

GLOBAL
EDITION



Digital Fundamentals

ELEVENTH EDITION

Thomas L. Floyd

ALWAYS LEARNING

PEARSON

Eleventh Edition | Global Edition



Digital Fundamentals

Thomas L. Floyd

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Product Manager: Lindsey Prudhomme Gill
Program Manager: Maren Beckman
Project Manager: Rex Davidson
Editorial Assistant: Nancy Kesterson
Team Lead Program Manager: Laura Weaver
Team Lead Project Manager: JoEllen Gohr
Head of Learning Asset Acquisition, Global Editions: Laura Dent
Acquisitions Editor, Global Editions: Karthik Subramanian
Project Editor, Global Editions: K.K. Neelakantan
Senior Production Manufacturing Controller, Global Editions: Trudy Kimber
Director of Marketing: David Gesell
Senior Marketing Coordinator: Stacey Martinez
Senior Marketing Assistant: Les Roberts
Procurement Specialist: Deidra M. Skahill
Media Project Manager: Noelle Chun
Media Project Coordinator: April Cleland
Media Production Manager, Global Editions: Vikram Kumar
Creative Director: Andrea Nix
Art Director: Diane Y. Ernsberger
Cover Designer: Lumina Datamatics Ltd.
Cover Image: © echo3005/Shutterstock
Full-Service Project Management: Sherrill Redd/iEnergizer Aptara®, Inc.

Credits and acknowledgments for materials borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text.

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

© Pearson Education Limited 2015

The right of Thomas L. Floyd to be identified as the author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Digital Fundamentals, 11th edition, ISBN 978-0-13-273796-8, by Thomas L. Floyd, published by Pearson Education © 2015.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-07598-8

ISBN 13: 978-1-292-07598-3

Typeset by Aptara®, Inc. in Times Roman.

Printed and bound by Courier Kendallville in The United States of America.

CONTENTS

CHAPTER 1 **Introductory Concepts** 15

- 1-1 Digital and Analog Quantities 16
- 1-2 Binary Digits, Logic Levels, and Digital Waveforms 19
- 1-3 Basic Logic Functions 25
- 1-4 Combinational and Sequential Logic Functions 27
- 1-5 Introduction to Programmable Logic 34
- 1-6 Fixed-Function Logic Devices 40
- 1-7 Test and Measurement Instruments 43
- 1-8 Introduction to Troubleshooting 54

CHAPTER 2 **Number Systems, Operations, and Codes** 65

- 2-1 Decimal Numbers 66
- 2-2 Binary Numbers 67
- 2-3 Decimal-to-Binary Conversion 71
- 2-4 Binary Arithmetic 74
- 2-5 Complements of Binary Numbers 77
- 2-6 Signed Numbers 79
- 2-7 Arithmetic Operations with Signed Numbers 85
- 2-8 Hexadecimal Numbers 92
- 2-9 Octal Numbers 98
- 2-10 Binary Coded Decimal (BCD) 100
- 2-11 Digital Codes 104
- 2-12 Error Codes 109

CHAPTER 3 **Logic Gates** 125

- 3-1 The Inverter 126
- 3-2 The AND Gate 129
- 3-3 The OR Gate 136
- 3-4 The NAND Gate 140
- 3-5 The NOR Gate 145
- 3-6 The Exclusive-OR and Exclusive-NOR Gates 149
- 3-7 Programmable Logic 153
- 3-8 Fixed-Function Logic Gates 160
- 3-9 Troubleshooting 170

CHAPTER 4 **Boolean Algebra and Logic Simplification** 191

- 4-1 Boolean Operations and Expressions 192
- 4-2 Laws and Rules of Boolean Algebra 193
- 4-3 DeMorgan's Theorems 199

4-4	Boolean Analysis of Logic Circuits	203
4-5	Logic Simplification Using Boolean Algebra	205
4-6	Standard Forms of Boolean Expressions	209
4-7	Boolean Expressions and Truth Tables	216
4-8	The Karnaugh Map	219
4-9	Karnaugh Map SOP Minimization	222
4-10	Karnaugh Map POS Minimization	233
4-11	The Quine-McCluskey Method	237
4-12	Boolean Expressions with VHDL	240
	Applied Logic	244

