point with the MSD and proceed to the LSD.The number $0.C8_{16} = 0.78125_{10}$.

Now the decimal fraction 0.136 will be converted into its hexadecimal equivalent with five-place precision.

$0.136 \times 16 = 2.176 = 0.176$	overflow	$2 = 2$	→ MSD
$0.176 \times 16 = 2.816 = 0.816$		$2 = 2$	
$0.816 \times 16 = 13.056 = 0.056$		$13 = D$	
$0.056 \times 16 = 0.896 = 0.896$		$0 = 0$	
$0.896 \times 16 = 14.336 = 0.336$		$14 = E$	→ LSD

The number $0.22D0E_{16}$ approximately equals 0.136_{10}. If you convert $0.22D0E_{16}$ back to decimal (using positional notation), you will find $0.22D0E_{16} = 0.1359996795654296875_{16}$. This example shows that extending the precision of your conversion is of little value unless extreme accuracy is required.

As shown in this section, conversion of an integer from decimal to hexadecimal requires a different technique than for conversion of a fraction. Therefore, when you convert a hexadecimal number composed of an integer and a fraction, you must separate the integer and fraction, then perform the appropriate operation on each. After you convert them, you must recombine the integer and fraction. For example, the decimal number 124.78125 is converted into its hexadecimal equivalent.

$124.78125_{10} = 124_{10} + 0.78125_{10}$

$124 \div 16 = 7$ with remainder $12 = C$ ← LSD

$7 \div 16 = 0$ $7 = 7$ ← MSD

$\boxed{124_{10} = 7C_{16}}$

$0.78125 \times 16 = 12.5 = 0.5$ overflow $12 = C$ ← MSD

$0.50000 \times 16 = 8.0 = 0$ $8 = 8$ ← LSD

$\boxed{0.78125_{10} = 0.C8_{16}}$

$124.78125_{10} = 124_{10} + 0.78125_{10} = 7C_{16} + 0.C8_{16} = 7C.C8_{16}$

First separate the decimal integer and fraction. Then convert the integer and fraction to hexadecimal.

Finally, recombine the integer and fraction.

Converting Between the Hexadecimal and Binary Number Systems

Previously, the octal number system was described as an excellent shorthand form to express large binary quantities. This method is very useful with many microprocessors. The trainer used with this course uses the hexadecimal number system to represent binary quantities. As a result, frequent conversions from binary-to-hexadecimal are necessary. Figures 1-8 and 1-9 illustrate the relationship between hexadecimal and binary integers and fractions.

As you know, four bits of a binary number exactly equal 16_{10} value combinations. Therefore, you can represent a 4-bit binary number with a 1-digit hexadecimal number:

$$1101_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 8 + 4 + 0 + 1 = 13_{10} = D_{16}$$

Because of this relationship, converting binary to hexadecimal is simple and straightforward. For example, binary number 10110110 is converted into its hexadecimal equivalent.

10110110_2
REWRITE AS
MSB LSB
1011 0110
YIELDS
$B6_{16}$

To convert a binary number to hexadecimal, first separate the number into groups containing four bits, beginning with the least significant bit. Then convert each 4-bit group into its hexadecimal equivalent. Don't forget to use letter digits as required. This gives you a hexadecimal number equal in value to the binary number.

Now convert a larger binary number (10101101101) into its hexadecimal equivalent.

10101101101_2
REWRITE AS
MSB LSB
0101 0110 1101
YIELDS
$56D_{16}$

Again, the binary number is separated into 4-bit groups beginning with the LSB. However, the third group contains only three bits. Since each group must contain four bits, a zero must be added after the MSB. The third group will then have four bits with no change in the value of the binary number. Now each 4-bit group can be converted into its hexadecimal equivalent. **Whenever you add zeros to a binary integer, always place them to the left of the most significant bit.**

Binary fractions can also be converted to their hexadecimal equivalents using the same process, with one exception; the binary bits are separated into groups of four, beginning with the most significant bit (at the radix point). For example, the binary fraction 0.01011011 is converted into its hexadecimal equivalent.

