MINICOMPUTERS IN DATA PROCESSING AND SIMULATION

MINICOMPUTERS IN DATA PROCESSING AND SIMULATION

BRANKO SOUČEK

Professor of Electrical Engineering
Institute Rudjer Bošković and University of Zagreb

WILEY-INTERSCIENCE, a Division of JOHN WILEY & SONS, INC.
New York · London · Sydney · Toronto

Library of Congress Catalog Card Number: 77-37436

ISBN 0-471-81390-7

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

PREFACE

In recent years, the application of digital computers to new engineering and scientific problems has resulted in many decisive advances. The introduction of integrated circuits and tens of thousands of small computers (minicomputers) is rapidly changing the profile of laboratory research and of measurement and control techniques. Many processes previously performed exclusively by analog hardware can now be practically realized with digital hardware and with minicomputers.

The purpose of this book is to present the theory and applications of digital simulation and measurement of data with emphasis on the use of minicomputers.

The book has ten chapters. The first part of the book (Chapters 1, 2, 3, 4, 5, and 6) concentrates on the basic principles of digital systems. It explains digital codes, logical systems, computer programming, organization, and interfacing. This material will provide the basic working knowledge necessary for the design of laboratory digital systems and for the programming and interfacing of minicomputers.

The second part of the book (Chapters 7, 8, 9, and 10) describes in detail new techniques of digital simulation and measurement of data. It explains the direct connection of the computer into the measuring chain, data simulation, data sampling and quantizing, data collection, and data analyzing. Unified treatment of material that is otherwise scattered over many publications and research reports is given. Previously unpublished methods based on research at Brookhaven National Laboratory and other laboratories are presented.

It is hoped that this volume will be useful to engineers and scientists performing experimental work in electronics, physics, chemistry, and biomedicine. The book is written as a reference for practicing engineers and scientists, as well as a textbook for students. The treatment is kept as straightforward as possible, with emphasis on functions and systems. A minimal background—like that offered to undergraduates—is assumed, although many fundamentals are reviewed.

Chapter 1 introduces the number systems and digital codes. It concentrates on the binary codes most widely used in computer technology. Arithmetic operations with binary numbers are explained. Different codes for information coding are shown and compared.

Chapter 2 deals with logical operations and digital circuits. It starts with a definition of binary variables and then explains basic logical operations. Fundamental concepts of digital-system design are introduced through consecutive levels: basic logical circuits, basic functional circuits, examples of the design of digital systems, and computer-aided digital-system design. The chapter ends with the description of the basic organization of digital computers, storage devices, and input-output devices.

Chapter 3 introduces computer instructions. It is shown that instructions commonly used in small scientific computers can be grouped into the following classes: word transmission, arithmetic and logical operations, control, register reference, and input-output. Definitions and examples of the use of the basic instructions are given. Different modes of computer addressing are explained.

Chapter 4 deals with programming. The basic procedures of programming are presented in logical order. The programming of control operations, loops, subroutines, arithmetic and logical operations, and input-output operations is explained. The chapter concentrates on the symbolic programming of those operations, and then explains briefly the corresponding FORTRAN statements. In this way, the reader learns symbolic coding and the meaning of operations as well as the philosophy of programming in algebraic language. For each operation, a few examples of short programs are given.

Chapter 5 discusses interfacing and describes direct connection of the small computer to the laboratory devices. Basic computer organization, implementation of instructions, and circulation of instructions and data between memory and main registers are explained. Basic modes of data transfer in and out of the computer are described: unconditional, conditional, and interrupt programmed transfer and direct memory access. For each mode, examples of programming and interfacing are given for typical laboratory operations. It is shown that through simple selection and timing many devices can simultaneously communicate with the laboratory computer. Some methods for comparison of the small computer input-output systems are given.

