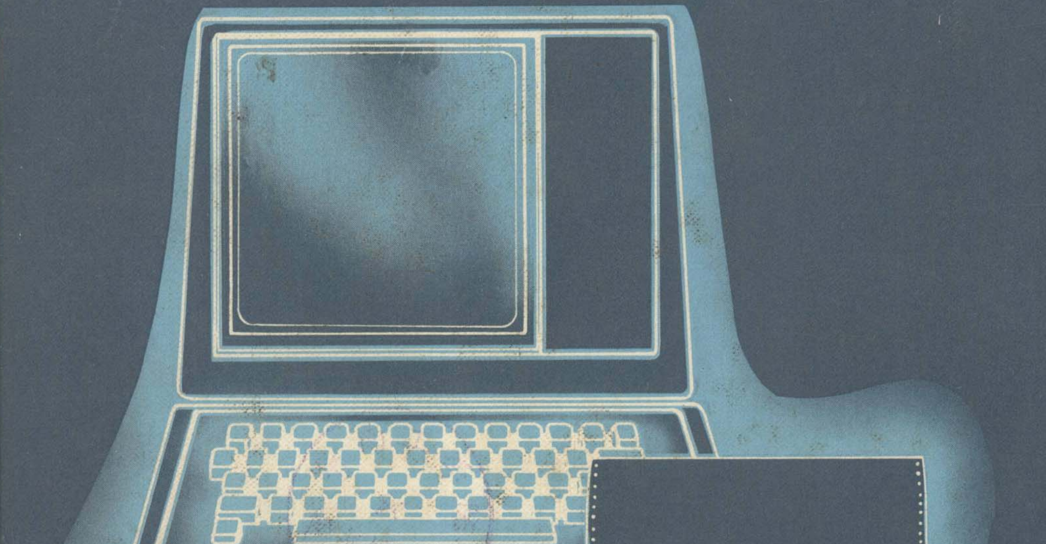
Microcomputer Systems



Hardware/Software Design

Robert M. Blasewitz & Frank Stern

HAYDEN

MICROCOMPUTER SYSTEMS

Hardware/Software Design

Robert M. Blasewitz
Frank Stern

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

To my loving parents, Paul and Cecilia and Dr. Robert I. Bickford, who believed in me from the beginning.

RMB

To my wife, Sandra, and daughter, Julie.

FPS

Library of Congress Cataloging in Publication Data

Blasewitz, Robert M.
Microcomputer systems.

Includes bibliographical references and index.
1. Microcomputers—Design and construction.
2. INTEL 8080 (Computer) I. Stern, Frank.
II. Title.
TK7888.3.B57 001.64 82-943
ISBN 0-8104-5123-9 AACR2

Copyright © 1982 by HAYDEN BOOK COMPANY, INC. All rights reserved. No part of this book may be reprinted, or reproduced, or utilized in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage and retrieval system, without permission in writing from the Publisher.

Printed in the United States of America

1 2 3 4 5 6 7 8 9 PRINTING

82 83 84 85 86 87 88 89 90 YEAR

MICROCOMPUTER SYSTEMS

Hardware/Software Design

Preface

This book is an excyclopedic look at the art of systems design based upon the 8080 microprocessor. It describes computer systems design from the bottom up, with emphasis on architectural components. Such treatment permits students who are trained in switching and logic theory but have very little programming background to understand evolved microcomputer systems with little difficulty. For these students, the book is supplied with selected fragments of software programs to illustrate specific concepts of computer architecture and systems design. The detailed hardware material it provides will serve as an introduction to those students with a programming background. Consequently, the book satisfies two needs, those of hardware and software design, and the integration of both into a system.

The presentation carries the student from an elementary overview through a detailed discussion of the fundamental aspects of the subject into several specialized and advanced topics. The text is highly modular, allowing instructors the option of constructing a semester course by drawing from chapters of their choice. The first four chapters couple the basics of logic design, both combinatorial and sequential, with state-of-the-art small-scale and large-scale integration techniques. The remaining chapters cover the components of a computer system, including memory, microprocessors, input-output, and software. The examples given allow students to become conversant with software as well as computer architecture. Advanced topics in the book include data acquisition systems, microcomputer boards, 16-bit microprocessors, and advanced technology.

Because computer technology is changing so rapidly, one of the pitfalls of any textbook in this field is the inclusion of material that easily becomes obsolete. This text deals with the problem of obsolescence by focusing on trends and general observations of prior technological development. Although a number of other microprocessors could have been chosen, the 8080 microprocessor was selected for its clear-cut design, wide use, and representative capability and performance. Use of the 8080 as the baseline microprocessor allows the student to extend his or her knowledge to 16-bit microprocessor systems without undergoing a tumultuous transition period.