0.01011011_2
REWRITE AS
MSB LSB
$0.0101 \quad 1011$
YIELDS
$0.5B_{16}$

Again, you must separate the binary number into groups of four, beginning with the radix point. Then convert each 4-bit group into its hexadecimal equivalent. This gives you a hexadecimal number equal in value to the binary number.

Now convert a larger binary fraction (0.1101001101) into its hexadecimal equivalent.

0.1101001101_2
REWRITE AS
MSB LSB
$0.1101 \quad 0011 \quad 0100$
YIELDS
$0.D34_{16}$

Separate the binary number into 4-bit groups, beginning at the radix (binary) point (MSB). Note that the third group contains only two bits. Since each group must contain four bits, two zeros must be added after the LSB. The third group will then have four bits with no change in the value of the binary number. Now, each 4-bit group can be converted into its hexadecimal equivalent. **Whenever you add zeros to a binary fraction, always place them to the right of the least significant bit.**

Now, a binary number containing both an integer and a fraction (110110101.01110111) will be converted into its hexadecimal equivalent.

$$110110101.01110111_2$$

REWRITE AS

MSB ↘ LSB ↙

$$\underline{0001}\quad 1011\quad 0101.0111\quad 0111$$

YIELDS

$$1B5.77_{16}$$

The integer part of the number is separated into groups of four, **beginning** at the radix point. Note that three zeros were added to the third group to complete the group. The fractional part of the number is separated into groups of four, **beginning** at the radix point. (No zeros were needed to complete the fractional groups.) The integer and fractional 4-bit groups are then converted to hexadecimal. The number $110110101.01110111_2 = 1B5.77_{16}$. **Never** shift the radix point in order to form 4-bit groups.

Converting hexadecimal to binary is just the opposite of the previous process; simply convert each hexadecimal number into its 4-bit binary equivalent. For example, convert the hexadecimal number 8F.41 into its binary equivalent.

$$8F.41_{16}$$

YIELDS

MSB ↘ LSB ↙

$$1000\quad 1111.0100\quad 0001$$

REWRITE AS

$$10001111.01000001_2$$

Convert each hexadecimal digit into a 4-bit binary number. Then condense the 4-bit groups to form the binary value equal to the hexadecimal value. The number $8F.41_{16} = 10001111.01000001_2$.

Now, the hexadecimal number 175.4E will be converted into its binary equivalent.

$175.4E_{16}$
YIELDS
MSB LSB
0001 0111 0101.0100 1110
REWRITE AS
101110101.0100111_2

Again, each hexadecimal digit is converted into its 4-bit binary equivalent. However, in this example there are three insignificant zeros in front of the MSB and one after the LSB. Since these zeros have no value, they should be removed from the final result.

Self-Test Review

20. The base or radix of the hexadecimal number system is _____.

21. Convert the following decimal integers to hexadecimal.

    A. 783
    B. 5372
    C. 957

22. Convert the following decimal fractions to hexadecimal. Do not use greater than four-place precision.

    A. 0.653
    B. 0.109375
    C. 0.4567

23. Convert 1573.125_{10} to its hexadecimal equivalent.

24. Convert the following binary numbers to hexadecimal.

    A. 100001101.01011
    B. 11111011001.01
    C. 110001101.00010010101

25. Convert the following hexadecimal numbers to binary.

    A. AE7.D2
    B. 2C5.21F8
    C. 1B6.64E

Answers

20. 16_{10}.