Chapter 6 presents minicomputers, which are becoming the basic tool in scientific and industrial laboratories. Detailed descriptions of typical instructions, input-output timing, and interface modules are given. Examples show the characteristic features of the minicomputers: variable and fixed input-output instructions, one-directional and bidirectional input-output buses, time-shared buses, priority interrupts. Special attention is given to advanced addressing modes which one finds in minicomputers with more than one accumulator and index register. The use of stacks and interfacing of the unibus are explained.

The second part of the book describes major uses of minicomputers in data simulation and measurement. Computers offer many advantages and can be used in a variety of applications, such as nuclear reactor monitoring, automatic film scanning, automated control and analysis of chemical samples, measuring of the responses of the brain, blood studies, clinical medicine, image processing, air and water pollution analysis, industrial process control, and communication. Although there is a large variety of uses, most of them can be described through the following basic operations: connection to the data source, data simulation, sampling and quantizing, data collection and analyzing, and instrumentation control.

Chapter 7 introduces the methods for data simulations. Direct simulations of a system is frequently the only possible method for conducting realistic studies and for experiment design. The chapter first describes techniques for sine, ramp, and rectangular wave generation. It concentrates after that on random data simulation and on Monte Carlo techniques. Poisson-weighted summations, which cover a large number of cases in engineering, physics, chemistry, and biology, have been simulated in three different ways: using a programmed random-number generator, using one physical source of random data, and using two physical sources of random data connected to the small computer. Electronic generation of pseudorandom numbers and time intervals is also described.

Chapter 8 deals with sampling and quantizing. It explains the very important sampling theorem and quantizing theorem. Procedures for the digitizing of continuous functions and for digital calculations can lead to serious errors if the rules derived from these theorems are not followed. The cases where fine sampling and quantizing are necessary are shown, as well as cases where just 1 bit provides enough information. At the end, a review of analog-digital conversion techniques is given and the choice of sampling and quantizing equipment is discussed.

Chapter 9 describes the methods for data collection. This is the basic operation in a large number of experiments, and it can be readily done with a minicomputer. Data buffering is explained first. Buffering increases the efficiency of the use of computer time and memory space and reduces the dead-time losses. Lists and list-searching techniques, suitable for laboratory applications, are described next. Sequential, linked, multiple-linked, and ordered lists are shown. Key-to-address transformation methods suitable for on-line use are described.

Chapter 10 deals with data analysis. Considerable progress is achieved if data are analyzed as they are collected. The simplest analysis is measuring the frequency of occurrence of data, or probability distribution measurement. Interfaces and program flow charts for the basic systems are given. Also, high-resolution systems, with millions of channels, are described. They are based on recently introduced techniques of associative list-processing and pseudorandom digital transformation and on the use of newly developed rotating memories. Techniques for random track addressing are shown, as well as the use of drums for data buffering. Remarkable results have been recently achieved through such techniques.

The treatment is rounded out with a large number of illustrations, examples, experimental results, and a bibliography of over 100 items. Each chapter is provided with problems for exercises.

A large number of solved examples appear throughout the book. These examples are designed as practical exercises for students or beginners in the use of the techniques.

I am employing the material in this book for teaching the following courses:

- Introduction to computer programming and interfacing.
- Computers in data measurement, processing, and simulation.

I can exchange information on these subjects with readers and teachers, and should be happy to receive comments, suggestions, and new material related to this book.

Branko Souček

September 1971
Zagreb, Yugoslavia

ACKNOWLEDGMENTS

I wish to acknowledge numerous contributions to this book.

Some of the experiments described have been performed at the "Rudjer Bošković" Institute. I should like to express my thanks to all members of the Laboratory for Cybernetics who collaborated with me in these experiments. I am particularly grateful to V. Škarić, director of the Institute, for his encouragement and support.

I acquired a large amount of my "know-how" working at the Brookhaven National Laboratory. My thanks are due to R. L. Chase and W. A. Higinbotham, division heads, who enabled me to work with them for a number of years in a highly stimulating and friendly atmosphere. Research with on-line computers has been carried on at that laboratory since 1963 and much pioneering work with small computers has been performed.