此为试读,需要完整PDF请访问: www.ertongbook.com

Before closing, we would like to thank the many people who have contributed to this manuscript. Only through their efforts could a book of this magnitude reach completion.

Special thanks are due to Thomas Fooks and Norman Some for their encouragement and support of this project from its inception. A great deal of gratitude is also due to Dr. Robert Donnell and C. J. Lawrence for their many hours of consultation on the initial draft.

We would also like to thank Ingrid Dietiker, Linda Ciaccio, and Mary Buelow for the many hours they spent typing the manuscript and for having the patience to put up with the many changes made. Thanks also go to Jane Silber for her work on the problem sections and homework answers.

Last but not least we must express our sincerest appreciation to our wives, Virginia and Sandra, for the great encouragement they provided and for putting up with all those lost weekends.

ROBERT M. BLASEWITZ
FRANK STERN

Contents

Chapter 1

Introduction

The fundamental requirement of a computer is the ability to represent and store numbers and to perform operations on the numbers represented. Chapter 1 introduces various number systems and digital codes. Since the binary system has proved to be the most natural and efficient number system for machine use, it is explained in great detail. Other useful codes for representing information are also presented and compared.

1.1 Decimal and Binary Number Systems

Our present system of numbers has ten separate symbols, namely 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, which are called Arabic numerals. This decimal system counts in units of tens and was probably developed because of man's ten digits—his fingers. A number larger than 9 is represented through a convention that assigns a significance to the place or position occupied by a digit. In general, any number N_b in a fixed, positive, integral base b number system may be represented in positional notation as

$$N_b = A_{n-1}A_{n-2} \ldots A_2A_1A_0A_{-1}A_{-2} \ldots A_{-(m-1)}A_{-m}$$

where each A represents a digit in the order given, b = base of the system, n = number of integral digits, and m = number of fractional digits. (For decimal number 374.26, b = 10, n = 3, m = 2, $A_2 = 3$, $A_1 = 7$, $A_0 = 4$, $A_{-1} = 2$, and $A_{-2} = 6$.)

The numerical significance of this order is calculated as follows:

$$N_b = A_ib^{n-1} + A_ib^{n-2} \ldots A_ib^1 + A_ib^0 + A_ib^{-1} + A_ib^{-2} \ldots A_ib^{-(m-1)} + A_ib^{-m}$$

where $A_ib^{n-1} = A_{n-1}$, $A_ib^{n-2} = A_{n-2}$, $A_ib^1 = A_1$, and so forth. The positional coefficients A_i are such that

$$0 \leqslant A_i \leqslant (b-1)$$

Thus one can see that a number is expressed as a sum of the powers of its base b multiplied by the appropriate coefficients A_i. [In the decimal system, obviously, $0 \leqslant A_i \leqslant (b-1) \leqslant (10-1) \leqslant 9$.]

In general, we may represent any number N_b by the following summation:

$$N_b = \sum_{x=-m}^{n-1} A_i b^x$$

where x equals the power of the positional coefficient.

For example, the numerical significance of the decimal number 7043 $[A_i = 7, 0, 4, 3;\ m = 0;\ n = 4]$ is calculated as follows:

$$\begin{aligned} 7043_{10} &= (7 \times 10^3) + (0 \times 10^2) + (4 \times 10^1) + (3 \times 10^0) \\ &= 7000 + 0 + 40 + 3 \\ &= 7043 \end{aligned}$$

The great beauty and simplicity of our decimal number system can now be seen. It is necessary to learn only the ten basic numerals and the positional value system in order to represent any number.

It is, of course, quite feasible, and often very useful, to work with a number system that has a base other than 10, for example, the duodecimal system (base 12) is very handy when dealing with time, inches, feet, and dozens or grosses. However, in digital systems, a number with the base 2 is extremely useful. Such a system, called a *binary system*, uses just two simple digits, 0 and 1. Its advantage in regard to digital electronics lies in the fact that we may arrange a one-to-one correspondence between the two digits 0 and 1 and the two possible truth values (true or false) of a logical variable represented by the symbols 0 and 1. These binary variables involve only the presence or absence of a signal level and are especially easy to generate, transmit, and store reliably. We sometimes pay for this great convenience. Most problems or variables admit a much greater variety of possible states than just two and must therefore be represented in terms of ordered combinations of binary variables called *codes*.

In the binary system, the individual digits represent the coefficients of 2 rather than 10, as in the decimal number system. Again, any binary number may be represented as follows:

$$N_2 = \sum_{x=-m}^{n-1} A_i 2^x$$

where x, m, and n are defined as before. Since $(b-1) = (2-1) = 1$,

$$0 \leqslant A_i \leqslant 1$$

The binary number 11001, for example, is expressed as follows:

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

In other words, binary 11001 = decimal 25.