21. A. $783 \div 16 = 48$ with remainder $15 = F$ ← LSD
$48 \div 16 = 3$ $0 = 0$
$3 \div 16 = 0$ $3 = 3$ ← MSD

$\boxed{783_{10} = 30F_{16}}$

B. $5372_{10} = 14FC_{16}$
C. $957_{10} = 3BD_{16}$

22. A. $0.653 \times 16 = 10.448 = 0.448$ with overflow $10 = A$ ← MSD
$0.448 \times 16 = 7.168 = 0.168$ $7 = 7$
$0.168 \times 16 = 2.688 = 0.688$ $2 = 2$
$0.688 \times 16 = 11.008 = 0.008$ $11 = B$ ← LSD

$\boxed{0.653_{10} = 0.A72B_{16}}$

B. $0.109375_{10} = 0.1C_{16}$
C. $0.4567_{10} = 0.74EA_{16}$

23. A. $1573.125_{10} = 1573_{10} + 0.125_{10}$
$1573 \div 16 = 98$ with remainder $5 = 5$ ← LSD
$98 \div 16 = 6$ $2 = 2$
$6 \div 16 = 0$ $6 = 6$ ← MSD

$\boxed{1573_{10} = 625_{16}}$

$0.125 \times 16 = 2.00 = 0$ with overflow $2 = 2$ ← MSD ← LSD

$\boxed{0.125_{10} = 0.2_{16}}$

$1573.125_{10} = 1573_{10} + 0.125_{10} = 625_{16} + 0.2_{16} = 625.2_{16}$

24. A. $100001101.01011_2 = 0001\ 0000\ 1101.0101\ 1000_2$
$= 10D.58_{16}$

B. $11111011001.01_2 = 7D9.4_{16}$
C. $110001101.00010010101_2 = 18D.12A_{16}$

25. A. $AE7.D2_{16} = 1010\ 1110\ 0111.1101\ 0010_2$
$= 101011100111.1101001_2$
B. $2C5.21F8_{16} = 1011000101.0010000111111_2$
C. $1B6.64E_{16} = 110110110.01100100111_2$

BINARY CODES

Converting a decimal number into its binary equivalent is called "coding." A decimal number is expressed as a binary code or binary number. The **binary number system**, as discussed, is known as the pure binary code. This name distinguishes it from other types of binary codes. This section will discuss some of the other types of binary codes used in computers.

Binary Coded Decimal

The decimal number system is easy to use because it is so familiar. The binary number system is less convenient to use because it is less familiar. It is difficult to quickly glance at a binary number and recognize its decimal equivalent. For example, the binary number 1010011 represents the decimal number 83. It is difficult to tell immediately by looking at the number what its decimal value is. However, within a few minutes, using the procedures described earlier, you could readily calculate its decimal value. The amount of time it takes to convert or recognize a binary number quantity is a distinct disadvantage in working with this code despite the numerous hardware advantages. Engineers recognized this problem early and developed a special form of binary code that was more compatible with the decimal system. Because so many digital devices, instruments and equipment use decimal input and output, this special code has become very widely used and accepted. This special compromise code is known as binary coded decimal (BCD). The BCD code combines some of the characteristics of both the binary and decimal number systems.

8421 BCD Code The BCD code is a system of representing the decimal digits 0 through 9 with a four-bit binary code. This BCD code uses the standard 8421 position **weighting system** of the pure binary code. The standard 8421 BCD code and the decimal equivalents are shown in Figure 1-11, along with a special Gray code that will be described later. As with the pure binary code, you can convert the BCD numbers into their decimal equivalents by simply adding together the weights of the bit positions whereby the binary 1's occur. Note, however, that there are only ten possible valid 4-bit code arrangements. The 4-bit binary numbers representing the decimal numbers 10 through 15 are invalid in the BCD system.

DECIMAL	8421 BCD	GRAY	BINARY
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0010
3	0011	0010	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	0101	0110
7	0111	0100	0111
8	1000	1100	1000
9	1001	1101	1001
10	0001 0000	1111	1010
11	0001 0001	1110	1011
12	0001 0010	1010	1100
13	0001 0011	1011	1101
14	0001 0100	1001	1110
15	0001 0101	1000	1111

Figure 1-11
Codes.

