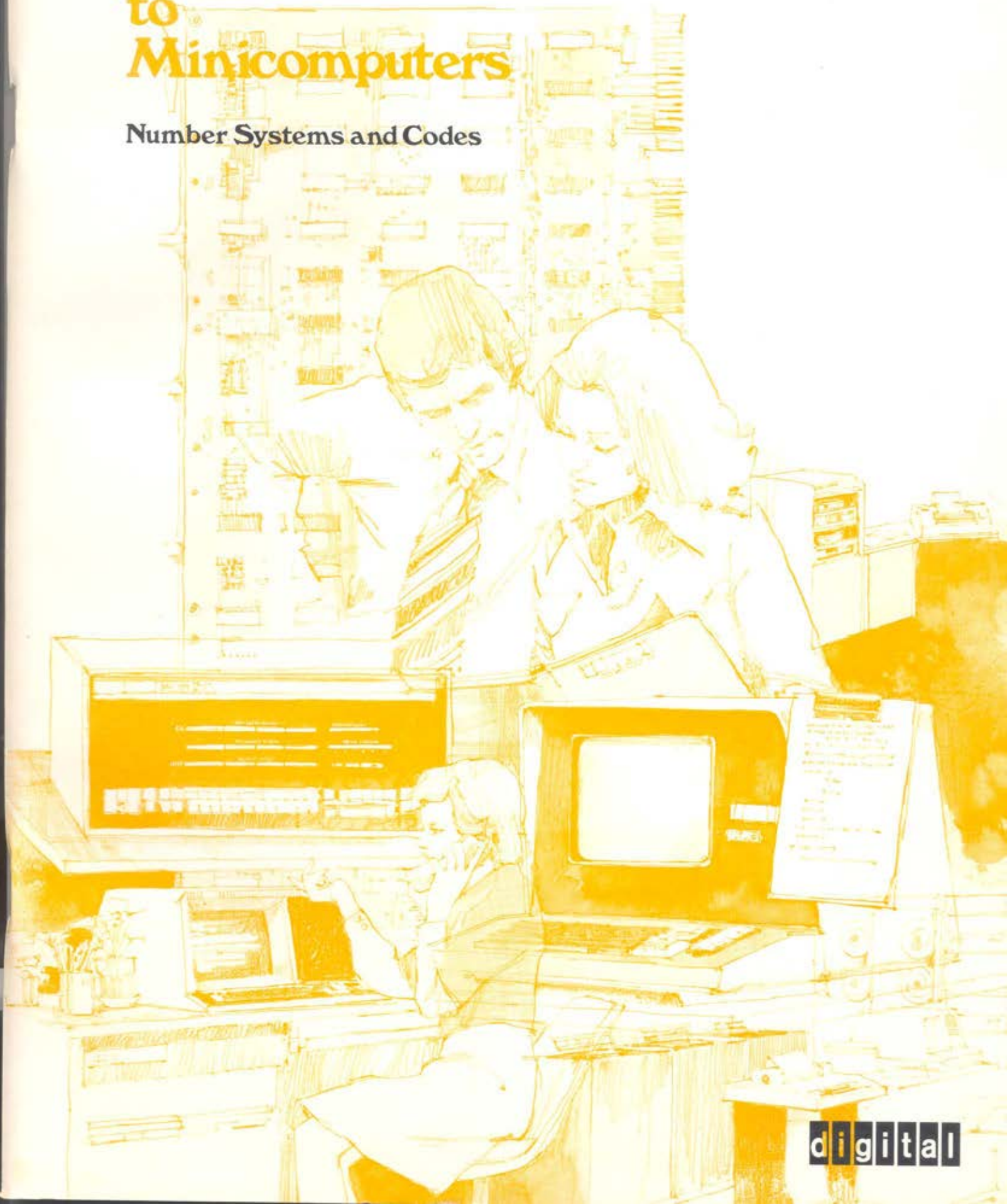
Introduction to Minicomputers

Number Systems and Codes



digital

1st Printing, June 1976
2nd Printing (Rev), October 1977
3rd Printing, August 1979

Printed in U.S.A.

