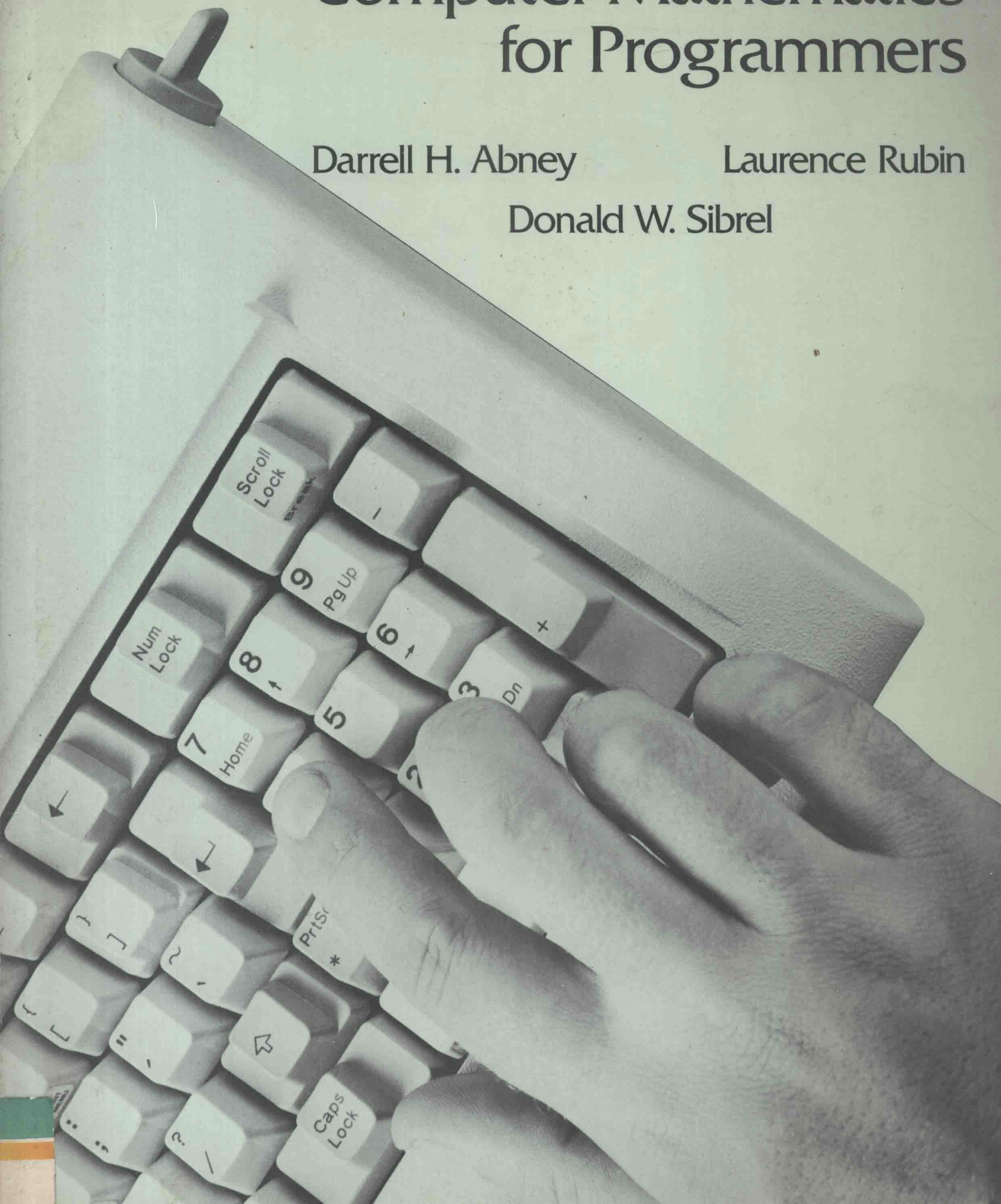
Computer Mathematics for Programmers

Darrell H. Abney
Laurence Rubin
Donald W. Sibrel



COMPUTER MATHEMATICS FOR PROGRAMMERS

Darrell H. Abney

Laurence Rubin

Donald W. Sibrel

Nashville State Technical Institute

Academic Press, Inc.
(Harcourt Brace Jovanovich, Publishers)

Orlando San Diego San Francisco New York
London Toronto Montreal Sydney Tokyo São Paulo

Cover photo and chapter-opening photos by Carroll Morgan.

