MICROPROCESSOR PROGRAMMING and SOFTWARE DEVELOPMENT

F. G. DUNCAN

72.276
D911

MICROPROCESSOR PROGRAMMING and SOFTWARE DEVELOPMENT

F. G. DUNCAN
University of Bristol, England

Prentice/Hall PHI International

ENGLEWOOD CLIFFS, NEW JERSEY   LONDON   NEW DELHI
SINGAPORE   SYDNEY   TOKYO   TORONTO   WELLINGTON

11 6128

5506128

To T. A. M. and to S. W. D.

Library of Congress Cataloging in Publication Data

Duncan, Fraser G., 1932–
Microprocessor programming and software development.

Bibliography: p.
Includes index.
1. Microprocessors–Programming. I. Title.
QA76.6.D84 001.6'42 78-27429
ISBN 0-13-581405-7

British Library Cataloguing in Publication Data

Duncan, Fraser George
Microprocessor programming and software development.
1. Microprocessors–Programming
I. Title
001.6'42 QA76.6
ISBN 0-13-581405-7

ISBN 0-13-581405-7

PRENTICE-HALL INTERNATIONAL, INC., LONDON
PRENTICE-HALL OF AUSTRALIA PTY. LTD., SYDNEY
PRENTICE-HALL OF CANADA, LTD., TORONTO
PRENTICE-HALL OF INDIA PRIVATE LIMITED, NEW DELHI
PRENTICE-HALL OF JAPAN, INC., TOKYO
PRENTICE-HALL OF SOUTHEAST ASIA PTE., LTD., SINGAPORE
PRENTICE-HALL, INC., ENGLEWOOD CLIFFS, NEW JERSEY
WHITEHALL BOOKS LIMITED, WELLINGTON, NEW ZEALAND

Type set in Monotype Times Roman and Univers
by HBM Typesetting Ltd., Chorley, Lancs.
Printed and Bound in Great Britain
by A. Wheaton & Co. Ltd., Exeter

79 80 81 82 83 5 4 3 2 1

Preface

Microprocessor programming and software development is intended to provide new users of microprocessors with an introduction to programming and to the basic design of software. More precisely, it is meant as a teaching text and reference document for those newcomers to microprocessors and to programming who wish to understand fully both the programs they will write and the programs they will use.

The book originated in its author's own attempts to come to terms with a number of different microprocessors. It has subsequently derived a great deal from extensive discussions with students and colleagues facing the same task, and from the experience of many courses and seminars involving audiences of a wide variety of background and interest.

This book is offered in the first place as a text for students specializing in computer science in universities, polytechnics and similar institutions. It assumes no previous knowledge of programming, and its references to matters of logic design imply no more advanced experience of that subject than is normally acquired by such students in their first or second year. In the University of Bristol the material was first introduced to third-year students; by the time microprocessors have een established as a teaching medium throughout the undergraduate syllabus it will be a first-year text.

Mature students of many different backgrounds have worked through the mat ial of the book, some by attending evening or vacation courses, others by unaided private study. Their very valuable criticisms and suggestions have been taken into account. It is hoped, therefore, that the book will be of use to other prospective users of microprocessors who, already competent in their own fields, require an introduction to programming that will give them a sound basis for their own future work and the foundation of a critical understanding of the work of others. Such a reader will find the book self-contained as regards its principal subject matter; but if he finds difficulty in reading the first chapter he will no doubt consult an elementary text on logic design. Having done so he will be in a better position to understand the structure of a microprocessor system and to meet the task of using a microprocessor in a context of his own specification.

The approach taken by the book is practical, and, while much of the material is more generally applicable, all detailed discussion is based on four widely used processors, the Motorola 6800, Intel 8080 and 8085, and Zilog Z80.[1] A self-contained software package for the 8080 is given as an integral part of the work, and provides material for illustration throughout.

Chapter 1 gives a brief outline sketch of the 'hardware' aspects of microprocessors and microprocessor systems from a programmer's point of view.