CHAPTER 5 **Combinational Logic Analysis** 261

5-1	Basic Combinational Logic Circuits	262
5-2	Implementing Combinational Logic	267
5-3	The Universal Property of NAND and NOR gates	272
5-4	Combinational Logic Using NAND and NOR Gates	274
5-5	Pulse Waveform Operation	279
5-6	Combinational Logic with VHDL	283
5-7	Troubleshooting	288
	Applied Logic	294

CHAPTER 6 **Functions of Combinational Logic** 313

6-1	Half and Full Adders	314
6-2	Parallel Binary Adders	317
6-3	Ripple Carry and Look-Ahead Carry Adders	324
6-4	Comparators	327
6-5	Decoders	331
6-6	Encoders	341
6-7	Code Converters	345
6-8	Multiplexers (Data Selectors)	347
6-9	Demultiplexers	356
6-10	Parity Generators/Checkers	358
6-11	Troubleshooting	362
	Applied Logic	365

CHAPTER 7 **Latches, Flip-Flops, and Timers** 387

7-1	Latches	388
7-2	Flip-Flops	395
7-3	Flip-Flop Operating Characteristics	406
7-4	Flip-Flop Applications	409
7-5	One-Shots	414
7-6	The Astable Multivibrator	423
7-7	Troubleshooting	427
	Applied Logic	429

CHAPTER 8 **Shift Registers** 449

- 8-1 Shift Register Operations 450
- 8-2 Types of Shift Register Data I/Os 451
- 8-3 Bidirectional Shift Registers 462
- 8-4 Shift Register Counters 465
- 8-5 Shift Register Applications 469
- 8-6 Logic Symbols with Dependency Notation 476
- 8-7 Troubleshooting 478
- Applied Logic 480

CHAPTER 9 **Counters** 497

- 9-1 Finite State Machines 498
- 9-2 Asynchronous Counters 500
- 9-3 Synchronous Counters 507
- 9-4 Up/Down Synchronous Counters 515
- 9-5 Design of Synchronous Counters 519
- 9-6 Cascaded Counters 527
- 9-7 Counter Decoding 531
- 9-8 Counter Applications 534
- 9-9 Logic Symbols with Dependency Notation 539
- 9-10 Troubleshooting 541
- Applied Logic 545

CHAPTER 10 **Programmable Logic** 561

- 10-1 Simple Programmable Logic Devices (SPLDs) 562
- 10-2 Complex Programmable Logic Devices (CPLDs) 567
- 10-3 Macrocell Modes 574
- 10-4 Field-Programmable Gate Arrays (FPGAs) 577
- 10-5 Programmable Logic software 585
- 10-6 Boundary Scan Logic 595
- 10-7 Troubleshooting 602
- Applied Logic 608

CHAPTER 11 **Data Storage** 627

- 11-1 Semiconductor Memory Basics 628
- 11-2 The Random-Access Memory (RAM) 633
- 11-3 The Read-Only Memory (ROM) 646
- 11-4 Programmable ROMs 652
- 11-5 The Flash Memory 655
- 11-6 Memory Expansion 660
- 11-7 Special Types of Memories 666
- 11-8 Magnetic and Optical Storage 670
- 11-9 Memory Hierarchy 676
- 11-10 Cloud Storage 680
- 11-11 Troubleshooting 683

CHAPTER 12 Signal Conversion and Processing 697

- 12-1 Analog-to-Digital Conversion 698
- 12-2 Methods of Analog-to-Digital Conversion 704
- 12-3 Methods of Digital-to-Analog Conversion 715
- 12-4 Digital Signal Processing 723
- 12-5 The Digital Signal Processor (DSP) 724

CHAPTER 13 Data Transmission 739

- 13-1 Data Transmission Media 740
- 13-2 Methods and Modes of Data Transmission 745
- 13-3 Modulation of Analog Signals with Digital Data 750
- 13-4 Modulation of Digital Signals with Analog Data 753
- 13-5 Multiplexing and Demultiplexing 759
- 13-6 Bus Basics 764
- 13-7 Parallel Buses 769
- 13-8 The Universal Serial Bus (USB) 775
- 13-9 Other Serial Buses 778
- 13-10 Bus Interfacing 784