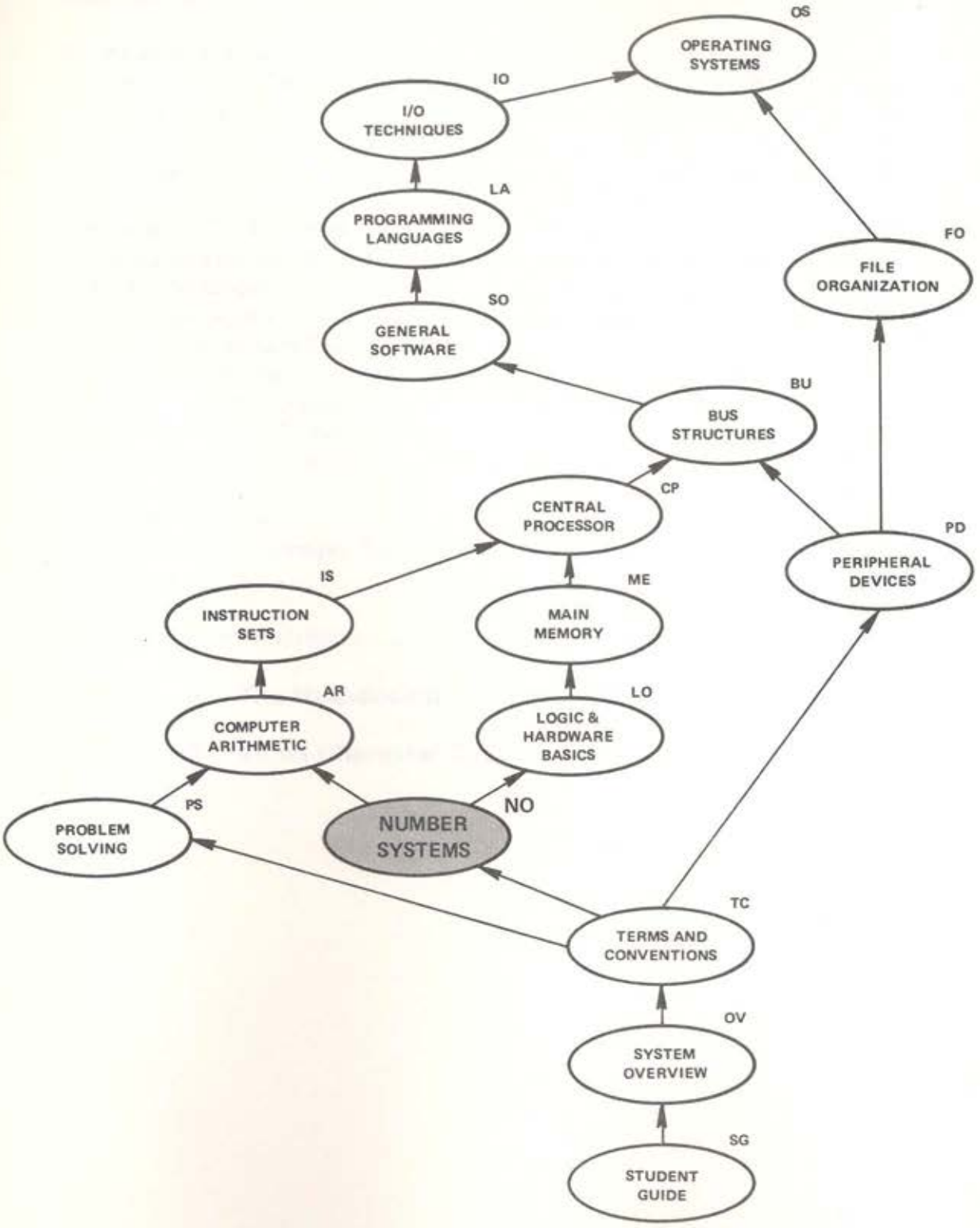
INTRODUCTION TO MINICOMPUTERS

Number Systems and Codes

Student Workbook

Audio-Visual Course by Digital Equipment Corporation

COURSE MAP


OS
OPERATING SYSTEMS
IO
I/O TECHNIQUES
LA
PROGRAMMING LANGUAGES
SO
GENERAL SOFTWARE
FO
FILE ORGANIZATION
BU
BUS STRUCTURES
CP
CENTRAL PROCESSOR
PD
PERIPHERAL DEVICES
IS
INSTRUCTION SETS
ME
MAIN MEMORY
AR
COMPUTER ARITHMETIC
LO
LOGIC & HARDWARE BASICS
PS
PROBLEM SOLVING
NUMBER SYSTEMS
NO
TC
TERMS AND CONVENTIONS
OV
SYSTEM OVERVIEW
SG
STUDENT GUIDE

CONTENTS

Number Systems and Codes

Introduction

When people process information manually, they deal with alphabetic and numeric characters. However, because digital computers are binary machines, they represent information *internally* with 1s and 0s. Therefore, computers must perform mathematical operations using the binary number system and must represent alphanumeric information using binary codes.

To prepare information effectively for computer processing, people must understand how the computer will manipulate the information. This lesson not only covers binary numbers and coding, but also presents the octal number system, which many people use as a convenient, shorter form in place of binary when dealing with computer information.

This module contains three lessons. The first lesson describes the *binary number system* that computers use, the *octal number system* that people use to represent binary, and the *decimal number system* that we all use in our everyday lives. The second lesson describes *methods for converting numbers* from one system to another. Finally, the third lesson discusses *binary codes* that computers use to represent alphanumeric information.

Number Systems

OBJECTIVES

1. Given the names of three number systems, be able to write both the base and the range of actual values for each number system.

2. Given a number, be able to write the actual and place values of any digit within the number.

3. Given a number in decimal, octal, or binary form, be able to: (1) identify the decimal equivalent of the actual and place values for any digit in the number; and (2) write the identified equivalent in decimal, octal, or binary form, as required.

4. Given consecutive decimal numbers, be able to write the octal and binary equivalents of each decimal number.

SAMPLE TEST ITEMS

1. Complete the table below.

Number System	Base	Actual Values
Decimal	_____	_____
Octal	_____	_____
Binary	_____	_____

2. Give the actual value and place value of the digit 8 in the number 3859. (Circle one answer for each.)

   Actual Value = _____ Place Value = _____

SAMPLE TEST ITEMS

3. Decimal Equivalents (Circle one answer only.)

The decimal equivalent of the *actual value* of the digit 9 in the *decimal* number 1978 is _____ .

a) 10 b) 9 c) 81 d) 90 e) 900

The decimal equivalent of the *place value* of the digit 9 in the *decimal* number 1978 is _____ .

a) 9 b) 90 c) 900 d) 100 e) 1000

The decimal equivalent of the *actual value* of the digit 5 in the *octal* number 2756 is _____ .

a) 1 b) 8 c) 5 d) 10 e) 4

The decimal equivalent of the *place value* of the digit 7 in the *octal* number 2756 is _____ .

a) 800 b) 100 c) 8 d) 64 e) 512

The decimal equivalent of the *place value* of the digit 1 in the *binary* number 1000 is _____ .

a) 32 b) 16 c) 8 d) 2 e) 1

4. Write the octal and binary equivalents for each decimal number in the space provided. (Note that several answers are already given to you.)

Decimal	Octal	Binary
3	3	11
4		
5		
6		
7		
8		
. . .		

Mark your place in this workbook and view Lesson 1 in the A/V program, "Number Systems and Codes."

The three most common number systems are: the *decimal system,* which we use in our everyday work; the *binary system,* which digital computers use because it is easy to implement electronically; and the *octal* system, which computer operators and programmers use to represent binary numbers because it is easier to work with. Octal numbers, however, must still be converted to binary before the computer can process them.

As shown in Table 1, each number system has a unique *base* or *radix.* This base corresponds to the number of *digits* or *actual values* that are used in that number system. For example, the decimal number system uses 10 digits, 0 through 9, and therefore has a radix of 10.

NOTES

Here we are referring to the decimal number system as whole numbers (integers) only. No quantities are expressed using fractions or a decimal point.

The actual (absolute) value of any digit in any number system is fixed.

Table 1 Actual Values

Number System	Base (Radix)	Actual Values (digits)
Decimal	10	0 through 9
Octal	8	0 through 7
Binary	2	0 *and* 1

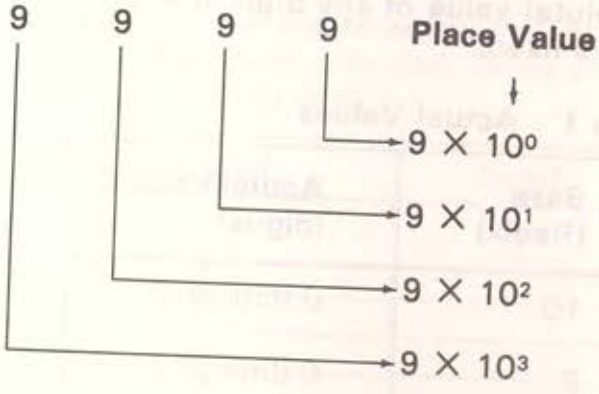
Place Values

We know that, in the decimal system, two or more digits are needed to represent any value greater than 9. Whenever a number contains two or more digits, each digit is assigned a specific value called a *place* or *positional value*. This place value equals the base of the number system raised to some *power*. In the decimal system, place values are based on powers of ten (that is, units, tens, hundreds, thousands, etc.) The place values for a 5-digit decimal number are given in Table 2. Note that both the place value in powers of ten and the decimal equivalent of the place value are given for each digit position.

Table 2 Decimal Place Values

Digit Position	5	4	3	2	1
Place Value	10^4	10^3	10^2	10^1	10^0
Decimal Equivalent of Place Value	10,000	1,000	100	10	1

For example, in a decimal number, such as 9999, the meaning of each digit is the actual value of the digit, multiplied by the place value as follows:



Although the actual value of the digit 9 is fixed, its place value is different for each position in the number. The further left the digit is located, the greater its place value. Notice that the place value increases by a power of ten (we are dealing with a base ten system) each time the digit is moved one place to the left. By convention, the leftmost digit is always referred to as the *most significant digit* (MSD), and the rightmost digit is always referred to as the *least significant digit* (LSD).

Place values, however, are *not* unique to the decimal system. They are also used in binary and octal number systems. Notice that in the *binary* system, place values are based on powers of *two* rather than ten, and in the *octal* system place values are based on powers of *eight*. Table 3 shows the place values for the binary and octal number systems with their decimal equivalent.