In Chapter 2 the machine instructions and their effects are described and discussed in detail. A common notation for the instruction sets is developed in a manner intended to enable the reader to pass easily from one machine to another. This notation differs radically from the sets of so-called 'mnemonics' constituting the 'assembly languages' designed by the manufacturers; its advantages, however, are considerable, and particularly to the user who does not wish to be restricted to one machine. The application of certain types of instructions, such as those for addition and subtraction, is discussed at length; the reader who feels that this discussion is more than his anticipated needs demand should omit it from his first reading and treat it as reference material for possible later use.

Chapter 3 is given over to tabulations in common form of the instruction sets of the four microprocessors, 6800, 8080, 8085 and Z80, together with an alphabetical 'dictionary' of the written forms, introduced in Chapter 2, of the instructions of the combined set. This superficially forbidding mass of reference material will repay study; at a later stage it will be found that a copy of one or other of the tables will be a convenient and usually sufficient reference during program-writing.

Programming is introduced in Chapter 4 through a series of graded examples. The first of these provide a basis for the simple input, output and processing of characters and numbers; later ones correspond to subroutines of a software package, which are given later in full.

Chapter 5 consists of a set of more or less disjoint sections on topics whose only common feature is that they are not primarily concerned with the processing of numerical quantities. In some, such as that on sorting, a fairly detailed treatment of at least one practical example has been possible; but in others, such as that on files and records, anything more than a brief description of the topic would have required considerably more space than was available.

In Chapter 6 a set of basic software aids to programming is described, and users' instructions for such a set—that given in the software package—are detailed. The chapter ends with discussion of possibilities for enhancing the programmer's power of expression through further software.

Chapter 7 is devoted to a necessarily subjective account of the historical context of microprocessor software, brief references to available packages, and an inconclusive discussion of possibilities for the future. A reader who, in addition to working through the programming exercises of this book, has also used one or other of the available high-level language compilers should be in a position to involve himself in this

[1]The extent to which the material applies to another processor will depend, of course, on how different it is from these four. While, no doubt, some of it applies to processors of word-lengths other than 8 bits, it is unlikely that very much of it can usefully be applied to 'bit-slice' devices.

discussion and perhaps carry it forward to the point of contributing his own ideas to the design and implementation of new languages or other, better, means of expression.

Chapter 8 contains the complete text, with detailed commentary, of the software package which has already served as a source of illustrations of programming techniques, instructions for the use of which were given in Chapter 6. This package, occupying 6K bytes of PROM, consists of a monitor, a PROM-programming program, an assembler and disassembler, and a set of arithmetic (including 32-bit floating point) subroutines. The assembler requires at least 1K, and, preferably, 2K, bytes of RAM for working space. However, as described in Chapter 8, parts of the package can be used with much less PROM and RAM. This software has been in use for some time, and it will, it is hoped, be useful to the reader. It is, of course, imperfect, and unlikely to satisfy the critical user for long. The reader who has discovered for himself its usefulness, its limitations and its shortcomings, and who has studied it carefully and observed its structure, its scars, flaws and blemishes, should be well on the way to designing and implementing good software and other programs of his own.

Programming is a practical skill which can be acquired only by practice on actual machines. Clearly, a reader who has an 8080 or 8085 or Z80 system on which the software of Chapter 8 will run will derive much more from this book than will another who has only some other machine. But the latter will be infinitely better off than a third reader who has no access to any machine. All three readers would benefit by getting together to learn to program both of their machines. It is well known that a bilingual programmer is far better able to adapt to new machines and languages than a programmer who can express himself in only one way. A beginner in the fortunate position of being able to buy his own equipment will learn far more from two different small systems than from one larger one.

There is very little, if anything, in this book about cross-assemblers, cross-compilers, debugging aids with elaborate diagnostics or development systems. Such things are distractions to the beginner and symptoms of muddle and despair when used by an experienced programmer. That of course is a provocative over-simplification. The truth is that the system on which a program is to run is the right system on which to develop that program; that, while mistakes are inevitable, muddle and needless complication are not; and that simple, well-designed and easily understood tools, in programming as well as in any other constructional activity, are much to be preferred to elaborate, complicated and unpredictable mechanisms.