I am using part of the material described in the book in my classes at the University of Zagreb. I acknowledge the support received from Professors V. Muljević and H. Požar.

In writing a book like this one it was necessary to borrow some data from the manuals of computer manufacturers. I should like to thank the following companies for giving me permission to take material from their publications:

Digital Equipment Corporation
Hewlett-Packard Company
Rand Corporation
Varian Data Machines

I am grateful to I. N. Hooton for stimulating discussions and for allowing me to quote from his work.

I acknowledge the stimulation gained over a long period from many teachers and colleagues. I also acknowledge the help received from my students and the constructive criticism provided by a number of friends who reviewed parts of the text. Here I can name only a few: H. Babić, D. G. Dimmler, M. Dugaćki, C. P. Hohberger, N. E. Greenlaw, D. W. Potter, V. Radeka, S. Rankowitz, M. J. Rosenblum, H. Slivjanovski, and R. J. Spinrad.

CONTENTS

1

NUMBER SYSTEMS AND DIGITAL CODES

1.1. DECIMAL AND BINARY NUMBER SYSTEMS

Definitions

The fundamental requirement of a computer is the ability to represent and store numbers and to perform operations on the numbers thus represented. The numbers can be represented in different number systems. From early childhood most individuals are trained to represent quantities in the decimal number system. This system counts in units of tens and was probably developed because man has ten fingers. The base of this number system is 10, since ten basic digits are used (ten fingers). The basic digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. By giving to these digits different positions, or weights, we can express numbers larger than 10. Number 10 is base of the decimal system. One can build a number system with a different base, *b*.

In general, a number $(N)_b$ in a fixed, positive, integral base *b* number system is represented in positional notation as

$$(N)_b = (P_n P_{n-1} \cdots P_1 P_0 \cdot P_{-1} P_{-2} \cdots)$$

The number *b* is the base of the system. This is a positive integer and is fixed throughout the system. The positional digits P_i are integers such that

$$0 \leq P_i \leq b - 1 \qquad i = \cdots -2, -1, 0, 1, 2 \cdots n$$

The value of each position in a number is known as its position coefficient or weight. For example:

$$\begin{aligned} 346 = 3 \times 100 &= 300 \\ 4 \times 10 &= 40 \\ 6 \times 1 &= \underline{6} \\ &346 \end{aligned}$$

A simple decimal weighting table is

$$\cdots 10^3\ 10^2\ 10^1\ 10^0 \cdot 10^{-1}\ 10^{-2}$$

In general, in a base b system the successive digit positions, left to right, have the weights

$$\cdots b^3\ b^2\ b^1\ b^0 \cdot b^{-1}\ b^{-2} \cdots$$

The symbol "·" is called the radix point. The portion of the number to the right of the radix point is called the fractional part of the number, and the portion to the left is called the integral part. In the decimal system, the radix point is called the decimal point.

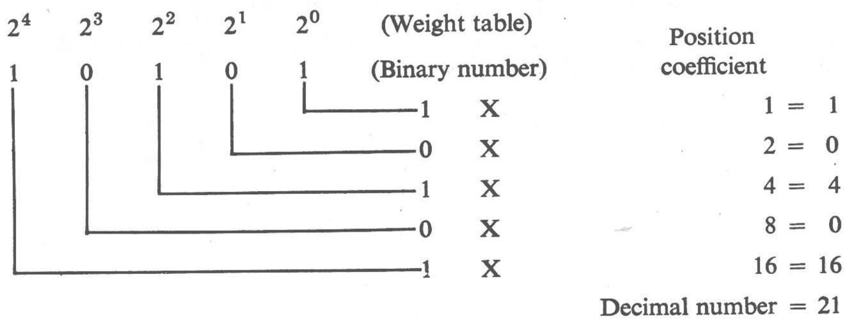
Computing machines can be built on the basis of any number systems. However, all modern digital computers are based on the binary (base 2) system. Why has this new number system been brought into use? The reason is that it is easier to distinguish between two entities than between ten entities. Most physical quantities have only two states: a light bulb is on or off; switches are on or off; material is magnetized or demagnetized; current is positive or negative; holes in paper tape or cards are punched or not punched; and so on. It is easier and more reliable to design circuits which must differentiate between only two conditions (binary 0 and binary 1) than between ten conditions (decimal 0 through 9).