Table 3 Binary and Octal Place Values

Decimal	10^4	10^3	10^2	10^1	10^0
Decimal Equivalent	10,000	1,000	100	10	1

Octal	8^4	8^3	8^2	8^1	8^0
Decimal Equivalent	4096	512	64	8	1

Binary	2^4	2^3	2^2	2^1	2^0
Decimal Equivalent	16	8	4	2	1

Remember that any base raised to the zero power is always one, and any base raised to the first power is the number of the base itself.

Counting

Whenever the base of *any number system is equaled,* a *zero* is placed in that position, and the next most significant position is increased by a one. This is known as a *carry*. Examples of counting, using this carry principle, are shown in Table 4.

Table 4 Counting

Decimal	Octal	Binary
0	0	0
1	1	1
2	2	one carry 10
3	3	11
4	4	two carries 100
5	5	101
6	6	one carry 110
7	7	111
8	one carry 10	three carries 1000
9	11	1001
one carry 10	12	one carry 1010
.	.	.
.	.	.
.	.	.

NOTE

It takes more digits to represent a number if the base is small. In other words, the smaller the base, the more numbers needed. For example, the number nine requires four binary bits, two octal digits or one decimal digit.

Table 5 illustrates counting with larger numbers. Follow the sequence of numbers in each base until you feel that you can count in each number system. Then read on.

Table 5 Counting

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
.	.	.
.	.	.
.	.	.
.	.	.
500	764	111 110 100
501	765	111 110 101
502	766	111 110 110
503	767	111 110 111
504	770	111 111 000
505	771	111 111 001
506	772	111 111 010
507	773	111 111 011
508	774	111 111 100
509	775	111 111 101
510	776	111 111 110
511	777	111 111 111
512	1000	1 000 000 000
513	1001	1 000 000 001

To avoid confusion when working with two or more number systems, the radix (base) is normally appended as a subscript to the number.

Examples:

967_{10} (decimal 967)

234_{8} (octal 234)

101_{2} (binary 101)

Before proceeding to the next lesson. do the practice exercises that begin on the following page. There are 20 exercises.

EXERCISES

1. Write the name of the number system used in each of the following applications:

   a. Used in normal, everyday work _____

   b. Used in digital computers _____

   c. Used by programmers _____

2. Fill in the *base* and the *actual values* (digits) for each of the number systems listed in the following table:

Number System	Base	Actual Values (Digits)
Decimal		
Octal		
Binary		

3. Write the name of the appropriate number system indicated for each of the following numbers:

   a. 7346_8 _____

   b. 1733_{10} _____

   c. 1110_2 _____

SOLUTIONS

1. Write the name of the number system used in each of the following applicatons:

   a. Used in normal, everyday work — decimal

   b. Used in digital computers — binary

   c. Used by programmers — octal

2. Fill in the *base* and the *actual values* (digits) for each of the number systems listed in the following table:

Number System	Base	Actual Values (Digits)
Decimal	10	0 through 9
Octal	8	0 through 7
Binary	2	0 and 1

3. Write the name of the appropriate number system indicated for each of the following numbers:

   a. 7346_8 — octal

   b. 1733_{10} — decimal

   c. 1110_2 — binary

EXERCISES

4. Another name for "base" is _____ .

5. What number system is used by digital computers?

   a. Decimal b. Binary c. Octal

6. What is the most significant digit (MSD) in the number 27613?

   a. 1 b. 2 c. 3 d. 6 e. 7

7. What is the least significant digit (LSD) in the number 27613?

   a. 1 b. 2 c. 3 d. 6 e. 7

8. What is the value of any number raised to the *zero* power?

   a. The number itself b. 0 c. 1 d. 2

9. What is the value of any number raised to the first power?

   a. The number itself b. 0 c. 1 d. 2

SOLUTIONS

4. Another name for "base" is "radix" .

5. What number system is used by digital computers?

   a. Decimal (b. Binary) c. Octal

6. What is the most significant digit (MSD) in the number 27613?

   a. 1 (b. 2) c. 3 d. 6 e. 7

7. What is the least significant digit (LSD) in the number 27613?

   a. 1 b. 2 (c. 3) d. 6 e. 7

8. What is the value of any number raised to the *zero* power?

   a. The number itself b. 0 (c. 1) d. 2

9. What is the value of any number raised to the first power?

   (a. The number itself) b. 0 c. 1 d. 2

EXERCISES

10. Fill in the table below with the place values of the digit positions and their decimal equivalents for each indicated number system.

Number System	Digit Position	5	4	3	2	1
Decimal	Place Value					
	Decimal Equivalent					
Octal	Place Value					
	Decimal Equivalent					
Binary	Place Value					
	Decimal Equivalent					

11. What is the decimal equivalent of the *actual value* of the digit 7 in the *decimal* number 1978?

    a. 7 b. 10 c. 49 d. 70 e. 700

12. What is the decimal equivalent of the *place value* of the digit 7 in the *decimal* number 1978?

    a. 7 b. 10 c. 70 d. 100 e. 700

13. What is the decimal equivalent of the *actual value* of the digit 7 in the *octal* number 2376?

    a. 1 b. 7 c. 8 d. 10 e. 56

10. Fill in the table below with the place values of the digit positions and their decimal equivalents for each indicated number system.

Number System	Digit Position	5	4	3	2	1
Decimal	Place Value	10^4	10^3	10^2	10^1	10^0
	Decimal Equivalent	10,000	1,000	100	10	1
Octal	Place Value	8^4	8^3	8^2	8^1	8^0
	Decimal Equivalent	4,096	512	64	8	1
Binary	Place Value Decimal	2^4	2^3	2^2	2^1	2^0
	Equivalent	16	8	4	2	1

11. What is the decimal equivalent of the *actual value* of the digit 7 in the *decimal* number 1978?

(a. 7) b. 10 c. 49 d. 70 e. 700

12. What is the decimal equivalent of the *place value* of the digit 7 in the *decimal* number 1978?

a. 7 (b. 10) c. 70 d. 100 e. 700

13. What is the decimal equivalent of the *actual value* of the digit 7 in the *octal* number 2376?

a. 1 (b. 7) c. 8 d. 10 e. 56

EXERCISES

14. What is the decimal equivalent of the *place value* of the digit 7 in the *octal* number 2376?

    a. 7 b. 8 c. 10 d. 56 e. 64

15. What is the decimal equivalent of the *place value* of the digit 5 in the *decimal* number 1549?

    a. 5 b. 10 c. 50 d. 100 e. 500

16. What is the decimal equivalent of the *place value* of the digit 5 in the *octal* number 2567?

    a. 5 b. 8 c. 40 d. 64 e. 320

17. What is the decimal equivalent of the *place value* of the digit 1 in the *binary* number 10000?

    a. 1 b. 2 c. 8 d. 16 e. 32

18. What is the decimal equivalent of the *place value* of the digit 1 in the *center* of the *binary* number 10101?

    a. 1 b. 2 c. 3 d. 4 e. 8

19. If we are counting and have used all the available digits in a number system, two actions are necessary to count further:

    a. In the digit position we are currently using, we put down a _____.

    b. In the next column to the left, we introduce a _____.

SOLUTIONS

14. What is the decimal equivalent of the *place value* of the digit 7 in the *octal* number 2376?

a. 7 (b. 8) c. 10 d. 56 e. 64

15. What is the decimal equivalent of the *place value* of the digit 5 in the *decimal* number 1549?

a. 5 b. 10 c. 50 (d. 100) e. 500

16. What is the decimal equivalent of the *place value* of the digit 5 in the *octal* number 2567?

a. 5 b. 8 c. 40 (d. 64) e. 320

17. What is the decimal equivalent of the *place value* of the digit 1 in the *binary* number 10000?

a. 1 b. 2 c. 8 (d. 16) e. 32

18. What is the decimal equivalent of the *place value* of the digit 1 in the *center* of the *binary* number 10101?

a. 1 b. 2 c. 3 (d. 4) e. 8

19. If we are counting and have used all the available digits in a number system, two actions are necessary to count further:

a. In the digit position we are currently using, we put down a zero .

b. In the next column to the left, we introduce a carry .