CHAPTER 14 Data Processing and Control 801

- 14-1 The Computer System 802
- 14-2 Practical Computer System Considerations 806
- 14-3 The Processor: Basic Operation 812
- 14-4 The Processor: Addressing Modes 817
- 14-5 The Processor: Special Operations 823
- 14-6 Operating Systems and Hardware 828
- 14-7 Programming 831
- 14-8 Microcontrollers and Embedded Systems 838
- 14-9 System on Chip (SoC) 844

ON WEBSITE: <http://www.pearsonglobaleditions.com/floyd>

CHAPTER 15 Integrated Circuit Technologies 855

- 15-1 Basic Operational Characteristics and Parameters 856
- 15-2 CMOS Circuits 863
- 15-3 TTL (Bipolar) Circuits 868
- 15-4 Practical Considerations in the Use of TTL 873
- 15-5 Comparison of CMOS and TTL Performance 880
- 15-6 Emitter-Coupled Logic (ECL) Circuits 881
- 15-7 PMOS, NMOS, and E²CMOS 883

ANSWERS TO ODD-NUMBERED PROBLEMS A-1

GLOSSARY A-31

INDEX A-42

Number Systems, Operations, and Codes

CHAPTER OUTLINE

- 2-1 Decimal Numbers
- 2-2 Binary Numbers
- 2-3 Decimal-to-Binary Conversion
- 2-4 Binary Arithmetic
- 2-5 Complements of Binary Numbers
- 2-6 Signed Numbers
- 2-7 Arithmetic Operations with Signed Numbers
- 2-8 Hexadecimal Numbers
- 2-9 Octal Numbers
- 2-10 Binary Coded Decimal (BCD)
- 2-11 Digital Codes
- 2-12 Error Codes

CHAPTER OBJECTIVES

- Review the decimal number system
- Count in the binary number system
- Convert from decimal to binary and from binary to decimal
- Apply arithmetic operations to binary numbers
- Determine the 1's and 2's complements of a binary number
- Express signed binary numbers in sign-magnitude, 1's complement, 2's complement, and floating-point format
- Carry out arithmetic operations with signed binary numbers
- Convert between the binary and hexadecimal number systems
- Add numbers in hexadecimal form
- Convert between the binary and octal number systems
- Express decimal numbers in binary coded decimal (BCD) form

- Add BCD numbers
- Convert between the binary system and the Gray code
- Interpret the American Standard Code for Information Interchange (ASCII)
- Explain how to detect code errors
- Discuss the cyclic redundancy check (CRC)

KEY TERMS

Key terms are in order of appearance in the chapter.

- | | |
|-------------------------|---------------------------------|
| ■ LSB | ■ BCD |
| ■ MSB | ■ Alphanumeric |
| ■ Byte | ■ ASCII |
| ■ Floating-point number | ■ Parity |
| ■ Hexadecimal | ■ Cyclic redundancy check (CRC) |
| ■ Octal | |

VISIT THE WEBSITE

Study aids for this chapter are available at <http://www.pearsonglobaleditions.com/floyd>

INTRODUCTION

The binary number system and digital codes are fundamental to computers and to digital electronics in general. In this chapter, the binary number system and its relationship to other number systems such as decimal, hexadecimal, and octal are presented. Arithmetic operations with binary numbers are covered to provide a basis for understanding how computers and many other types of digital systems work. Also, digital codes such as binary coded decimal (BCD), the Gray code, and the ASCII are covered. The parity method for detecting errors in codes is introduced. The TI-36X calculator is used to illustrate certain operations. The procedures shown may vary on other types.

2-1 Decimal Numbers

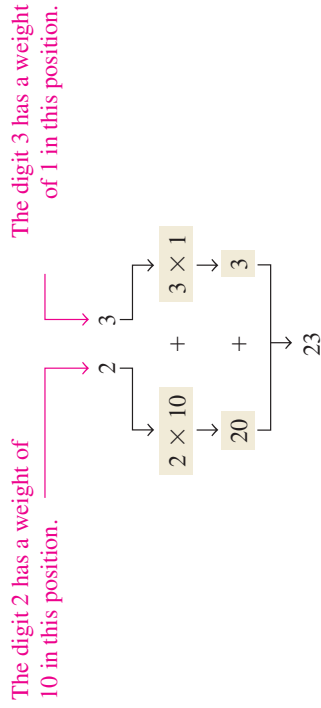
You are familiar with the decimal number system because you use decimal numbers every day. Although decimal numbers are commonplace, their weighted structure is often not understood. In this section, the structure of decimal numbers is reviewed. This review will help you more easily understand the structure of the binary number system, which is important in computers and digital electronics.