Academic Press, Inc.
Orlando, Florida 32887

United Kingdom Edition published by
Academic Press, Inc. (London) Ltd.
24/28 Oval Road, London NW1 7DX

ISBN: 0-12-042150-X
Library of Congress Catalog Card Number: 84-70635

Printed in the United States of America

COMPUTER MATHEMATICS FOR PROGRAMMERS

Preface

This book was written primarily for business and data processing students, but it can also be used by any student who has a need for selected topics in finite mathematics.

A knowledge of elementary algebra would be very helpful for students using this text. Students with stronger algebra backgrounds can skip Chapter 5 without loss of continuity. Students taking a separate flowcharting or pseudocode course could omit Chapter 4.

A quarter-length course could include Chapters 1, 2, 3, 6, 7, 9, and 10. Chapters 4, 5, 8, and 11 could be added for semester-length courses or if your course needs more material. For students with little background in algebra, Chapter 5 should be included. If your students have a good background in algebra, more chapters can be covered than mentioned above.

The authors feel that the topics covered are general enough to serve the needs of a variety of students. The book was originally designed for data processing students at Nashville State Technical Institute and, as a consequence, it is weighted toward that curriculum. The chapters on Algebra, Boolean Algebra, Flowcharting, Computer Arithmetic, and Statistics were added as a result of suggestions solicited from other schools.

The authors have designed the text with student readability in mind, with an emphasis on complete step-by-step examples throughout the book. The text features an abundance of exercises and practice exams for each chapter. Technical jargon and formal proofs have been de-emphasized throughout the book.

ACKNOWLEDGMENTS

The authors would like to express their appreciation to all the personnel at Academic Press, who have given us invaluable assistance on this project. We are especially indebted to Dale Brown, the Computer Science Editor, who personally guided this project to its completion.

We would also like to express our thanks to the faculty and students of the Mathematics Department at Nashville State Technical Institute who field-tested many of the chapters and gave their ideas for improvement of the text.

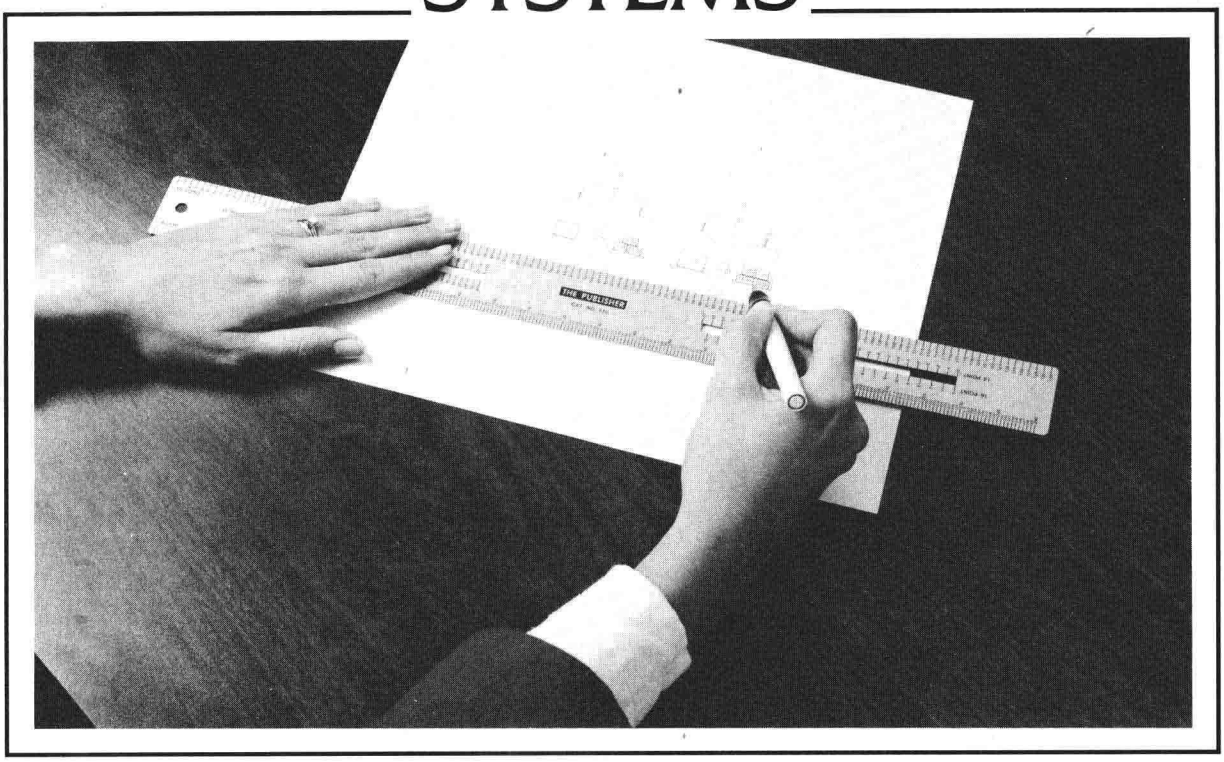
In addition, we would like to thank the following reviewers who spent many hours reading the manuscript and making numerous suggestions: Jan Buzydlowski, Community College of Philadelphia; Robert W. Hamilton, Amarillo College; Marsha Hardacre, Black Hawk Community College; Margie Hobbs, State Technical Institute at Memphis; Robert W. Lacey; Robert Ludwig, Rockland Community College; Janet Matthes, Southwest Wisconsin Vocational Technical Institute; Kianpour Mihankhah, Henderson Community College; Wesley E. Nance, Cerritos College; Anita J. Proffitt, Indiana University–Purdue University–Indianapolis; Greg Schaefer, Southwest Wisconsin Vocational Technical Institute; Kenneth W. Thompson, Florida Junior College at Jacksonville.