EXERCISES

20. Fill in the table below by counting in octal and binary.

Decimal	Octal	Binary
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
.	.	.
.	.	.
.	.	.
500	764	111110100
501		
502		
503		
504		
505		
506		
507		
508		
509		
510		
511		
512		
513		

20. Fill in the table below by counting in octal and binary.

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
.	.	.
.	.	.
.	.	.
.	.	.
500	764	111 110 100
501	765	111 110 101
502	766	111 110 110
503	767	111 110 111
504	770	111 111 000
505	771	111 111 001
506	772	111 111 010
507	773	111 111 011
508	774	111 111 100
509	775	111 111 101
510	776	111 111 110
511	777	111 111 111
512	1000	1 000 000 000
513	1001	1 000 000 001

Conversion Techniques

OBJECTIVE

Given a number in the decimal, octal, or binary number system, be able to convert the number to its decimal, octal, or binary equivalent.

SAMPLE TEST ITEMS

1. Circle the *decimal* equivalent of each of the following *binary* numbers:

   a. $101\ \ 111\ \ 110_2$

      1) 382_{10}
      2) 380_{10}
      3) 282_{10}
      4) 280_{10}

   b. $010\ \ 010\ \ 011_2$

      1) 139_{10}
      2) 136_{10}
      3) 146_{10}
      4) 147_{10}

2. Circle the *octal* equivalent of each of the following *decimal* numbers:

   a. 28_{10}

      1) 32_8
      2) 33_8
      3) 34_8
      4) 35_8

   b. 57_{10}

      1) 71_8
      2) 72_8
      3) 73_8
      4) 74_8

Mark your place in this workbook and view Lesson 2 in the A/V program, "Number Systems and Codes."

It is often necessary to convert from one number system to another. The six conversion techniques covered in this lesson are:

1. Binary to Decimal
2. Octal to Decimal
3. Decimal to Binary
4. Decimal to Octal
5. Binary to Octal
6. Octal to Binary

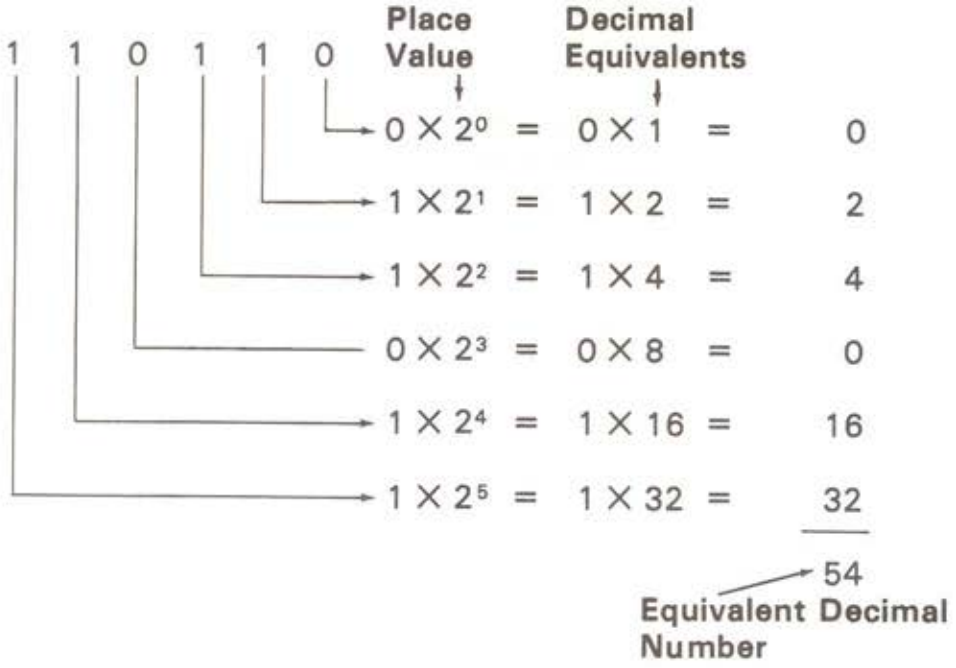
Binary-to-Decimal Conversion

When converting a *binary* (base 2) number to a decimal (base 10) number, two rules must be followed:

1. Multiply each binary digit by its corresponding place value.

2. Add the products.

Below is an example of converting the binary number *1 1 0 1 1 0* to decimal.



A simpler method would be to use the decimal equivalent of the *binary place values*. When this method is used, the place values corresponding to the binary 1s are added to get the decimal equivalent number.

For example, suppose we want the decimal value of binary *1 1 0 1 1 0*.

Decimal Equivalent of Binary Place Values →	32	16	8	4	2	1	
Binary Number →	1	1	0	1	1	0	
Therefore: →	32 +	16 +		4 +	2	=	54

(Equivalent Decimal Number)

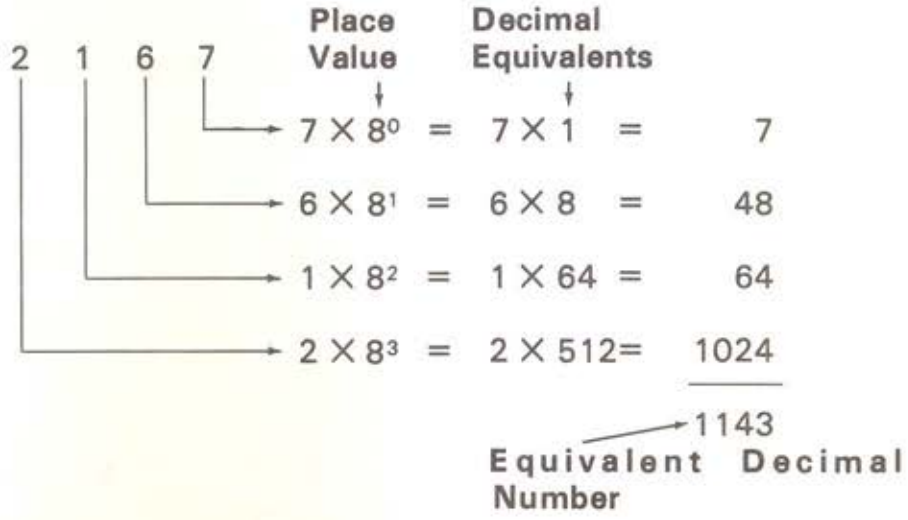
Octal-to-Decimal Conversion

When converting an *octal* (base 8) number to a decimal number, there are only two rules to be followed:

1. Multiply each octal digit by its corresponding place value.

2. Add the products.

   The example below shows how to convert the octal number 2167 to decimal.



Decimal-to-Binary Conversion

When converting a decimal number to a binary number, the following rules apply:

1. *Divide* the decimal number by 2 and save the remainder.
2. Divide the *quotient* from the previous division by 2 and save the remainder.
3. Continue step 2 until the quotient is zero.
4. The remainders saved from the division make up the digits of the binary number. The first remainder is the least significant digit (LSD), and the last remainder is the most significant digit (MSD).