The base of the binary system is 2. The radix point can be called the binary point. The possible digits are 0 and 1. The successive digit position, left to right, have the weights:

$$\cdots 2^3\ 2^2\ 2^1\ 2^0 \cdot 2^{-1}\ 2^{-2}\ 2^{-3} \cdots$$

This weighting table is used to convert binary numbers to the more familiar decimal system. For example, let us find the decimal equivalent of the binary number 10101 (Table 1.1).

Table 1.1



The process of counting can be used to point out differences and similarities between 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 0 (10) in this position, we carry 1 to the immediate-left position. Since the binary number system utilizes only two digits, particular positions of the number can go only through two changes, and then we carry 1 to the immediate-left position. Thus the numbers used in the binary number system to count up to a decimal value 10 are as shown in Table 1.2.

Table 1.2

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

There are some special names widely used in computer technology, such as "bit," "byte," and "word."

Bit. A *b*inary digi*t* is usually referred to as a *bit*. Thus a number such as 1010 is referred to as a 4-bit binary number, and 101 as a 3-bit number, and so on. The bit at the left end of the number is called the most significant bit (it has the largest weight). The bit at the right end of the number is called the least significant bit (it has the smallest weight). Figure 1.1 presents a binary number composed of 16 bits.

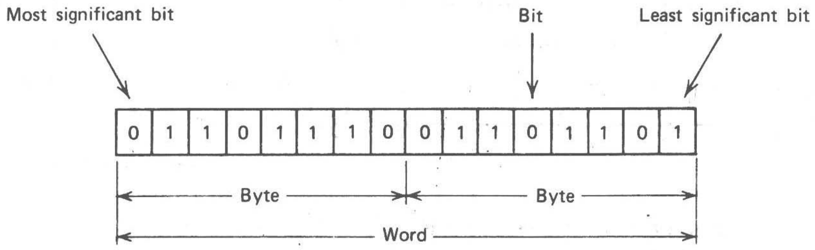


Figure 1.1. Bit, byte, word.

Byte. The evolution of the computer and data equipment has brought about an 8-bit unit for information exchange between devices. Such an 8-bit unit is referred to as a "byte." Many new types of digital computers and controllers thus express the numbers of 8, 16, 24, or 32 bits (1, 2, 3, or 4 bytes). Figure 1.1 presents a binary number composed of 2 bytes.

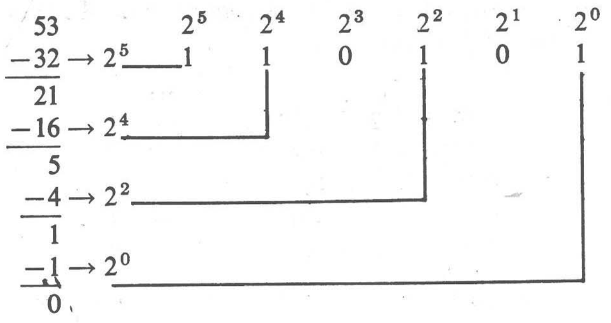
Word. The computer is composed of a large number of cells or registers for storing binary information. Most of the registers in a given machine are of the same length, *n*. Each register can be used to store *n* bits of binary information. Information stored in one register is also called a *word.* Figure 1.1 presents a word of a 16-bit computer.

Decimal-to-Binary Conversion

Decimal numbers can have, in general, an integer part and a fraction 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 division-multiplication method.