此为试读，需要完整PDF请访问：www.ertongbook.com

The process of counting can be used to point out similarities between the decimal and binary systems. In the decimal system, counting consists of increasing the digit in a particular position in the order 0, 1, 2, . . . , 8, 9. When we reach 10 in this position, we carry 1 to the immediate left position. Since the binary system can only go through two stages, we carry 1 to the immediate left position much more rapidly than in the decimal system. Thus, the numbers used in the binary system to count to a decimal value of 20 are those shown in Table 1-1.

Table 1-1. Equivalent Numbers in Decimal and Binary Notation

Decimal notation	*Binary notation*
A_1A_0	$A_4A_3A_2A_1A_0$
00	00000
01	00001
02	00010
03	00011
04	00100
05	00101
06	00110
07	00111
08	01000
09	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111
16	10000
17	10001
18	10010
19	10011
20	10100
$N_{10} = (A_1 10^1) + (A_0 \times 10^0)$	$N_2 = (A_4 2^4) + (A_3 2^3) + (A_2 2^2) + (A_1 2^1) + (A_0 2^0)$

Notice that fractional binary numbers may be converted in the same general way as in the decimal system. That is, just as 12.236 is equivalent to

$$(1 \times 10^1) + (2 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) + (6 \times 10^{-3})$$

in the decimal system, 1101.11101 is equivalent to

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

in the binary system.

Thus, it is easy to see where each digit in either system represents a certain weight, or weighing factor, to be used in representing a complete number system. For example, consider the conversion of binary 101.11 to a decimal number as shown in Table 1-2.

Table 1-2. Binary-to-Decimal Conversion

101.11 binary

A_2	A_1	A_0	A_{-1}	A_{-2}	*Weights*
1	0	1	1	1	
2^2 4	2^1 2	2^0 1	2^{-1} 1/2	2^{-2} 1/4	Binary Decimal
4 +	0 +	1	1/2 +	1/4	
	5	+		3/4	

5.75 decimal

This conversion leads us to the various mathematical methods or techniques for converting from the binary number system to the decimal number system. Decimal numbers can have, in general, an integer part and a fractional part. Each part should be converted separately into a binary equivalent. The complete representation is obtained by combining the two along with the binary point. There are two commonly used methods for converting decimal numbers to binary equivalents: the subtraction method and the division-multiplication method.

Subtraction Method

In this method, one must subtract the highest power of 2 from the decimal number and place a 1 in the appropriate weighing position of the partially completed binary number. This procedure must then be continued until the decimal number is reduced to zero—a tedious and laborious method for converting numbers. Although convenient for numbers of small magnitude since it can be performed mentally, it is rarely used for large numbers. The following example will help to illustrate the procedure of the subtraction method. Consider the number 53 in base 10:

1. Subtract from the decimal number the largest power of 2 contained therein. Since $2^5 = 32$ and $2^6 = 64$, the former is the largest power that can be subtracted:

$$\begin{array}{rr} & 53 \\ - & \underline{32} \text{ (or } 2^5\text{)} \\ R = & 21 \end{array}$$

Weight matrix

A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1					

2. Subtract the next largest power of 2 from the remainder R. Since $2^3 = 8$, $2^4 = 16$, and $2^5 = 32$, we subtract $2^4 = 16$:

$$\begin{array}{rl} & 21 \\ - & \underline{16} \text{ (or } 2^4) \\ R = & 5 \end{array}$$

Weight matrix					
A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1	1				

3. Subtract the next largest power of 2 from the remainder R. If after the last subtraction, the next largest power of 2 cannot be subtracted, place a zero in the weight matrix position. For example, since $2^3 = 8$ cannot be subtracted from $R = 5$, $A_3 = 0$. The next largest power of 2 that can be subtracted from $R = 5$ is $2^2 = 4$, as shown below:

$$\begin{array}{rl} & 5 \\ - & \underline{4} \text{ (or } 2^2) \\ R = & 1 \end{array}$$

Weight matrix					
A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1		

4. Follow step 3 given above, filling in the weight matrix as before. Since $2^0 = 1$ and $2^1 = 2$, we subtract the former:

$$\begin{array}{rl} & 1 \\ - & \underline{1} \text{ (or } 2^0) \\ R = & 0 \end{array}$$

Weight matrix					
A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1	0	1

We have now completed the conversion, and our weight matrix looks as follows:

A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	0	1

$$\begin{aligned} (53)_{10} &= A_5 \times 2^5 + A_4 \times 2^4 + A_3 \times 2^3 + A_2 \times 2^2 \\ &\quad + A_1 \times 2^1 + A_0 \times 2^0 \\ &= (A_5\, A_4\, A_3\, A_2\, A_1\, A_0)_2 \\ &= (1\ 1\ 0\ 1\ 0\ 1)_2 \end{aligned}$$

Fractional numbers can be converted in the same manner; for example, consider the fraction .5625 in base 10. To convert this number to a binary fraction, proceed as follows:

1. Subtract the largest fractional base 2 number from the decimal number. Since $2^{-1} = .50$ and $2^{-2} = .25$, we subtract the former:

$$\begin{array}{rl} & 0.5625 \\ - & \underline{.5000} \text{ (or } 2^{-1}) \\ R = & .0625 \end{array}$$

Weight matrix			
A_{-1}	A_{-2}	A_{-3}	A_{-4}
2^{-1}	2^{-2}	2^{-3}	2^{-4}
1			

2. Subtract the next largest base two number from the remainder. Since $2^{-4} = .0625$, we subtract it:

$$\begin{array}{rl} & .0625 \\ - & \underline{.0625} \text{ (or } 2^{-4}) \\ R = & .0000 \end{array}$$

Weight matrix			
A_{-1}	A_{-2}	A_{-3}	A_{-4}
2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	0	0	1

The subtraction method thus yields the following result:

$$(0.5625)_{10} = (.10010)_2$$

It should be noted that the remainder will sometimes not converge to zero very rapidly for a finite number of significant decimal digits. In effect, there may very well be a roundoff error in representing decimal numbers in binary form, irrespective of the number of binary digits available for the representation.

Division/Multiplication Method

Given a decimal integer, this method uses division by 2 to yield a conversion to a binary integer. If there is a remainder, we must place a 1 in the weight matrix under the lowest binary weight; if there is no remainder, we must place a zero in the weight matrix. We then divide the result of the first division by 2 and repeat the process until the result has been reduced to zero. For example, consider the decimal number 53:

	Remainder	Weight matrix					
		A_5	A_4	A_3	A_2	A_1	A_0
		2^5	2^4	2^3	2^2	2^1	2^0
53/2	1						1
26							
26/2	0					0	
13							
13/2	1				1		
6							
6/2	0			0			
3							
3/2	1		1				
1							
1/2	1	1					
0							
		$(1\ 1\ 0\ 1\ 0\ 1)_2 = (53)_{10}$					

If a decimal fraction must be converted to binary, apply a number of multiplications by 2. If the product is less than 1, the most significant binary digit is zero; if the product is greater than 1, the most significant binary digit is 1. The second digit is obtained by the same rule, operating this time on the fractional part of the product obtained from the first step. This process is continued until the desired degree of accuracy is obtained. As an example, consider the conversion of $(0.5623)_{10}$ to binary shown at the top of the following page.

Rounding off our result to the seven digits obtained thus far, we have

$$(0.5623)_{10} = (.1000111)_2$$

Since we have not carried our procedure far enough, we have at this point a roundoff error of 0.0076. As was said previously, a binary equivalent may not terminate and will thus produce a roundoff error. Since a word in a computer is of finite length, it is clear that the representation of a decimal fraction will usually involve an error, the magnitude of which will depend upon the length of the computer word.

		Weight matrix						
		A_{-1}	A_{-2}	A_{-3}	A_{-4}	A_{-5}	A_{-6}	A_{-7}
		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
$\begin{array}{r}0.5623\\ \times\quad 2\\ \hline 1.1246\end{array}$	>1	1						
$\begin{array}{r}0.1246\\ \times\quad 2\\ \hline 0.2492\end{array}$	<1		0					
$\begin{array}{r}0.2492\\ \times\quad 2\\ \hline 0.4984\end{array}$	<1			0				
$\begin{array}{r}0.4984\\ \times\quad 2\\ \hline 0.9968\end{array}$	<1				0			
$\begin{array}{r}0.9968\\ \times\quad 2\\ \hline 1.9936\end{array}$	>1					1		
$\begin{array}{r}0.9936\\ \times\quad 2\\ \hline 1.9872\end{array}$	>1						1	
$\begin{array}{r}0.9872\\ \times\quad 2\\ \hline 1.9744\end{array}$	>1							1

1.2 Other Number Systems

Octal Number System

To express a number in the binary number system, it is necessary to use substantially more digits than are required by the decimal number system. Since computers are built to serve man, it becomes necessary to have number systems that are easily manipulated and understood by both

此为试读,需要完整PDF请访问: www.ertongbook.com