The example below shows how the decimal number *38* is converted to binary.

Base	Number	Quotient	Remainder
2	$\overline{)38}$	19	0 (LSD)
2	$\overline{)19}$	9	1
2	$\overline{)9}$	4	1
2	$\overline{)4}$	2	0
2	$\overline{)2}$	1	0
2	$\overline{)1}$	0	1 (MSD)

Therefore: $38_{10} = 1\ 0\ 0\ 1\ 1\ 0_2$

Decimal-to-Octal Conversion

When converting a decimal number to an octal number, the following rules apply:

1. Divide the decimal number by 8 and save the remainder.
2. Divide the quotient from the previous division by 8 and save the remainder.
3. Continue step 2 until the quotient is zero.
4. The remainders produced by the divisions make up the digits of the octal number. The first remainder is the least significant digit (LSD) and the last remainder is the most significant digit (MSD).

The example below shows how the decimal number *1908* is converted to octal.

Base	Number	Quotient	Remainder
8	$\overline{)1908}$	238	4 (LSD)
8	$\overline{)238}$	29	6
8	$\overline{)29}$	3	5
8	$\overline{)3}$	0	3 (MSD)

Therefore: $1908_{10} = 3564_8$

Binary-to-Octal Conversion

Conversions between octal and binary number systems are relatively simple. Eight is an integral power of two, i.e., $8 = 2^3$. This means that *any octal digit is directly equivalent to three binary digits* and vice versa. This equivalency is shown in Table 6.

Table 6 Octal-Binary Equivalencies

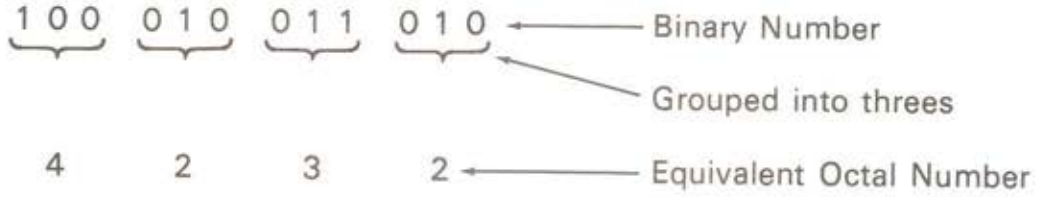
Octal Digit	Binary Digits
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Once you become familiar with these equivalencies, conversions between binary and octal number systems can be performed simply by inspection. The procedure is as follows:

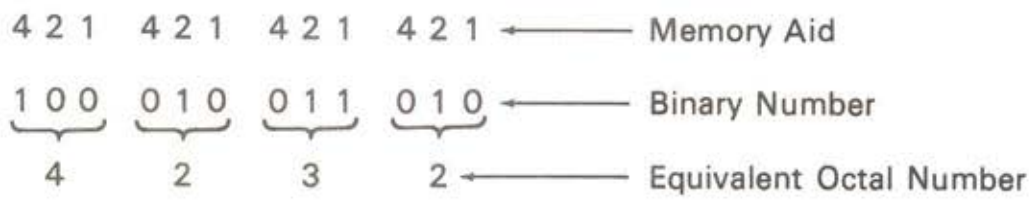
1. Start with the least significant digit in the binary number and arrange the digits into groups of three.

2. Replace each group of digits with the equivalent octal digit shown in Table 6.

The example below shows how binary *100010011010* can be converted to octal.



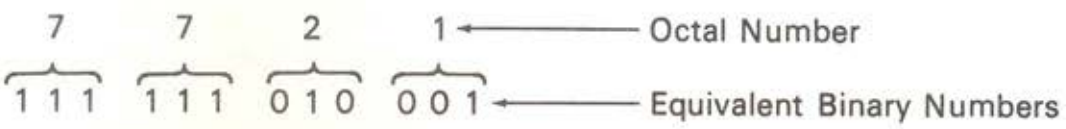
As a memory aid, the binary place values 4, 2, and 1 can be placed over each group of three binary digits to aid in finding their octal equivalent. Examples are shown below.



Octal-to-Binary Conversion

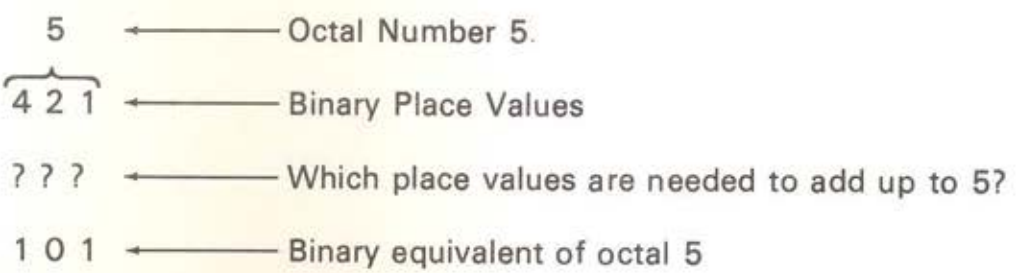
Since any octal digit is directly equivalent to three binary digits and vice versa, octal-to-binary conversion is also a simple matter. To convert an octal number to its binary equivalent, replace each octal digit with the equivalent *three binary digits* from Table 6.

The example below shows how octal *7721* is converted to binary.



Again, as a memory aid, the binary place values 4, 2, and 1 can be used to find the binary equivalent of each octal digit.

For example, assume that you need a binary number equivalent to the octal number 5. You can use the binary place values 4, 2, and 1 as follows: write down the place values 4, 2, and 1, and ask yourself which of these values add up to the octal number (in this case 5). Put 1s under the place values you need, and 0s under the rest.



Before going to the next lesson, do the practice exercises that begin on the following page. There are 6 exercises, each on a different type of conversion. Each exercise contains 10 problems. Do as many problems in each exercise as you feel necessary to master that type of conversion.

EXERCISES

1. Convert the following *binary* numbers to their *decimal* equivalents.

   a. $000\,010\,110_2$ _____ (decimal)

   b. $010\,100\,011_2$ _____ (decimal)

   c. $101\,000\,000_2$ _____ (decimal)

   d. $111\,010\,010_2$ _____ (decimal)

   e. $001\,111\,110_2$ _____ (decimal)

   f. $100\,011\,010_2$ _____ (decimal)

   g. $011\,011\,011_2$ _____ (decimal)

   h. $111\,000\,111_2$ _____ (decimal)

   i. $010\,001\,101_2$ _____ (decimal)