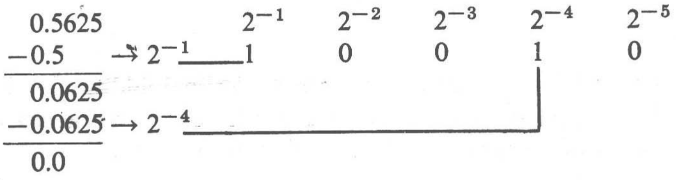
Subtraction method. (We follow here the procedure of Ref. 4). Subtract the highest possible power of two from the decimal number and place a 1 in the appropriate weighting position of the partially completed binary number. Continue this procedure until the decimal number is reduced to 0.

If, after the first subtraction, the next-lower power of 2 cannot be subtracted, place a 0 in the appropriate weighting position. An example for an integer is



Hence $(53)_{10} = (110101)_2$.

An example for a fraction is



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

Division method. If decimal integer should be converted into a binary equivalent, apply a number of divisions by 2. Divide the decimal number by 2. If there is a remainder, put a 1 in the lowest binary digit; if there is no remainder, put a 0 in the lowest binary digit. Divide the result from the first operation by 2 and repeat the process. Continue until the result has been reduced to 0 (Ref. 4). For example:

$$\begin{array}{r|l}
2 & 53 \quad \text{Remainder} \\
2 & 26 \quad \rightarrow \quad 1 \\
2 & 13 \quad \rightarrow \quad 0 \\
2 & 6 \quad \rightarrow \quad 1 \\
2 & 3 \quad \rightarrow \quad 0 \\
2 & 1 \quad \rightarrow \quad 1 \\
 & 0 \quad \rightarrow \quad 1
\end{array} \qquad (53)_{10} = (110101)_2$$

Multiplication method. If decimal fraction should be converted into binary equivalent, apply a number of multiplication 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 according to the same rule, operating this time on the fractional part of the product obtained in the first step. This process is continued until the desired degree of accuracy is reached. For example:

$$\begin{array}{lcl} & & \text{Carry} \\ 0.5625 \times 2 = 1.1250 & \rightarrow & 1 \\ 0.1250 \times 2 = 0.2500 & \rightarrow & 0 \\ 0.250 \times 2 = 0.5 & \rightarrow & 0 \\ 0.5 \times 2 = 1.0 & \rightarrow & 1 \\ 0.0 \times 2 = 0.0 & \rightarrow & 0 \end{array}$$

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

1.2. ARITHMETIC OPERATIONS WITH BINARY NUMBERS

Addition

Binary addition follows the same pattern as decimal addition except that a carry to the next position is not generated after the sum reaches 10, but is generated when the sum reaches 2 (1 + 1). For example:

$$\begin{array}{r@{\;=\;}l} 101 & 5_{10} \\ \underline{+010} & 2_{10} \\ 111 & 7_{10} \end{array}$$

$$\begin{array}{r@{\;}l} 11 & \leftarrow \text{carries} \\ 111 & = 7_{10} \\ \underline{+101} & = 5_{10} \\ 1100 & = 12_{10} \end{array}$$

Let us follow the second example (add 111 to 101). First, $1 + 1 = 0$ plus a carry of 1. In the second column, 1 plus the carry $1 = 0$ plus another carry. The third column is $1 + 1 = 0$ with a carry plus the previous carry, or $1 + 1 + 1 = 11$. Our answer is 1100 (decimal 12), which is the correct solution for $7 + 5$.

The addition of binary numbers in digital machines is performed by a special unit called a *functional adder.*

Subtraction

Binary numbers may be directly subtracted in a manner similar to decimal subtraction. It is entirely possible to design a machine that contains a functional subtractor as well as a functional adder. This procedure is not followed by the computer designers. It has been noted that if one wishes to perform a subtraction operation, one merely changes the sign of the subtrahend and proceeds with an addition operation. It is only important to find the proper way to express the negative numbers.