To represent a decimal number in BCD notation, substitute the appropriate 4-bit code for each decimal digit. For example, the decimal integer 834 in BCD would be 1000 0011 0100. Each decimal digit is represented by its equivalent 8421 4-bit code. A space is left between each 4-bit group to avoid confusing the BCD format with the pure binary code. This method of representation also applies to decimal fractions. For example, the decimal fraction 0.764 would be 0.0111 0110 0100 in BCD. Again, each decimal digit is represented by its equivalent 8421 4-bit code, with a space between each group.

An advantage of the BCD code is that the ten BCD code combinations are easy to remember. Once you begin to work with binary numbers regularly, the BCD numbers may come to you as quickly and automatically as decimal numbers. For that reason, by simply glancing at the BCD representation of a decimal number you can make the conversion almost as quickly as if it were already in decimal form. As an example, convert a BCD number into its decimal equivalent.

$$0110\ 0010\ 1000.1001\ 0101\ 0100 = 628.954_{10}$$

The BCD code simplifies the man-machine interface but it is less efficient than the pure binary code. It takes more bits to represent a given decimal number in BCD than it does with pure binary notation. For example, the decimal number 83 in pure binary form is 1010011. In BCD code the decimal number 83 is written as 1000 0011. In the pure binary code, it takes only seven bits to represent the number 83. In BCD form, it takes eight bits. It is inefficient because, for each bit in a data word, there is usually some digital circuitry associated with it. The extra circuitry associated with the BCD code costs more, increases equipment complexity, and consumes more power. Arithmetic operations with BCD numbers are also more time consuming and complex than those with pure binary numbers. With four bits of binary information, you can represent a total of $2^4 = 16$ different states or the decimal number equivalents 0 through 15. In the BCD system, six of these states (10-15), are wasted. When the BCD number system is used, some efficiency is traded for the improved communications between the digital equipment and the human operator.

Decimal-to-BCD conversion is simple and straightforward. However, binary-to-BCD conversion is not direct. An intermediate conversion to decimal must be performed first. For example, the binary number 1011.01 is converted into its BCD equivalent.

First the binary number is converted to decimal.

$$\begin{aligned} 1011.01_2 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) \\ &= 8 + 0 + 2 + 1 + 0 + 0.25 \\ &= 11.25_{10} \end{aligned}$$

Then the decimal result is converted to BCD.

11.25_{10} = 0001 0001.0010 0101

To convert from BCD to binary, the previous operation is reversed. For example, the BCD number 1001 0110.0110 0010 0101 is converted into its binary equivalent.

First, the BCD number is converted to decimal.
$1001\ 0110.0110\ 0010\ 0101 = 96.625_{10}$
Then the decimal result is converted to binary.
$96.625_{10} = 96_{10} + 0.625_{10}$

$96 \div 2 = 48$	with remainder	0	← LSB
$48 \div 2 = 24$		0	
$24 \div 2 = 12$		0	
$12 \div 2 = 6$		0	
$6 \div 2 = 3$		0	
$3 \div 2 = 1$		1	
$1 \div 2 = 0$		1	← MSB

$96_{10} = 1100000_2$

$0.625 \times 2 = 1.25 = 0.25$	with overflow	1	← MSB
$0.250 \times 2 = 0.50 = 0.50$		0	
$0.500 \times 2 = 1.00 = 0$		1	← LSB

$0.625_{10} = 0.101_2$

$96.625_{10} = 96_{10} + 0.625_{10} = 1100000_2 + 0.101_2 = 1100000.101_2$

Therefore:

$1001\ 0110.0110\ 0010\ 0101 = 96.625_{10} = 1100000.101_2$

Because the intermediate decimal number contains both an integer and fraction, each number portion is converted as described under "Binary Number System." The binary sum (integer plus fraction) 1100000.101 is equivalent to the BCD number 1001 0110.0110 0010 0101.