Contents

APPENDICES

1

NUMBER SYSTEMS



1.0 Introduction

We all use numbers. If we buy a list of items at a store, numbers are used to indicate the quantity of each item as well as the total amount we have to pay. Numbers can represent almost anything: age, money, weight, height, social security identification, and so on.

In a computer, numbers can represent much more. Names, addresses, gender, and dates are all placed in a computer's memory as numbers. Most computers use a numbering system that differs from our familiar decimal system, because it is more efficient and lends itself to the electronics of the computer. For us to understand computers and computer operations we need to learn these computer number systems. With practice, you will learn how to count in these systems as well as how to convert from one to the other.

The number systems we are about to study are all *positional* in nature. This means that the position of a digit indicates the value of that digit. In the number 7252 the last 2 indicates 2 units and the first 2 means 200—quite a difference! There are systems that are not positional in nature. Historically, many civilizations used other types of systems to count and represent numbers. Here are two examples:

Roman:

1	5	10	50	100	500	1000	
I	V	X	L	C	D	M	(Symbols used by Romans)

In the Roman system, the number 17 is represented by XVII, and 1562 is MDLXII. An addition problem looks like:

$$\begin{array}{rr} & \text{DCCLXVII} \quad (767) \\ + & \text{CCXXI} \quad (221) \\ \hline & \text{DCCCCLXXXVIII} \quad (988) \end{array}$$

Can you imagine using this system in our modern society?

Traditional Chinese-Japanese:

1	2	3	4	5	6	7	8	9
一	二	三	四	五	六	七	八	九

10	10^2	10^3
十	百	千

So 526 is

五百二十六

Notice that every power of 10 has to be explicitly written, whereas in our decimal system the position of the digit in the number implicitly tells us the power of 10.

1.1 Base Ten, or the Decimal System

You may recall that the number 709 means 7 one hundreds plus 0 tens plus 9 ones:

$$709 = 7(100) + 0(10) + 9(1)$$

You probably also know that 100 can be written as 10^2, 10 as 10^1, and 1 as 10^0, so that we can write 709 as follows:

$$709 = 7(100) + 0(10) + 9(1) = 7(10^2) + 0(10^1) + 9(10^0)$$

This last form of 709 is called the *expanded notation* for a decimal number. We shall see shortly that other number systems have expanded notations using bases other than 10. Here are a few examples of expanded decimal numbers:

Example 1 $4675 = 4(1000) + 6(100) + 7(10) + 5(1)$
$= 4(10^3) + 6(10^2) + 7(10^1) + 5(10^0).$