After completing this section, you should be able to

- ◆ Explain why the decimal number system is a weighted system
- ◆ Explain how powers of ten are used in the decimal system
- ◆ Determine the weight of each digit in a decimal number

The decimal number system has ten digits.

In the **decimal** number system each of the ten digits, 0 through 9, represents a certain quantity. As you know, the ten symbols (**digits**) do not limit you to expressing only ten different quantities because you use the various digits in appropriate positions within a number to indicate the magnitude of the quantity. You can express quantities up through nine before running out of digits; if you wish to express a quantity greater than nine, you use two or more digits, and the position of each digit within the number tells you the magnitude it represents. If, for example, you wish to express the quantity twenty-three, you use (by their respective positions in the number) the digit 2 to represent the quantity twenty and the digit 3 to represent the quantity three, as illustrated below.



The decimal number system has a base of 10.

The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a **weight**. The weights for whole numbers are positive powers of ten that increase from right to left, beginning with $10^0 = 1$.

$$\dots 10^5 10^4 10^3 10^2 10^1 10^0$$

For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with 10^{-1} .

$$10^2 10^1 10^0 . 10^{-1} 10^{-2} 10^{-3} \dots$$

← Decimal point

The value of a digit is determined by its position in the number.

The value of a decimal number is the sum of the digits after each digit has been multiplied by its weight, as Examples 2-1 and 2-2 illustrate.

EXAMPLE 2-1

Express the decimal number 47 as a sum of the values of each digit.

Solution

The digit 4 has a weight of 10, which is 10^1 , as indicated by its position. The digit 7 has a weight of 1, which is 10^0 , as indicated by its position.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

Related Problem*

Determine the value of each digit in 939.

*Answers are at the end of the chapter.

EXAMPLE 2-2

Express the decimal number 568.23 as a sum of the values of each digit.

Solution

The whole number digit 5 has a weight of 100, which is 10^2 , the digit 6 has a weight of 10, which is 10^1 , the digit 8 has a weight of 1, which is 10^0 , the fractional digit 2 has a weight of 0.1, which is 10^{-1} , and the fractional digit 3 has a weight of 0.01, which is 10^{-2} .

$$\begin{aligned} 568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500 + 60 + 8 + 0.2 + 0.03} \end{aligned}$$

Related Problem

Determine the value of each digit in 67.924.

CALCULATOR SESSION**Powers of Ten**

Find the value of 10^3 .

TI-36X **Step 1:** 1 0 y^x
Step 2: 3 =
 1000

SECTION 2-1 CHECKUP

Answers are at the end of the chapter.

- What weight does the digit 7 have in each of the following numbers?
 (a) 1370 (b) 6725 (c) 7051 (d) 58.72
- Express each of the following decimal numbers as a sum of the products obtained by multiplying each digit by its appropriate weight:
 (a) 51 (b) 137 (c) 1492 (d) 106.58

2-2 Binary Numbers

The binary number system is another way to represent quantities. It is less complicated than the decimal system because the binary system has only two digits. The decimal system with its ten digits is a base-ten system; the binary system with its two digits is a base-two system. The two binary digits (bits) are 1 and 0. The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit. The weights in a binary number are based on powers of two.

After completing this section, you should be able to

- ♦ Count in binary
- ♦ Determine the largest decimal number that can be represented by a given number of bits
- ♦ Convert a binary number to a decimal number

Counting in Binary

The binary number system has two digits (bits).

To learn to count in the binary system, first look at how you count in the decimal system. You start at zero and count up to nine before you run out of digits. You then start another digit position (to the left) and continue counting 10 through 99. At this point you have exhausted all two-digit combinations, so a third digit position is needed to count from 100 through 999.

A comparable situation occurs when you count in binary, except that you have only two digits, called *bits*. Begin counting: 0, 1. At this point you have used both digits, so include another digit position and continue: 10, 11. You have now exhausted all combinations of two digits, so a third position is required. With three digit positions you can continue to count: 100, 101, 110, and 111. Now you need a fourth digit position to continue, and so on. A binary count of zero through fifteen is shown in Table 2–1. Notice the patterns with which the 1s and 0s alternate in each column.

