21927

MICROCOMPUTER MATH

BY WILLIAM BARDEN, JR.

Microcomputer Math

William Barden, Jr., is a computer consultant specializing in small configuration computer systems. He has nearly 20 years experience in computer programming and computer systems analysis and design on a variety of computer systems. Mr. Barden is a member of the Association for Computing Machinery and the IEEE. His major interest is home computer systems. Amateur radio, mathematical games, and sailing are among his other interests.

Mr. Barden's books for SAMS include *How to Buy and Use Minicomputers and Microcomputers, How to Program Microcomputers, Microcomputers for Business Applications, Z-80 Microcomputer Handbook, and Z-80 Microcomputer Design Projects.*

Microcomputer Math

by

William Barden, Jr.

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

Indianapolis, IN 46268

SECOND EDITION
FIRST PRINTING—1982

International Standard Book Number: 0-672-21927-1
Library of Congress Catalog Card Number: 81-86554

Edited by: *Richard Krajewski*
Illustrated by: *T. R. Emrick*

Printed in the United States of America.

Preface

It isn't very long after the purchase of a microcomputer system that the microcomputer user encounters ominous references to "binary numbers," "hexadecimal value," "ANDing two numbers to get the result," or "shifting the result to multiply by two." Sometimes these references assume that the reader knows the binary system and operations; other times one gets the distinct feeling that the writer of the reference manual doesn't really know all that much about the operations either.

The purpose of *Microcomputer Math* is to remove some of the mystery surrounding the specialized math operations that are used in both BASIC language and assembly language. Such operations as binary number representation, octal and hexadecimal representation, two's complement operation, addition and subtraction of binary numbers, "flag" bits in microcomputers, logical operations and shifting, multiplication and division algorithms, multiple-precision operations, fractions and scaling, and floating-point operations are explained in detail, along with practical examples and exercises for self-testing.

If you can add, subtract, multiply, and divide decimal numbers, then you can easily perform the same operations in binary and other number bases, such as hexadecimal. This book will show you how.

This book makes an excellent companion to any course in assembly language or advanced BASIC.

Microcomputer Math is organized into ten chapters. Most of the chapters are based upon material contained in preceding chapters. Each chapter has self-test exercises at the end. It is to your benefit to go through the exercises, as they help fix the material in your mind, but we won't hold it against you if you use the book for reference only.

As you read, you'll notice some words are set in **boldface.** Most of these are computer terms that are defined in the Glossary. Using the Glossary, even the complete novice can easily learn and profit from this book.

The book is arranged as follows:

Chapter 1 discusses the binary system from the ground up and covers the conversions between binary and decimal numbers, while Chapter 2 describes octal and hexadecimal numbers, and conversions between these "bases" and decimal numbers. Hexadecimal numbers are used in both BASIC and assembly language.

Signed numbers and two's complement notation are covered in Chapter 3. Two's complement notation is used for negative numbers.

Chapter 4 discusses carries, overflow, and flags. These topics are used mostly in machine language or assembly language programming, but can be important in special BASIC programs also.

Logical operations, such as BASIC ANDs, ORs, and NOTs, are described in Chapter 5, along with the types of shifting possible in machine language. Then, Chapter 6 discusses multiplication and division algorithms, including both "unsigned" and "signed" operations.

Chapter 7 describes multiple-precision operations. Multiple precision may be used in both BASIC and assembly language to implement "unlimited precision" of any number of digits.

Chapter 8 covers binary fractions and scaling. This material is necessary to understand the internal format of floating-point numbers in BASIC.

Next, ASCII codes and ASCII conversions as they relate to numeric quantities are described in Chapter 9.

Finally, Chapter 10 provides an explanation of floating-point number representation as these numbers are used in many Microsoft BASIC interpreters.

Then, in the last section of the book, Appendix A contains the answers to the self-test questions, Appendix B is a glossary of terms, and Appendix C contains a listing of binary, octal, decimal, and hexadecimal numbers from 0 through 1023. The listing may be used for conversion from one type of number to the other. Finally, Appendix D contains a listing of two's complement numbers from −1 to −128, a handy reference that is not usually found in other texts.

WILLIAM BARDEN, JR.

To Dave Gunzel

A programmed instruction course that uses this book as a reference is available for many microcomputers from:

William Barden, Jr., Inc.
Post Office Box 3568
Mission Viejo, CA 92692

Send a self-addressed envelope for further information.

Contents

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

CHAPTER 10

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

CHAPTER 1

The Binary System— Where It All Begins

In the binary system, all numbers are represented by an "on/off" condition. Let's look at a quick example of binary in easy-to-understand terms.

BIG ED LEARNS BINARY

"Big Ed" Hackenbyte owns "Big Ed's," a restaurant that serves quick lunches and slow dinners in the middle of the area near San Jose, CA. This area, known as "Silicon Valley," has dozens of companies manufacturing microprocessor components for microcomputers.

Ed has eight serving people—Zelda, Olive, Trudy, Thelma, Fern, Fran, Selma, and Sidney. Depersonalization being what it is, they are assigned numbers for payroll reasons. The numbers assigned are:

	Number		Number
Zelda	0	Fern	4
Olive	1	Fran	5
Trudy	2	Selma	6
Thelma	3	Sidney	7

When Big Ed first implemented a "call board," he had eight lights, one for each of the serving people, as shown in Fig. 1–1. One day, though, Bob Borrow, a design engineer at a microprocessor components company known as Inlog, called Ed over.

"Ed, you could be a lot more efficient with your call board, you know. I can show you how we would have designed the board with one of our microprocessors."