To see how negative numbers can be represented in the computer, consider a mechanical register, such as a car mileage indicator. If the register rotates forward, it performs addition. If the register rotates backward, it performs subtraction. An example for a 5-digit register rotating backward is[4]

```
00004
00003
00002
00001
00000
99999
99998
99997
```

It should be clear that the number 99997 corresponds to -3. To prove that, we can perform addition

$$\begin{array}{r} 00004 \\ +\ 99997 \\ \hline 1\ 00001 \end{array}$$

If we neglect the carry to the left, we have effectively performed the operation of subtracting.

$$4 - 3 = 1$$

The number 99997 in this example is called the ten's complement of 3. Thus in a decimal number system negative numbers can be presented in the form of the ten's complement and the negative sign could be omitted.

In digital machines, in the same manner, the two's complement of binary numbers is used to represent negative numbers and to carry out binary subtraction.

The two's complement of a number is defined as that number which when added to the original number will result in a sum of unity. For example, the binary number 010 110 110 110 has a two's complement equal to 101 001 001 010 as shown in the following addition

$$\begin{array}{r} 010\ 110\ 110\ 110 \\ 101\ 001\ 001\ 010 \\ \hline 1\ 000\ 000\ 000\ 000 \end{array}$$

In order to obtain two's complement of a number, two steps should be followed.

1. Obtain the one's complement which is formed by getting each bit to the opposite value

$$\begin{array}{ll} 010\ 110\ 110\ 110 & \text{Number}\\ 101\ 001\ 001\ 001 & \text{One's complement of the number} \end{array}$$

2. Two's complement is equal to the one's complement plus 1

$$\begin{array}{rrl} & 101\ 001\ 001\ 001 & \text{One's complement of the number}\\ + & 1 & \text{Add 1}\\ \hline & 101\ 001\ 001\ 010 & \text{Two's complement of the number} \end{array}$$

An example of subtraction:

$$\begin{array}{rrl} & \multicolumn{2}{l}{7 - 3 = 4}\\ & 0011 & 3_{10}\\ & 1100 & \text{One's complement of 3}\\ & 1101 & \text{Two's complement of 3}\\ + & & \\ & 0111 & 7_{10}\\ \hline & 1\ 0100 & 4_{10} \end{array}$$

$$\begin{array}{rrr} & \multicolumn{2}{l}{12 - 5 = 7}\\ & 0101 & 5_{10}\\ & 1010 & \\ & 1011 & \\ + & & \\ & 1100 & 12_{10}\\ \hline & 1\ 0111 & 7_{10} \end{array}$$

Multiplication

In binary multiplication, the partial product is moved one position to the left as each successive multiplier is used. If the multiplier is 0, the partial product is 0; if the multiplier is 1, the partial product is equal to the multiplicand. Examples:

$$\begin{array}{rl} \multicolumn{2}{l}{5 \times 3 = 15}\\ 101 & 5_{10}\\ 11 & 3_{10}\\ \hline 101 & \\ 101 & \\ \hline 1111 & 15_{10} \end{array}
\qquad
\begin{array}{rl} \multicolumn{2}{l}{5 \times 5 = 25}\\ 101 & 5_{10}\\ 101 & 5_{10}\\ \hline 101 & \\ 000 & \\ 101 & \\ \hline 11001 & 25_{10} \end{array}
\qquad
\begin{array}{rl} \multicolumn{2}{l}{5 \times 10 = 50}\\ 101 & 5_{10}\\ 1010 & 10_{10}\\ \hline 000 & \\ 101 & \\ 000 & \\ 101 & \\ \hline 110010 & 50_{10} \end{array}$$

Division

Following the rules of binary subtraction and multiplication, one can perform binary division in the same way as decimal division. Examples:

18 : 2 = 9   10 : 5 = 2   14 : 4 = 3.5