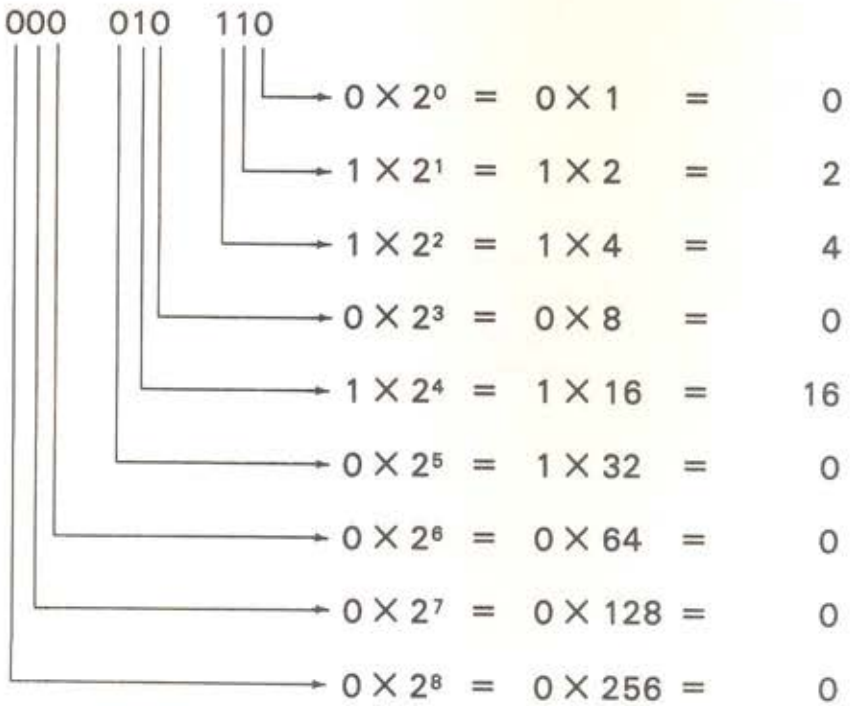
   j. $111\,111\,111_2$ _____ (decimal)

SOLUTIONS

1. Convert the following *binary* numbers to their *decimal* equivalents.

a. $000\ 010\ 110_2$ 22 (decimal)

NOTE
Here's how we solved the first problem.



b. $010\ 100\ 011_2$ 163 (decimal)

c. $101\ 000\ 000_2$ 320 (decimal)

d. $111\ 010\ 010_2$ 466 (decimal)

e. $001\ 111\ 110_2$ 126 (decimal)

f. $100\ 011\ 010_2$ 282 (decimal)

g. $011\ 011\ 011_2$ 219 (decimal)

h. $111\ 000\ 111_2$ 455 (decimal)

i. $010\ 001\ 101_2$ 141 (decimal)

j. $111\ 111\ 111_2$ 511 (decimal)

EXERCISES

2. Convert the following *octal* numbers to their *decimal* equivalents:

a. 527_8 _____ (decimal)

b. 136_8 _____ (decimal)

c. 742_8 _____ (decimal)

d. 433_8 _____ (decimal)

e. 277_8 _____ (decimal)

f. 4400_8 _____ (decimal)

g. 1000_8 _____ (decimal)

h. 4577_8 _____ (decimal)

i. 7777_8 _____ (decimal)

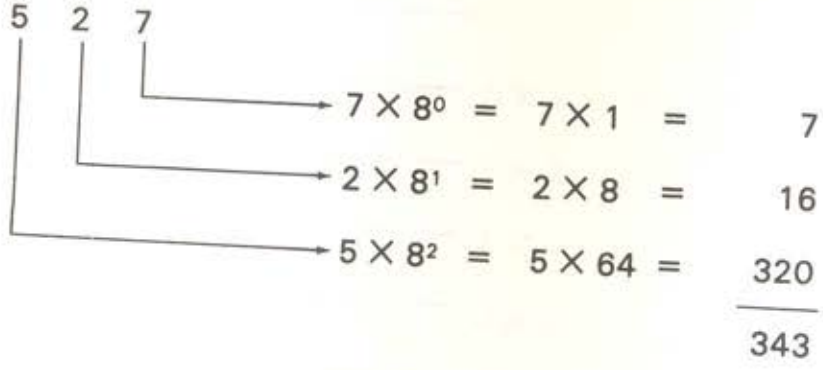
j. 1061_8 _____ (decimal)

SOLUTIONS

2. Convert the following *octal* numbers to their *decimal* equivalents:

a. 527_8 — 343 (decimal)

NOTE
Here's how we solved the first problem.



b. 136_8 — 94 (decimal)

c. 742_8 — 482 (decimal)

d. 433_8 — 283 (decimal)

e. 277_8 — 191 (decimal)

f. 4400_8 — 2304 (decimal)

g. 1000_8 — 512 (decimal)

h. 4577_8 — 2431 (decimal)

i. 7777_8 — 4095 (decimal)

j. 1061_8 — 561 (decimal)

EXERCISES

3. Convert the following *decimal* numbers to their *binary* equivalents:

a. 58_{10} _____ (binary)

b. 36_{10} _____ (binary)

c. 18_{10} _____ (binary)

d. 796_{10} _____ (binary)

e. 100_{10} _____ (binary)

f. 117_{10} _____ (binary)

g. 501_{10} _____ (binary)

h. 907_{10} _____ (binary)

i. 511_{10} _____ (binary)

j. 512_{10} _____ (binary)

SOLUTIONS

3. Convert the following *decimal* numbers to their *binary* equivalents:

a. 58_{10} — 111 010 (binary)

NOTE
Here's how we solved the first problem.

Base (Divisor)	Number	Quotient	Remainder
2	58	29	0 (LSD)
2	29	14	1
2	14	7	0
2	7	3	1
2	3	1	1
2	1	0	1 (MSD)

b. 36_{10} — 100 100 (binary)

c. 18_{10} — 010 010 (binary)

d. 796_{10} — 1 100 011 100 (binary)

e. 100_{10} — 0 001 100 100 (binary)

f. 117_{10} — 0 001 110 101 (binary)

g. 501_{10} — 0 111 110 101 (binary)

h. 907_{10} — 1 110 001 011 (binary)

i. 511_{10} — 0 111 111 111 (binary)

j. 512_{10} — 1 000 000 000 (binary)

EXERCISES

4. Convert the following *decimal* numbers to their *octal* equivalents:

a. 103_{10} _____ (octal)

b. 25_{10} _____ (octal)

c. 67_{10} _____ (octal)

d. 37_{10} _____ (octal)

e. 580_{10} _____ (octal)

f. 321_{10} _____ (octal)

g. 185_{10} _____ (octal)

h. 512_{10} _____ (octal)

i. 561_{10} _____ (octal)

j. 482_{10} _____ (octal)

SOLUTIONS

4. Convert the following *decimal* numbers to their *octal* equivalents:

a. 103_{10} 147 (octal)

NOTE
Here's how we solved the first problem.

Base (divisor)	Number	Quotient	Remainder
8	103	12	7 (LSD)
8	12	1	4
8	1	0	1 (MSD)

b. 25_{10} 31 (octal)

c. 67_{10} 103 (octal)

d. 37_{10} 45 (octal)

e. 580_{10} 1104 (octal)

f. 321_{10} 501 (octal)

g. 185_{10} 271 (octal)

h. 512_{10} 1000 (octal)

i. 561_{10} 1061 (octal)

j. 482_{10} 742 (octal)

EXERCISES

5. Convert the following *binary* numbers to their *octal* equivalents:

a. $101\ 111\ 101_2$ _____ (octal)

b. $110\ 110\ 111_2$ _____ (octal)

c. $010\ 101\ 101_2$ _____ (octal)

d. $011\ 110\ 100\ 001_2$ _____ (octal)

e. $111\ 111\ 111\ 111_2$ _____ (octal)

f. $010\ 110\ 101\ 100_2$ _____ (octal)

g. $001\ 101\ 011\ 101\ 111_2$ _____ (octal)

h. $010\ 110\ 000\ 011\ 001_2$ _____ (octal)

i. $010\ 100\ 011\ 101\ 111\ 011_2$ _____ (octal)

j. $110\ 110\ 100\ 101\ 010\ 111_2$ _____ (octal)

SOLUTIONS

5. Convert the following *binary* numbers to their *octal* equivalents:

a. $101\ 111\ 101_2$ — 575 (octal)

NOTE
Here's how we solved the first problem.