DECIMAL	8421 BCD	GRAY	BINARY
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0010
3	0011	0010	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	0101	0110
7	0111	0100	0111
8	1000	1100	1000
9	1001	1101	1001
10	0001 0000	1111	1010
11	0001 0001	1110	1011
12	0001 0010	1010	1100
13	0001 0011	1011	1101
14	0001 0100	1001	1110
15	0001 0101	1000	1111

Figure 1-11
Codes.

Special Binary Codes

Besides the standard pure binary coded form, the BCD numbering system is by far the most widely-used digital code. You will find one or the other in most of the applications that you encounter. However, there are several other codes that are used for special applications, such as the "Gray Code."

The Gray Code is a widely-used, non-weighted code system. Also known as the cyclic, unit distance or reflective code, the Gray code can exist in either the pure binary or BCD formats. The Gray code is shown in Figure 1-11. As with the pure binary code, the first ten codes are used in BCD operations. Notice that there is a change in only one bit from one code number to the next in sequence. You can get a better idea about the Gray code sequence by comparing it to the standard 4-bit 8421 BCD code and the pure binary code also shown in Figure 1-11. For example, consider the change from 7 (0111) to 8 (1000) in the pure binary code. When this change takes place, all bits change. Bits that were 1's are changed to 0's and 0's are changed to 1's. Now notice the code change from 7 to 8 in the Gray code. Here 7 (0100) changes to 8 (1100). Only the first bit changes.

The Gray code is generally known as an error minimizing code because it greatly reduces confusion in the electronic circuitry when changing from one state to the next. When binary codes are implemented with electronic circuitry, it takes a finite period of time for bits to change from 0 to 1 or 1 to 0. These state changes can create timing and speed problems. This is particularly true in the standard 8421 codes where many bits change from one combination to the next. When the Gray code is used, however, the timing and speed errors are greatly minimized because only one bit changes at a time. This permits code circuitry to operate at higher speeds with fewer errors.

The biggest disadvantage of the Gray code is that it is difficult to use in arithmetic computations. Where numbers must be added, subtracted or used in other computations, the Gray code is not applicable. In order to perform arithmetic operations, the Gray code number must generally be converted into pure binary form.

Alphanumeric Codes

Several binary codes are called alphanumeric codes because they are used to represent characters as well as numbers. The two most common codes that will be discussed are ASCII and BAUDOT.

ASCII Code The American Standard Code for Information Interchange commonly referred to as ASCII, is a special form of binary code that is widely used in microprocessors and data communications equipment. A new name for this code that is becoming more popular is the American National Standard Code for Information Interchange (ANSCII). However, this course will use the most recognized term, ASCII. ASCII is a 6-bit binary code that is used in transferring data between microprocessors and their peripheral devices, and in communicating data by radio and telephone. With six bits, a total of $2^6 = 64$ different characters can be represented. These characters comprise decimal numbers 0 through 9, upper-case letters of the alphabet, plus other special characters used for punctuation and data control. A 7-bit code called full ASCII, extended ASCII, or USASCII can be represented by $2^7 = 128$ different characters. In addition to the characters and numbers generated by 6-bit ASCII, 7-bit ASCII contains lower-case letters of the alphabet, and additional characters for punctuation and control. The 7-bit ASCII code is shown in Figure 1-12.

COLUMN	0 (3)	1 (3)	2 (3)	3	4	5	6	7 (3)
ROW	BITS 765 / 4321	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	\	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[	k	{
12	1100	FF	FS	,	<	L	\	l	¦
13	1101	CR	GS	–	=	M	]	m	}
14	1110	SO	RS	.	>	N	^ (1)	n	~
15	1111	SI	US	/	?	O	— (2)	o	DEL

Figure 1-12
Table of 7-bit American Standard Code for Information Interchange.

NOTES:

(1) Depending on the machine using this code, the symbol may be a circumflex, an up-arrow, or a horizontal parenthetical mark.