Example 2 $44 = 4(10) + 4(1) = 4(10^1) + 4(10^0).$

Example 3 $50780 = 5(10000) + 0(1000) + 7(100) + 8(10) + 0(1)$
$= 5(10^4) + 0(10^3) + 7(10^2) + 8(10^1) + 0(10^0).$

Our decimal system and the other systems discussed in this chapter are positional in nature. The position of each digit, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, indicates the power of 10 that is to multiply that integer (integers being positive and negative whole numbers and zero). In the number 4675 the 6 is the integer that the second power of 10 is multiplied by, etc. Also note that in the decimal system (base 10) only the digits 0 through 9 are used. Later we will see that in the octal system we use only the integers 0 through 7 as the multipliers of powers of 8.

1.2 Base Two, or the Binary System

The French mathematician John Napier (1550–1617) expressed the idea that any weight, in whole pounds, placed on one side of a scale could be counterbalanced by a set of weights consisting of one each of 1 lb, 2 lb, 4 lb, 8 lb, and so forth. (See Figure 1.1.) The consequences of this observation are now seen in the way numbers are stored in modern-day computers as well as how they do arithmetic operations. This observation eventually led to the development of the *binary number system*. Figure 1.2 shows how the scale would balance 23 lb.

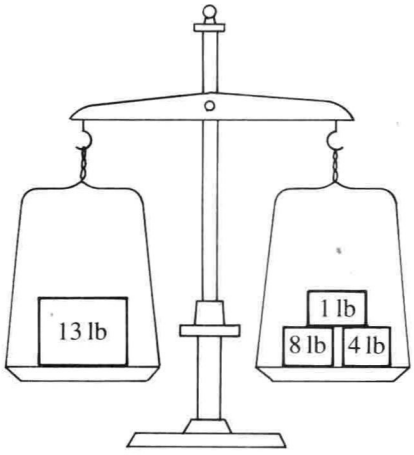


Figure 1.1

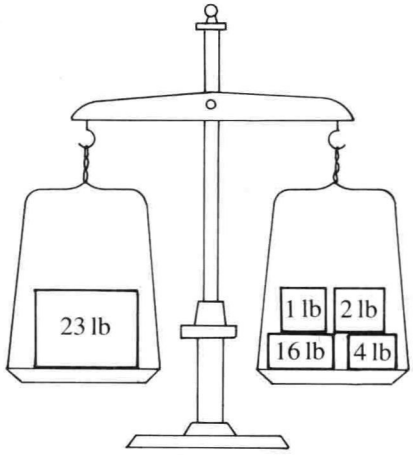


Figure 1.2

The binary system is a way of expressing numbers using only 0's and 1's. The position of the 0 or 1 indicates what power of decimal two will be multiplied by the 0 or 1. Such a number would look like 10011 or 10 or 111000. In order not to confuse these numbers with base 10 numbers we will indicate that 10011 is a binary number by writing $(10011)_2$.

The expanded notation of $(10011)_2$ is similar to the expanded notation of a decimal number, except that the base 2 is used instead of base 10. Therefore

$$(10011)_2 = \underline{1}(2^4) + \underline{0}(2^3) + \underline{0}(2^2) + \underline{1}(2^1) + \underline{1}(2^0) \qquad \text{(in decimal system)}$$

Note that the underlined integers are the digits of the given number. The following examples will illustrate how to change a given binary number into its equivalent decimal number.

Example 4 Change $(10011)_2$ into an equivalent decimal number.

SOLUTION: $(10011)_2 = 1(2^4) + 0(2^3) + 0(2^2) + 1(2^1) + 1(2^0)$ (in decimal system)
$= 1(16) + 0(8) + 0(4) + 1(2) + 1(1)$ $(2^0 = 1)$
$= 16 + 0 + 0 + 2 + 1$
$= 19$
So, $(10011)_2 = (19)_{10}$.

Example 5 Change $(10)_2$ into an equivalent decimal number.

SOLUTION: $(10)_2 = 1(2^1) + 0(2^0)$
$= 1(2) + 0$
$= 2$
So, $(10)_2 = (2)_{10}$.

Example 6 Change $(111011)_2$ into an equivalent decimal number.