Memory Aid → 4 2 1 4 2 1 4 2 1

Binary Number → 1 0 1 | 1 1 1 | 1 0 1

Octal Number → 5 7 5

b. $110\ 110\ 111_2$ — 667 (octal)

c. $010\ 101\ 101_2$ — 255 (octal)

d. $011\ 110\ 100\ 001_2$ — 3641 (octal)

e. $111\ 111\ 111\ 111_2$ — 7777 (octal)

f. $010\ 110\ 101\ 100_2$ — 2654 (octal)

g. $001\ 101\ 011\ 101\ 111_2$ — 15357 (octal)

h. $010\ 110\ 000\ 011\ 001_2$ — 26031 (octal)

i. $010\ 100\ 011\ 101\ 111\ 011_2$ — 243573 (octal)

j. $110\ 110\ 100\ 101\ 010\ 111_2$ — 664527 (octal)

EXERCISES

6. Convert the following *octal* numbers to their *binary* equivalents:

a. 354_8 _____ (binary)

b. 207_8 _____ (binary)

c. 736_8 _____ (binary)

d. 2635_8 _____ (binary)

e. 1421_8 _____ (binary)

f. 6374_8 _____ (binary)

g. 27765_8 _____ (binary)

h. 71342_8 _____ (binary)

i. 665573_8 _____ (binary)

j. 124637_8 _____ (binary)

SOLUTIONS

6. Convert the following *octal* numbers to their *binary* equivalents:

a. 354_8 — 011 101 100 (binary)

NOTE
Here's how we solved the first problem.

Octal Value → 3 5 4

Memory Aid → (4 2 1) (4 2 1) (4 2 1)

Binary Value → 0 1 1 1 0 1 1 0 0

b. 207_8 — 010 000 111 (binary)

c. 736_8 — 111 011 110 (binary)

d. 2635_8 — 010 110 011 101 (binary)

e. 1421_8 — 001 100 010 001 (binary)

f. 6374_8 — 110 011 111 100 (binary)

g. 27765_8 — 010 111 111 110 101 (binary)

h. 71342_8 — 111 001 011 100 010 (binary)

i. 665573_8 — 110 110 101 101 111 011 (binary)

j. 124637_8 — 001 010 100 110 011 111 (binary)

Character Codes

OBJECTIVES

1. Given the term "ASCII CODE" and several definitions of purpose for it, be able to select the one correct definition.

2. Given five examples of even parity character code, be able to select the one example that is incorrect.

SAMPLE TEST ITEMS

1. The purpose of a code such as ASCII is that it _____. (Circle one answer only.)

   a. Allows analog machines such as analog computers to represent character information that people are trained to use.

   b. Sets standards for training computer operators in various computer languages.

   c. Allows binary machines such as digital computers to represent character information that people are trained to use.

   d. Two of the above are correct.

   e. None of the above is correct.

2. Assuming even parity, only one of the following characters contains an error. (Circle the correct answer.)

   a. 11011110

   b. 1101001

   c. 01011010

   d. 11010100

   e. 11011100

Mark your place in this workbook and view Lesson 3 in the A/V program, "Number Systems and Codes."

People are taught to communicate using letters of the alphabet, punctuation marks, decimal numbers, and special symbols such as the dollar sign. However, a digital computer can process only binary data. Therefore, any information used must be coded in binary format prior to computer processing.

Codes

A binary-formatted code can represent different symbols only when sufficient binary elements are employed for each symbol. If we use only a single binary digit (or "bit") to represent each symbol, we limit ourselves to two choices: one symbol represented by the "ON" state, the other represented by the "OFF" state. With such an arrangement, we could let the ON, or one-state, represent "no," and the OFF, or zero-state, represent "yes." While it would be difficult, such an arrangement would allow messages of a very limited nature to be conveyed between two remote stations.

If instead of using one binary digit for each character we use two, we have more characters to choose from. We have two choices for a one-bit code: 0 or 1. We have four choices for a two-bit code: 00, 01, 10, or 11. Thus, if we choose a three-bit code, we have a choice of eight: 000, 001, 010, 011, 100, 101, 110, and 111. It can be shown that, for a code with n-bit characters, the number of characters available will be 2^n.

Although we could assign any arbitrary meaning to a code character, in most cases it is more practical to let the characters represent letters of the alphabet, numbers, punctuation marks, and spaces. ASCII is discussed in the following pages. Other codes are included in Appendix B.

ASCII Code – The American Standard Code for Information Interchange (ASCII) is a 7-bit code that has been established by the computer manufacturing, data processing, and I/O device manufacturing industries.

Many input/output devices on the market today are designed to conform to ASCII format. Some manufacturers, however, use minor variations of the ASCII code to make it more applicable to special-purpose devices. (A chart of a version of ASCII code is given in Appendix B.) One variation of the ASCII code is the Data Interchange Code. The primary difference between this code and ASCII is that some printing characters are replaced by non-printing control characters. The Data Interchange Code is readily adaptable to computer-to-computer communication.

Parity

Parity checking is a common technique used for detecting data errors where bits have changed value due to unwanted electrical interference or equipment malfunction. Parity checking relies on the inclusion in the data of extra bits called *parity bits*. Let's see how parity is used to detect errors.

The 7-bit ASCII code for the letter R is 1010010 and, for the letter S, is 1010011. If *even* parity is used on a computer system, *eight* bits will be used to represent these characters. The extra bit, a parity bit, will be added to the left end in the positions shown below:

R = ? 1010010

S = ? 1010011

Even parity means that the parity bit *either a 1 or a 0,* is chosen so that the total number of 1s in the 8-bit character will be even. Therefore, in the case of the letter R, the parity bit will be a 1, and in the case of the letter S, the parity bit will be a 0.

R = 11010010

S = 01010011

In a similar fashion, the codes for *all* characters in a system designed with even parity have been generated with an even number of 1s.

In even parity systems, parity *checking* is the process of examining each code the computer is processing to verify that it contains an even number of 1s. If the number of 1s is *not* even, then a data error has occurred in the system, and the character that is being verified is not valid. The figure below shows two valid and two invalid characters on a system using even parity.

11010010	valid
01010011	valid
11110010	invalid
01010010	invalid

Once a character has been identified as invalid, computers typically output error messages to an operator or try to obtain another copy of the character.

NOTE

We have just described even parity, in which the number of 1s in each character is supposed to be even. Some systems are designed with odd parity. In odd parity systems, the number of 1s in each character is supposed to be odd.

Observe that the simple-bit type of parity we have just described does not allow *all* errors to be detected. For example, in an even parity system, if *two* bits are changed in the character R the resulting character still has an even number of 1s. That is,

R =	11010010	four 1s
	11011110	six 1s

In both even parity and odd parity systems, any electrical interference or equipment malfunction that causes an *even* number of bits to be changed will *not* be detected. Only changes of *odd* numbers of bits will be detected. More complex error-detection methods use *more than one bit* to detect data errors. With these methods, fewer errors go undetected. *No method is perfect.*

Now do the following exercises on character codes and parity before going on to the unit test.

EXERCISES

1. Write a brief explanation of the purpose of codes such as ASCII.

2. Fill in the codes for the symbols below using the table in Appendix B. Then fill in the parity bit using EVEN parity. (Note: the table in Appendix B has the parity bit position always equal to 1. This is common in systems that do not use parity checking.)