(2) Depending on the machine using this code, the symbol may be an underline, a back-arrow, or a heart.

(3) Explanation of special control functions in columns 0, 1, 2, and 7.

NUL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC1	Device Control 1
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3
EOT	End of Transmission	DC4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous Idle
BEL	Bell (audible signal)	ETB	End of Transmission Block
BS	Backspace	CAN	Cancel
HT	Horizontal Tabulation (punched card skip)	EM	End of Medium
		SUB	Substitute
LF	Line Feed	ESC	Escape
VT	Vertical Tabulation	FS	File Separator
FF	Form Feed	GS	Group Separator
CR	Carriage Return	RS	Record Separator
SO	Shift Out	US	Unit Separator
SI	Shift In	DEL	Delete
SP	Space (blank)		

Figure 1-12
(Continued.)

The 7-bit ASCII code for each number, letter or control function is made up of a 4-bit group and a 3-bit group. Figure 1-13 shows the arrangement of these two groups and the numbering sequence. The 4-bit group is on the right and bit 1 is the LSB. Note how these groups are arranged in rows and columns in Figure 1-12.

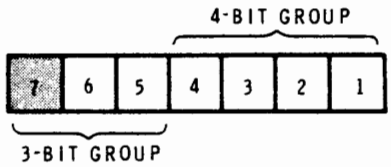


Figure 1-13
ASCII code word format.

To determine the ASCII code for a given number letter or control operation, locate that item in the table. Then use the 3- and 4-bit codes associated with the row and column in which the item is located. For example, the ASCII code for the letter L is 1001100. It is located in column 4, row 12. The most significant 3-bit group is 100, while the least significant 4-bit group is 1100. When 6-bit ASCII is used, the 3-bit group is reduced to a 2-bit group as shown in Figure 1-14.

In 7-bit ASCII code, an eighth bit is often used as a **parity** or check bit to determine if the data (character) has been transmitted correctly. The value of this bit is determined by the type of parity desired. **Even parity** means the sum of all the 1 bits, including the parity bit, is an even number. For example, if G is the character transmitted, the ASCII code is 1000111. Since four 1's are in the code, the parity bit is 0. The 8-bit code would be written 01000111.

OddParity means the sum of all the 1 bits, including the parity bit, is an odd number. If the ASCII code for G was transmitted with odd parity, the binary representation would be 11000111.

ROW	COLUMN	0	1	2	3
	BITS 4321 65	10	11	00	01
0	0000	SP(3)	0	@	P
1	0001	!	1	A	Q
2	0010	"	2	B	R
3	0011	#	3	C	S
4	0100	$	4	D	T
5	0101	%	5	E	U
6	0110	&	6	F	V
7	0111	'	7	G	W
8	1000	(	8	H	X
9	1001	)	9	I	Y
10	1010	*	:	J	Z
11	1011	+	;	K	[
12	1100	,	<	L	\
13	1101	–	=	M	]
14	1110	.	>	N	^ (1)
15	1111	/	?	O	— (2)

Figure 1-14
Table of 6-bit American Standard Code for Information Interchange.

NOTES:

(1) Depending on the machine using this code, the symbol may be a circumflex, an up-arrow, or a horizontal parenthetical mark.

(2) Depending on the machine using this code, the symbol may be an underline, a back-arrow, or a heart.

(3) SP — Space (blank) for machine control.

BAUDOT Code While the ASCII code is used almost exclusively with microprocessor peripheral devices (CRT display, keyboard terminal, paper punch/reader, etc.), there are many older printer peripherals that use the 5-bit BAUDOT code. With five data bits, this code can represent only $2^5 = 32$ different characters. To obtain a greater character capability, 26 of the 5-bit codes are used to represent two separate characters. As shown in Figure 1-15, one set of 5-bit codes represents the 26 upper-case alphabet letters. The same 5-bit codes also represent various figures and the decimal number series 0 through 9.