The binary number system has a base of 2.

InfoNote

In processor operations, there are many cases where adding or subtracting 1 to a number stored in a counter is necessary. Processors have special instructions that use less time and generate less machine code than the ADD or SUB instructions. For the Intel processors, the INC (increment) instruction adds 1 to a number. For subtraction, the corresponding instruction is DEC (decrement), which subtracts 1 from a number.

TABLE 2–1

Decimal Number	Binary Number			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The value of a bit is determined by its position in the number.

CALCULATOR SESSION

Powers of Two

Find the value of 2^5 .

TI-36X Step 1: 2 y^x

Step 2: 5 =

32

As you have seen in Table 2–1, four bits are required to count from zero to 15. In general, with n bits you can count up to a number equal to $2^n - 1$.

$$\text{Largest decimal number} = 2^n - 1$$

For example, with five bits ($n = 5$) you can count from zero to thirty-one.

$$2^5 - 1 = 32 - 1 = 31$$

With six bits ($n = 6$) you can count from zero to sixty-three.

$$2^6 - 1 = 64 - 1 = 63$$

An Application

Learning to count in binary will help you to basically understand how digital circuits can be used to count events. Let’s take a simple example of counting tennis balls going into a box from a conveyor belt. Assume that nine balls are to go into each box.

The counter in Figure 2–1 counts the pulses from a sensor that detects the passing of a ball and produces a sequence of logic levels (digital waveforms) on each of its four parallel outputs. Each set of logic levels represents a 4-bit binary number (HIGH = 1 and LOW = 0), as indicated. As the decoder receives these waveforms, it decodes each set of four bits and converts it to the corresponding decimal number in the 7-segment display. When the counter gets to the binary state of 1001, it has counted nine tennis balls, the display shows decimal 9, and a new box is moved under the conveyor belt. Then the counter goes back to its zero state (0000), and the process starts over. (The number 9 was used only in the interest of single-digit simplicity.)

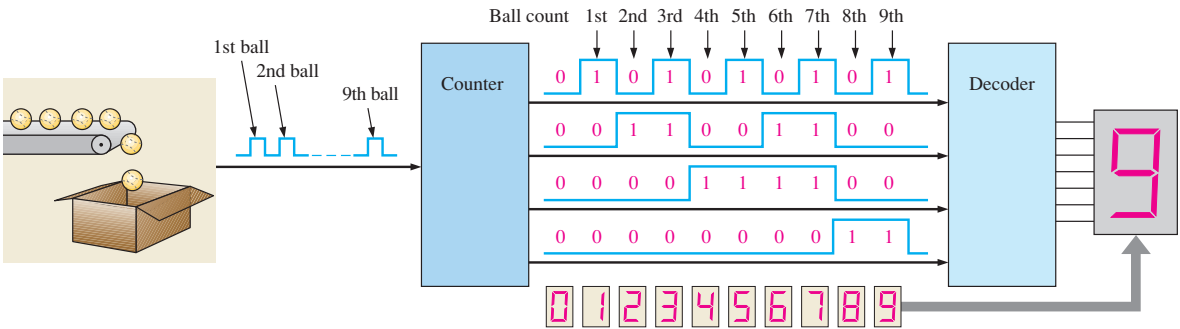


FIGURE 2-1 Illustration of a simple binary counting application.

The Weighting Structure of Binary Numbers

A binary number is a weighted number. The right-most bit is the **LSB** (least significant bit) in a binary whole number and has a weight of $2^0 = 1$. The weights increase from right to left by a power of two for each bit. The left-most bit is the **MSB** (most significant bit); its weight depends on the size of the binary number.

The weight or value of a bit increases from right to left in a binary number.

Fractional numbers can also be represented in binary by placing bits to the right of the binary point, just as fractional decimal digits are placed to the right of the decimal point. The left-most bit is the MSB in a binary fractional number and has a weight of $2^{-1} = 0.5$. The fractional weights decrease from left to right by a negative power of two for each bit.

The weight structure of a binary number is

$$2^{n-1} \dots 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} \dots 2^{-n}$$

↑
Binary point

where n is the number of bits from the binary point. Thus, all the bits to the left of the binary point have weights that are positive powers of two, as previously discussed for whole numbers. All bits to the right of the binary point have weights that are negative powers of two, or fractional weights.