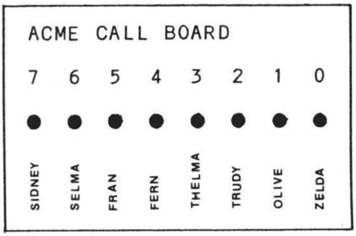

Fig. 1–1. Big Ed's call board.

Ed, being interested in the new technology, was receptive. The redesigned call board is shown in Fig. 1–2. It has three lights, controlled from the kitchen. When a serving person is called, a bell sounds. How is it possible to call any one of the eight serving people by lighting combinations of the three lights?

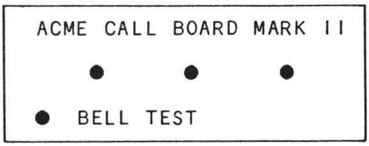

Fig. 1–2. A binary call board.

"You see, Ed, this board is very efficient. It uses five less lights than your first call board. There are eight different combinations of lights. We really call them **permutations,** as there's a definite order to the light arrangement. I've prepared a chart of the permutations of the lights and the serving person called." He gave Ed the chart shown in Fig. 1–3.

0	0	0	ZELDA	0
0	0	●	OLIVE	1
0	●	0	TRUDY	2
0	●	●	THELMA	3
●	0	0	FERN	4
●	0	●	FRAN	5
●	●	0	SELMA	6
●	●	●	SIDNEY	7

Fig. 1–3. Code chart for call board.

"There are only eight different permutations of lights, Ed—no more, no less. These lights are arranged in **binary** fashion. We use the binary system in our computers for two reasons. First of all, it saves parts. We reduced your number of lights from eight to three. Secondly, inexpensive computer components can usually represent only an on/off state, just as the lights are either on or off." He paused to sample some of his "Big Edburger." . . .

"I'll give these codes to my help to memorize," said Ed.

"Each serving person has only to memorize his unique code, Ed. I'll give you the key, so that you can decode which serving person is being called without reference to that chart."

"You see, each of the lights represents a power of two. The light on the right represents two to the zero power, the next light is two to the first power, and the leftmost light is two to the second power. This is really very similar to the decimal system, where each digit represents a power of ten." He scratched an example on the tablecloth, as shown in Fig. 1–4.

3 7 5

$$5 \times 10^0 = 5$$
$$7 \times 10^1 = 70$$
$$3 \times 10^2 = 300$$
$$375$$

Fig. 1–4. Comparison of decimal and binary notation.

1 0 1

$$1 \times 2^0 = 1$$
$$0 \times 2^1 = 0$$
$$1 \times 2^2 = 4$$
$$5$$

"Just as we can carry out the powers of ten to huge numbers, we can use as many powers of two as we want. We could use 32 lights, if we wanted. Now, to convert the three lights in binary to their decimal equivalent, add up the power of two for each light that is on." He scratched another figure on the tablecloth (Fig. 1–5).

"Well, that seems simple enough," Ed admitted. "From right to left, the lights represent 1, 2, and 4. If we had more waiters and waitresses, the lights would represent 1, 2, 4, 8, 16, 32, 64, 128 . . ." His voice trailed off, as he lost track of the next power of two.

"Exactly right, Ed. Typically, we have the equivalent of eight or sixteen lights in our microprocessors. We don't use lights, of course. We use semiconductor components that are either off or on." He scratched another figure on the tablecloth, which by now was overflowing with diagrams (Fig. 1–6).

"We call each of the eight or sixteen positions **bit positions.** 'Bit' is a contraction of the term 'binary digit.' After all, that's what we're talking about here, binary digits, just as decimal digits make up a decimal number. Your call board represents a 3-bit number."

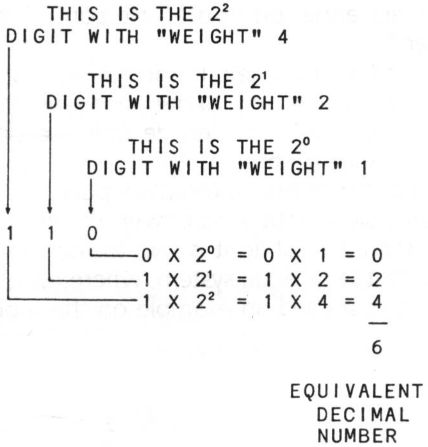

Fig. 1–5. Binary-to-decimal conversion.

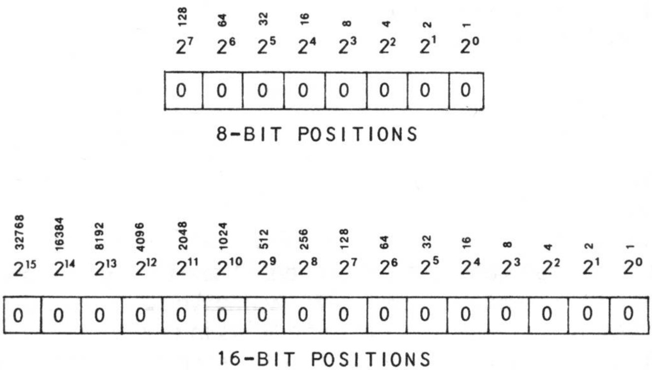

Fig. 1–6. An 8-bit and 16-bit representation.

"In eight bits we can represent any number between 0 and 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128. Adding all of those up, we get 255, the largest number that can be held in eight bits."

"How about sixteen bits?" asked Ed.

"You figure it out," said Bob. "I've got to get back to the job of designing microprocessors."

Ed drew up a list of all the powers of two up to fifteen. He then added them together to come up with the result shown in Fig. 1–7, a total of 65,535—the largest number that can be held in sixteen bits.

"This microprocessor business is easy," Ed said with a grin, as he bit into his Big Ed's Jumboburger with an 8-bit byte.