The remaining six 5-bit codes are used for machine control and do not have a secondary function. Two of these 5-bit codes determine which of the 26 double (letter/figure) characters can be transmitted/received. Bit number 11111 forces the printer to recognize all following 5-bit codes as **letters**. Bit number 11011 forces **figure** recognition of all the following 5-bit codes. For example, to type 56 NORTH 10 STREET, the following method is used.

Type — Figures 5 6 Space

Then — Letters N O R T H Space

Then — Figures 1 0 Space

Finally — Letters S T R E E T

Bit Numbers 5 4 3 2 1	Letters Case	Figures Case
0 0 0 0 0	Blank	Blank
0 0 0 0 1	E	3
0 0 0 1 0	Line Feed	Line Feed
0 0 0 1 1	A	—
0 0 1 0 0	Space	Space
0 0 1 0 1	S	Bell
0 0 1 1 0	I	8
0 0 1 1 1	U	7
0 1 0 0 0	Car. Ret.	Car. Ret.
0 1 0 0 1	D	$
0 1 0 1 0	R	4
0 1 0 1 1	J	(Apos)'
0 1 1 0 0	N	(Comma),
0 1 1 0 1	F	!
0 1 1 1 0	C	:
0 1 1 1 1	K	(
1 0 0 0 0	T	5
1 0 0 0 1	Z	"
1 0 0 1 0	L	)
1 0 0 1 1	W	2
1 0 1 0 0	H	Stop
1 0 1 0 1	Y	6
1 0 1 1 0	P	0
1 0 1 1 1	Q	1
1 1 0 0 0	O	9
1 1 0 0 1	B	?
1 1 0 1 0	G	&
1 1 0 1 1	Figures	Figures
1 1 1 0 0	M	.
1 1 1 0 1	X	/
1 1 1 1 0	V	;
1 1 1 1 1	Letters	Letters

Figure 1-15
5-bit BAUDOT code table.

Self-Test Review

26. The BCD code is more convenient to use than the binary code because:

    A. it uses less bits.
    B. it is more compatible with the decimal number system.
    C. it is more adaptable to arithmetic computations.
    D. there are more different coding schemes available.

27. Convert the following decimal numbers to 8421 BCD code.

    A. 1049
    B. 267
    C. 835

28. Convert the following 8421 BCD code numbers to decimal.

    A. 1001 0110 0010
    B. 0111 0001 0100 0011
    C. 1010 1001 1000
    D. 1000 0000 0101

29. Convert the following binary numbers to 8421 BCD code.

    A. 101110.01
    B. 1001.0101
    C. 11011011.0001

30. Convert the following 8421 BCD codes to binary.

    A. 0001 1000 0010.0101
    B. 0010 1001 0000.0010 0101
    C. 1101 0110 0011.0101
    D. 0110 1000.0001 0010 0101

31. Which code is best for error minimizing?

    A. 8421 BCD
    B. pure binary
    C. Gray

32. The ASCII and BAUDOT codes are a form of _____ _____ codes.

33. To determine if the correct ASCII character has been transmitted, a _____ bit is often added to the code.

34. Which type of parity is used when the 8-bit ASCII character 01000111 is transmitted?

    A. odd
    B. even

35. Refer to Figure 1-12 and convert the following characters into their ASCII 7-bit binary code.

    A. B
    B. X
    C. 3
    D. S

36. Refer to Figure 1-12 and convert the following ASCII 7-bit binary codes to their character equivalents.

    A. 0110010
    B. 1010110
    C. 1011010
    D. 1001110

Answers

26. B. More compatible with the decimal system.

27. A. 0001 0000 0100 1001
    B. 0010 0110 0111
    C. 1000 0011 0101

28. A. 962
    B. 7143
    C. Invalid (1010 represents a decimal digit greater than 9)
    D. 805

29. A. $101110.01_2 = (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2})$
    $= 32 + 0 + 8 + 4 + 2 + 0 + 0 + 0.25$
    $= 46.25_{10}$
    $46.25_{10} = 0100\ 0110.0010\ 0101$
    $101110.01_2 = 0100\ 0110.0010\ 0101$
    B. $1001.0101_2 = 1001.0011\ 0001\ 0010\ 0101$
    C. $11011011.0001_2 = 0010\ 0001\ 1001.0000\ 0110\ 0010\ 0101$