The powers of two and their equivalent decimal weights for an 8-bit binary whole number and a 6-bit binary fractional number are shown in Table 2–2. Notice that the weight doubles for each positive power of two and that the weight is halved for each negative power of two. You can easily extend the table by doubling the weight of the most significant positive power of two and halving the weight of the least significant negative power of two; for example, $2^9 = 512$ and $2^{-7} = 0.0078125$.

InfoNote

Processors use binary numbers to select memory locations. Each location is assigned a unique number called an *address*. Some microprocessors, for example, have 32 address lines which can select 2^{32} (4,294,967,296) unique locations.

TABLE 2-2

Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

Binary-to-Decimal Conversion

Add the weights of all 1s in a binary number to get the decimal value.

The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.

EXAMPLE 2-3

Convert the binary whole number 1101101 to decimal.

Solution

Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

$$\begin{aligned}
 \text{Weight: } & 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 \\
 \text{Binary number: } & 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \\
 1101101 = & 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\
 = & 64 + 32 + 8 + 4 + 1 = \mathbf{109}
 \end{aligned}$$

Related Problem

Convert the binary number 10010001 to decimal.

EXAMPLE 2-4

Convert the fractional binary number 0.1011 to decimal.

Solution

Determine the weight of each bit that is a 1, and then sum the weights to get the decimal fraction.

$$\begin{aligned}
 \text{Weight: } & 2^{-1} \ 2^{-2} \ 2^{-3} \ 2^{-4} \\
 \text{Binary number: } & 0.1 \ 0 \ 1 \ 1 \\
 0.1011 = & 2^{-1} + 2^{-3} + 2^{-4} \\
 = & 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}
 \end{aligned}$$

Related Problem

Convert the binary number 10.111 to decimal.

SECTION 2-2 CHECKUP

1. What is the largest decimal number that can be represented in binary with eight bits?
2. Determine the weight of the 1 in the binary number 10000.
3. Convert the binary number 10111101.011 to decimal.

2-3 Decimal-to-Binary Conversion

In Section 2-2 you learned how to convert a binary number to the equivalent decimal number. Now you will learn two ways of converting from a decimal number to a binary number.

After completing this section, you should be able to

- ♦ Convert a decimal number to binary using the sum-of-weights method
- ♦ Convert a decimal whole number to binary using the repeated division-by-2 method
- ♦ Convert a decimal fraction to binary using the repeated multiplication-by-2 method

Sum-of-Weights Method

One way to find the binary number that is equivalent to a given decimal number is to determine the set of binary weights whose sum is equal to the decimal number. An easy way to remember binary weights is that the lowest is 1, which is 2^0 , and that by doubling any weight, you get the next higher weight; thus, a list of seven binary weights would be 64, 32, 16, 8, 4, 2, 1 as you learned in the last section. The decimal number 9, for example, can be expressed as the sum of binary weights as follows:

$$9 = 8 + 1 \quad \text{or} \quad 9 = 2^3 + 2^0$$

Placing 1s in the appropriate weight positions, 2^3 and 2^0 , and 0s in the 2^2 and 2^1 positions determines the binary number for decimal 9.

$$\begin{array}{cccc} 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 0 & 0 & 1 \end{array} \quad \text{Binary number for decimal 9}$$

To get the binary number for a given decimal number, find the binary weights that add up to the decimal number.

EXAMPLE 2-5

Convert the following decimal numbers to binary:

- (a) 12 (b) 25
(c) 58 (d) 82

Solution

- (a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow 1100$
 (b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow 11001$
 (c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow 111010$
 (d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow 1010010$

Related Problem

Convert the decimal number 125 to binary.

Repeated Division-by-2 Method

A systematic method of converting whole numbers from decimal to binary is the *repeated division-by-2* process. For example, to convert the decimal number 12 to binary, begin by dividing 12 by 2. Then divide each resulting quotient by 2 until there is a 0 whole-number quotient. The **remainders** generated by each division form the binary number. The first remainder to be produced is the LSB (least significant bit) in the binary number, and the

To get the binary number for a given decimal number, divide the decimal number by 2 until the quotient is 0. Remainders form the binary number.