SOLUTION: $(111011)_2 = 1(2^5) + 1(2^4) + 1(2^3) + 0(2^2) + 1(2^1) + 1(2^0)$
$= 1(32) + 1(16) + 1(8) + 0(4) + 1(2) + 1(1)$
$= 32 + 16 + 8 + 0 + 2 + 1$
$= 59$
So, $(111011)_2 = (59)_{10}$.

The following visualization of a binary number may be helpful:

Binary	1 0 1 1	Note that we either use 8, 4, 2, or 1 or we don't, depending on whether or not we have a 1 or a 0. In this case we use 8, don't use 4, use 2, and use 1 for a sum of 11 in the decimal system.
	↓ ↓ ↓ ↓	
Decimal	2^3 2^2 2^1 2^0	
	↓ ↓ ↓ ↓	
Decimal	8 4 2 1	

1.3 Base Eight, or the Octal System

The octal system is another positional system which uses the digits 0, 1, 2, 3, 4, 5, 6, and 7 together with powers of 8 to represent a number. We denote an octal number such as 2675 as $(2675)_8$. The 5 in this octal number stands for the number of 8^0's or 1's present in the number, the 7 is the number of 8^1's present, the 6 is the number of 8^2's we have, and finally the 2 is the number of 8^3's.

Example 7 Convert $(2675)_8$ into an equivalent decimal number.

SOLUTION: $(2675)_8 = 2(8^3) + 6(8^2) + 7(8^1) + 5(8^0)$ (in decimal system)
$= 2(512) + 6(64) + 7(8) + 5(1)$
$= 1024 + 384 + 56 + 5$
$= (1469)_{10}$

Example 8 Convert $(5075)_8$ into an equivalent decimal number.

SOLUTION: $(5075)_8 = 5(8^3) + 0(8^2) + 7(8^1) + 5(8^0)$ (in decimal system)
$= 5(512) + 0(64) + 7(8) + 5(1)$
$= (2621)_{10}$

1.4 Base Sixteen, or the Hexadecimal System

The hexadecimal system uses the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Since our base is now going to be 16, we need integers 0 through 15. But 15 is a decimal number and would mean something entirely different in this base, so we use the letters A, B, C, D, E, and F to denote 10, 11, 12, 13, 14, and 15. Thus, 5ADF is a perfectly legitimate hexadecimal number. The 5 in this number is in the 16^3 position, the A is in the 16^2 position and means that we have A or 10 16^2's. This may seem confusing at first, but it is merely an extension of the way our decimal system works. The number 325 can also be a hexadecimal number, so we write $(325)_{16}$. The expanded notation of a hexadecimal number gives us its decimal equivalent.

Example 9 Convert $(ADB)_{16}$ into an equivalent decimal number.

SOLUTION:
$$\begin{aligned}(ADB)_{16} &= A(16^2) + D(16^1) + B(16^0) \quad \text{(in decimal system)}\\ &= 10(256) + 13(16) + 11(1)\\ &= 2560 + 208 + 11\\ &= (2779)_{10}\end{aligned}$$

Example 10 Convert $(3C29)_{16}$ into an equivalent decimal number.

SOLUTION:
$$\begin{aligned}(3C29)_{16} &= 3(16^3) + C(16^2) + 2(16^1) + 9(16^0)\\ &= 3(4096) + 12(256) + 2(16) + 9(1)\\ &= 12288 + 3072 + 32 + 9\\ &= (15401)_{10}\end{aligned}$$

Example 11 Convert $(12)_{16}$ into an equivalent decimal number.

SOLUTION:
$$\begin{aligned}(12)_{16} &= 1(16^1) + 2(16^0)\\ &= 1(16) + 2(1)\\ &= 16 + 2\\ &= (18)_{10}\end{aligned}$$

For convenience we list the various powers of 2, 8, and 16:

	Base		
Power	*2*	*8*	*16*
0	1	1	1
1	2	8	16
2	4	64	256
3	8	512	4096
4	16	4096	65536
5	32	32768	1048576
6	64	262144	
7	128		
8	256		

In the following sections we will be dividing numbers by 2, 8, and 16 and recording remainders as whole numbers. If you are using a hand-held calculator, there is an algorithm (step-by-step procedure) to determine this remainder. This algorithm will come in very handy when you are asked to divide many numbers by 16 or 8. Let's look at the problem of determining the remainder of 6542 divided by

16 on a calculator. On most calculators using algebraic notation you can use the following algorithm:

1. Key in number.
2. Divide by divisor.
3. Look at the answer (in decimal form).
4. Subtract whole part from answer.
5. Multiply decimal part by divisor.
6. Read remainder on display (whole number).