30. A. $0001\ 1000\ 0010.0101 = 182.5_{10}$

$182.5_{10} = 182_{10} + 0.5_{10}$

$182 \div 2 = 91$	with remainder	0	← LSB
$91 \div 2 = 45$		1	
$45 \div 2 = 22$		1	
$22 \div 2 = 11$		0	
$11 \div 2 = 5$		1	
$5 \div 2 = 2$		1	
$2 \div 2 = 1$		0	
$1 \div 2 = 0$		1	← MSB

$\boxed{182_{10} = 10110110_2}$

$0.5 \times 2 = 1.00 = 0$ overflow 1 ← MSB, LSB

$\boxed{0.5_{10} = 0.1_2}$

$182.5_{10} = 182_{10} + 0.5_{10} = 10110110_2 + 0.1_2 = 10110110.1_2$

$0001\ 1000\ 0010.0101 = 10110110.1_2$

B. $0010\ 1001\ 0000.0010\ 0101 = 100100010.01_2$
C. invalid (1101 represents a decimal digit greater than 9)
D. $0110\ 1000.0001\ 0010\ 0101 = 1000100.001_2$

31. Gray

32. Alpha numeric

33. Parity

34. Even

35. A. 1000010
B. 1011000
C. 0110011
D. 1010011

36. A. 2
B. V
C. Z
D. N

EXPERIMENTS 1 AND 2

Perform Experiments 1 and 2 in the Experiment Section, Unit 9 of the course. After you finish the experiment, return to this unit and complete the "Unit Examination."

UNIT EXAMINATION

This examination will test your knowledge of the important facts in this unit. It will tell you what you have learned and what you need to review. Answer all questions first; then check your work against the correct answers given later.

1. Indicate the base or radix of the following number systems.

   A. Octal _____.
   B. Decimal _____.
   C. Hexadecimal _____.
   D. Binary _____.

2. Write the following numbers using positional notation.

   A. 1101.011_2
   B. 1010.01_{10}
   C. 1001.101_8.
   D. 1110.11_{16}

3. Convert the following numbers to decimal.

   A. 10011.011_2
   B. 752.31_8
   C. $A8C.5F_{16}$

4. Convert the following numbers to binary.

   A. 105.0625_{10}
   B. 374.24_8
   C. $F19.6C_{16}$

5. Convert the following numbers to octal.

   A. 638.3125_{10}
   B. 10010101.0110101_2

6. Convert the following numbers to hexadecimal.

   A. 9587.03125_{10}
   B. $1101101101010.101110101_2$

7. The ASCII and BAUDOT codes are a form of _____ _____ codes.

8. Convert the following numbers to 8421 BCD code.

   A. 521.372_{10}
   B. 1010.011_2

9. Convert the 8421 BCD code 1001 0101.0111 0011 to decimal.

10. Convert the 8421 BCD code 0101 0011.0111 0101 to binary.

11. Which type of parity is used when the 8-bit ASCII character 01110111 is transmitted?

    A. Odd
    B. Even

12. The BAUDOT code uses _____ bit numbers to generate a character.

13. Using only your knowledge of binary codes, identify the Gray code.

Decimal	a	b	c	d
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0011	0010	0010	0101
3	0010	0011	0011	0110
4	0110	0100	0100	0111
5	0111	0101	1011	1000
6	0101	0110	1100	1001
7	0100	0111	1101	1010
8	1100	1000	1110	1011
9	1101	1001	1111	1100