Symbol	Parity	Binary Code
R		
S		
Q		
space		
Z		
$		

SOLUTIONS

1. The purpose of codes is to allow binary machines such as digital computers to represent character information that people are trained to use.

2. Fill in the codes for the symbols below using the table in Appendix B. Then fill in the parity bit using EVEN parity. (Note: the table in Appendix B has the parity bit position always equal to 1. This is common in systems that do not use parity checking.)

Symbol	Parity	Binary Code
R	1	1 0 1 0 0 1 0
S	0	1 0 1 0 0 1 1
Q	1	1 0 1 0 0 0 1
space	1	0 1 0 0 0 0 0
Z	0	1 0 1 1 0 1 0
$	0	0 1 0 0 1 0 0

EXERCISES

3. Assuming a parity bit and no data errors in the following character codes, which character code is used on a system with even parity?

   a. 11001011

   b. 11010110

   c. 11000001

   d. 11001001

   e. 01010100

4. Assuming even parity, which of the following characters contains a data error?

   a. 11011010

   b. 11010001

   c. 01011010

   d. 11010100

   e. 11011101

SOLUTIONS

3. Assuming a parity bit and no data errors in the following character codes, which character code is used on a system with even parity?

   a. 11001011

   b. 11010110

   c. 11000001

   d. 11001001

   e. 01010100

4. Assuming even parity, which of the following characters contains a data error?

   a. 11011010

   b. 11010001

   c. 01011010

   d. 11010100

   e. 11011101

Take the test for this module and evaluate your answers before studying another module.

Appendix A
The Hexadecimal Number System

Some computer manufacturers and programmers use the hexadecimal number system. The hexadecimal number system, sometimes referred to as "hex," has a base of *sixteen*. A base of sixteen simply means that the system uses sixteen digits. The symbols that commonly represent these digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Below is a comparison of counting in decimal and hexadecimal (hex).

Decimal	Hex	Decimal	Hex	Decimal	Hex
1	1	18	12	35	23
2	2	19	13	.	.
3	3	20	14	.	.
4	4	21	15	500	1F4
5	5	22	16	501	1F5
6	6	23	17	502	1F6
7	7	24	18	503	1F7
8	8	25	19	504	1F8
9	9	26	1A	505	1F9
10	A	27	1B	506	1FA
11	B	28	1C	507	1FB
12	C	29	1D	508	1FC
13	D	30	1E	509	1FD
14	E	31	1F	510	1FE
15	F	32	20	511	1FF
16	10	33	21	512	200
17	11	34	22	513	201

Appendix B
Some Character Codes

One Variation of the ASCII Code

Figure 1 summarizes the ASCII code and its binary and octal forms as used in the Model 33 ASR/KSR Teletype.

Model 33 ASR/KSR Teletype Code (ASCII) in Binary Form

Note: Bit 8 may be parity or always punched. Always-punched is shown in this table.

Figure 1 One Variation of ASCII Code

Baudot Code

The Baudot code is a 5-bit code used mainly for telegraphs and for some keyboards, printers, punches, and readers. Although 5 bits can accommodate only 32 unique codes, 2 of the codes are "figures" (FIGS) and "letters" (LTRS). Prefixing the FIGS or LTRS code before other bit combinations permits dual definition of the remaining codes. In other words, each bit combination could represent either a particular letter, or a particular number or special character, depending on whether a LTRS character or a FIGS character comes before it. This means that when a Baudot input/output device is interfaced to a computer, the computer must maintain proper FIGS/LTRS status in order to interpret the necessary information properly. This code is rarely used with computers. Unlike ASCII code, the Baudot code has not enough bits in each character to make unique code assignments for both alphanumeric and control characters. This is the reason for the use of a letters-figures status with Baudot.

Binary-Coded Decimal (BCD)

The Binary-Coded Decimal code is merely a compression of the Hollerith code. Where the Hollerith code is a fixed ratio code, the BCD code compresses this 12-bit code into a 6-bit code: 2 binary bits replace 3-zone bits, and 4 binary bits replace the 9-data bit.

Extended Binary-Coded Decimal Interchange Code (EBCDIC)

The EBCDIC code is merely the BCD code extended to 8 binary bits. The extra bits provide the capacity for additional control characters that are used in graphic representations and other special applications. The character capability is very similar to ASCII's.

CHARACTER		BIT POSITION				
LOWER CASE	UPPER CASE	1	2	3	4	5
A	-	●	●			
B	?	●			●	●
C	:		●	●	●	
D	$	●			●	
E	3	●				
F	!	●		●	●	
G	8		●		●	●
H	*			●		●
I	8		●	●		
J	'	●	●		●	
K	(	●	●	●	●	
L	)		●			●
M	.			●	●	●
N	,			●	●	
O	9				●	●
P	0		●	●		●
Q	1	●	●	●		●
R	4		●		●	
S	BELL	●		●		
T	5					●
U	7	●	●	●		
V	;		●	●	●	●
W	2	●	●			●
X	/	●		●	●	●
Y	6	●		●		●
Z	"	●				●
LETTERS (SHIFT TO LOWER CASE)		●	●	●	●	●
FIGURES (SHIFT TO UPPER CASE)		●	●		●	●
SPACE				●		
CARRIAGE RETURN					●	
LINE FEED			●			
BLANK						

PRESENCE OF ● INDICATES MARK BIT
ABSENCE OF ● INDICATES SPACE BIT

Figure 2 Baudot Code Format

Four-of-Eight Code

The Four-of-Eight code is a so-called fixed-ratio code that represents information with a fixed number of one bits and fixed number of zero bits. In the Four-of-Eight code, four bits are one and four bits are zero in every character. The difference is in the bit arrangement. Although this code is much less efficient than ASCII, or even Baudot, it is ideal in applications where high accuracy, and easy error detection and correction are required. A typical application is credit card verification.

Hollerith Code

The Hollerith code is used almost exclusively for punched-card applications. It is a 12-bit code, designed to represent the alphabet plus digits 1 through 9. The most commonly used cards are divided horizontally into columns, and vertically into rows. In this code each character has two parts called zone bits and data bits. The top three rows are used as zone bits, and the remaining nine rows as data bits. Each column can be used to record one character of data, either an alphabetic, numeric, or a special symbol. Numeric digits are represented by a single hole in the appropriate row, while alphabetic characters are indicated by a hole punched in both the zone area and the data area of the same row. Special characters may be represented by one punch in the zone area, and two punches in the data area. There are 3 zone bits and 9 data bits. As with the Four-of-Eight code, the Hollerith code lends itself to easy error detection.

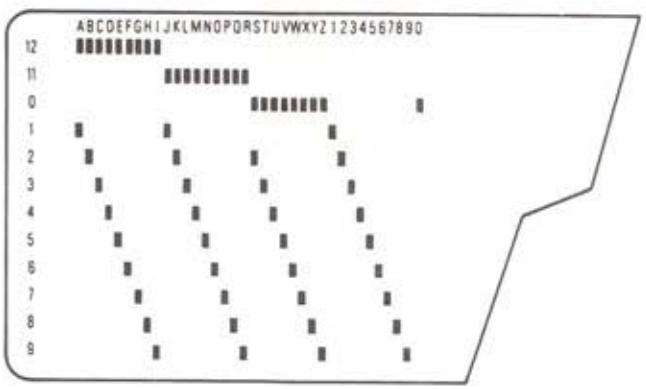


Figure 3 Hollerith Code Format