Here are the steps on a calculator for our suggested problem of $6542 \div 16$.

Key	*Display*
6542	6542
÷	6542
16	16
=	408.875000
−	408.875000
408	408
=	.875000
×	.875000
16	16
=	14

The remainder after dividing 6542 by 16 is therefore 14. The algorithm is the same for any positive divisor. Here are the key steps on a calculator for the problem $877 \div 8$.

Key	*Display*
877	877
÷	877
8	8
=	109.625000
−	109.625000
109	109
=	.625000
×	.625000
8	8
=	5

The remainder after dividing 877 by 8 is therefore 5.

Finally, here are the key steps for dividing a number such as 67 by 2:

Key	*Display*
67	67
÷	67
2	2
=	33.500000

We can stop here, as the remainder after division by 2 is either 0 or 1. It is 0 if there is no decimal portion of the quotient; otherwise the remainder is 1. So the remainder in this problem is 1.

1.5 Converting Decimal Numbers into Binary Numbers

In general when we are converting decimal numbers into numbers using other bases we will use a process of repeated divisions and keep a record of the remainders. To convert a decimal number into a binary number we use the following procedure:

Example 12 Convert $(25)_{10}$ into a binary number.

SOLUTION:

Step 1: Divide 25 by the base 2 which gives 12 with a remainder of 1. 1 is therefore the units position of the binary number.

Step 2: Divide 12 (the quotient obtained in step 1) by 2 which gives 6 with a remainder of 0. 0 is therefore the 2's position of the binary number.

Step 3: Divide 6 (the quotient obtained in step 2) by 2 which gives 3 with a remainder of 0 which is the 2^2 position.

Step 4: Divide 3 (the quotient obtained in step 3) by 2 which gives 1 with a remainder of 1 which is the 2^3 position.

Step 5: Divide 1 (the quotient obtained in step 4) by 2 which gives 0 with a remainder of 1 which is the 2^4 position.

When the quotient reaches 0, we end our division process, making sure to record the last remainder. If we now look at all the remainders in reverse order, we can write the binary number which is equivalent to the decimal number 25.

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

To check this answer, we will expand $(11001)_2$:

$$\begin{aligned}(11001)_2 &= 1(2^4)+1(2^3)+0(2^2)+0(2^1)+1(2^0)\\ &= 16 + 8 + 0 + 0 + 1\\ &= (25)_{10}\end{aligned}$$

To organize the above steps, we usually proceed as follows:

	Remainders
2)25	1
2)12	0
2)6	0
2)3	1
2)1	1
0	

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

Example 13 Convert $(14)_{10}$ into a binary number.

SOLUTION:

2)14	0
2)7	1
2)3	1
2)1	1
0	

$(14)_{10}=(1110)_2$

Example 14 Convert $(33)_{10}$ into a binary number.

SOLUTION:

2)33	1
2)16	0
2)8	0
2)4	0
2)2	0
2)1	1
0	

$(33)_{10}=(100001)_2$

1.6 Converting Decimal Numbers into Octal Numbers

Octal numbers which are equivalent to a given decimal number are obtained by the same process used in Example 12, except we now divide by 8 instead of 2. The remainders can be any digit between 0 and 7.

Example 15 Convert $(574)_{10}$ into an octal number.

SOLUTION:

8)574	6
8)71	7
8)8	0
8)1	1
0	

$(574)_{10}=(1076)_8$

Once again, we can check our answer by expanding $(1076)_8$ as follows:

$$\begin{aligned}(1076)_8 &= 1(8^3)+0(8^2)+7(8^1)+6(8^0)\\ &= 512 + 0 + 56 + 6\\ &= (574)_{10}.\end{aligned}$$

Example 16 Convert $(20)_{10}$ into an octal number.

SOLUTION:

8)20	4
8)2	2
0	

$(20)_{10}=(24)_8$

Example 17 Convert $(999)_{10}$ into an octal number.