The inevitable mistakes in this book, and any avoidable muddle and complication, are the fault of the author, and all expressed opinions are his alone. At the same time, credit for whatever is worthwhile in it belongs ultimately to the many colleagues, teachers, students and friends who, wittingly and unwittingly, have made it possible and who, because of their number, have to be acknowledged collectively. With specific regard to the book, however, thanks are due to Professor M. H. Rogers, Head of the Department of Mathematics, University of Bristol, for initially suggesting and continually supporting the work with microprocessors,[2] and for repeated reminders that the work ought to be documented; to Professor C. A. R. Hoare of

[2]Here, and elsewhere, an ambiguity is to be understood in at least two senses.

Oxford University, for much encouragement and constructive criticism; to Messrs H. Hirschberg, R. Decent and their colleagues at Prentice-Hall International for necessary stimulation and practical expertise; to the reviewers for pointing out that a book is meant to be read by readers; to Professor D. Zissos, Professor J. Mühlbacher, and Messrs K. R. Brooks, D. Harvey, G. Pritt, and M. G. Wilkins for their many varied and valued contributions; to Mesdames R. Martin, A. Plumley, A. Warren-Cox and Miss R. Wilkins for their typing; to those microprocessor companies whose products have brought pleasure back into programming; and last, but by no means least, to my wife and (on the whole) our three children, who have suffered much over the past year and yet have continued to provide agreeable conditions for working with microprocessors and even more agreeable means of escape from them.

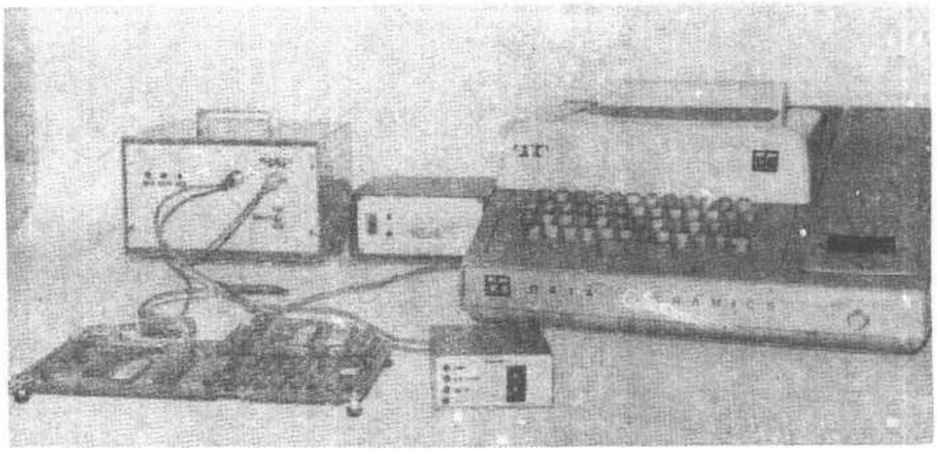
POSTSCRIPT (*June 1979*)

While this book has been in production, the software of Chapter 8 has been transcribed for an 8085 system with PROM (4×2716) in 0000-1FFF and RAM in 2000 onwards. Some of the improvements suggested in 8.7 have been effected. The assembler with slightly better facilities is in one 2K PROM (1800-1FFF); the monitor with new line-printer commands, the PROM programmer programs, and disassembler are in another (0000-07FF); the arithmetic and input–output subroutines are in a third (0800-0FFF); while the fourth (1000-17FF) has cassette and higher-level software under development. Users of this system should note the following transpositions:

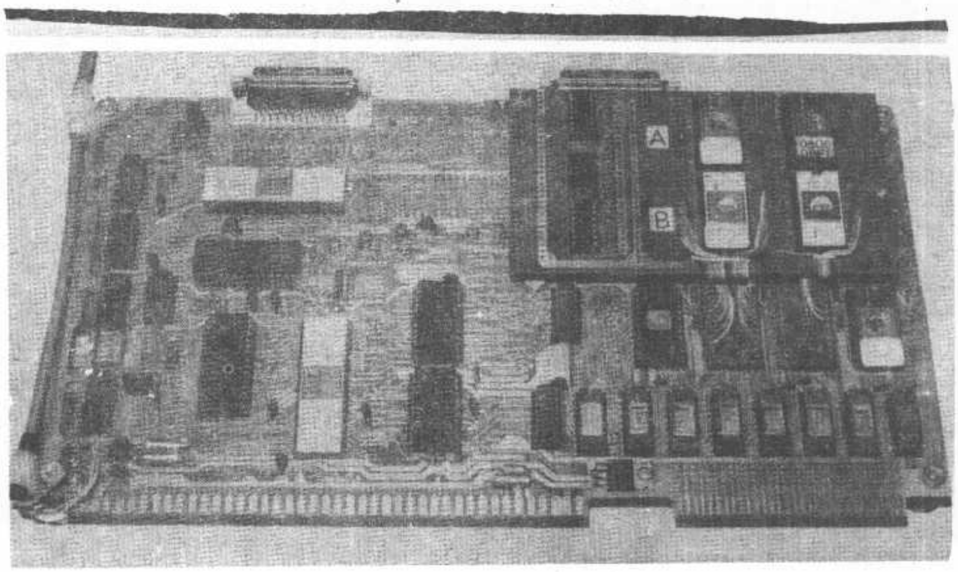
8080	*8085*	*8080*	*8085*
0360, 03C0	0371, 03C9	0D6F-0D7F	086F-087F
07E6-07FF	0886-089F	0DCA-0FFF	08CA-0AFF
0BE0-0BFF	08A0-08BF	2400-27FF	0C00-0FFF
0CF1-0CF7	08C1-08C7	2B80-2BFF	0B80-0BFF

This postscript provides a welcome opportunity to express thanks to Messrs A. Whittle, P. Woodward, and P. Day for their persistent care with the text and illustrations.

F.G.D.



The 8080 system described on pages 210 and 211. The PROM programmer is in the foreground; the power supply unit is on the left.



A closer view of the modified Intel SDK board with 6K of PROM and 2K of RAM.

Contents

1

Microprocessors and microprocessor systems

1.1 BASIC IDEAS

Physically, a *microprocessor* is a very large-scale integrated circuit containing the equivalent of several thousands of discrete components on a silicon chip about 4 mm square encapsulated within a dual-in-line package with (typically) 40 pins. Logically, it is a clocked sequential circuit, that is a circuit which will change its internal state, and hence its output signals, in phase with clock pulses. The clock pulses are usually, though not in every case, generated externally. Each change of internal state is determined by the processor's current internal state together with the set of input signals. So far this description applies equally to, say, a JK flip-flop; the difference is in the matter of complexity, for there are very many possible combinations of input signals, very many possible internal states, and very many possible combinations of output signals.

The *operation* of the microprocessor can be seen, however, in fairly simple terms. Essentially it amounts to the repetition of a cycle of internal states. This cycle, the *instruction cycle*, always comprises the following stages.

(i) A set of input signals (an *instruction*) is *read* (*latched*) into an internal register (the *instruction register*) of the processor.

(ii) The processor passes through a sequence of states determined by the bits composing the instruction, and possibly involving the reading of further input signals (*data*) or the generation of output signals (*results*). This is the *execution* of the instruction.

(iii) Finally, a set of output signals is generated (the *next instruction address*), which is used by external circuitry (normally the *store* or *memory*[1]) to determine the next instruction to be presented to the processor.

[1]'Memory: Part of a computer system used to store data.... Store: British word for "memory" ': Donald E. Knuth, *The Art of Computer Programming*. Volume 1: *Fundamental Algorithms*. Reading, Massachusetts: Addison-Wesley, 1968.
'*memory* (of a COMPUTER), see STORE. . . .
store. The most expensive part of a COMPUTER, where the information (both PROGRAM and data) is kept. The neutral term *store* is to be preferred to *memory* to avoid the danger of anthropomorphizing computers. . . . ': the late Christopher Strachey in *The Fontana Dictionary of Modern Thought*. London: Fontana Books, 1977.

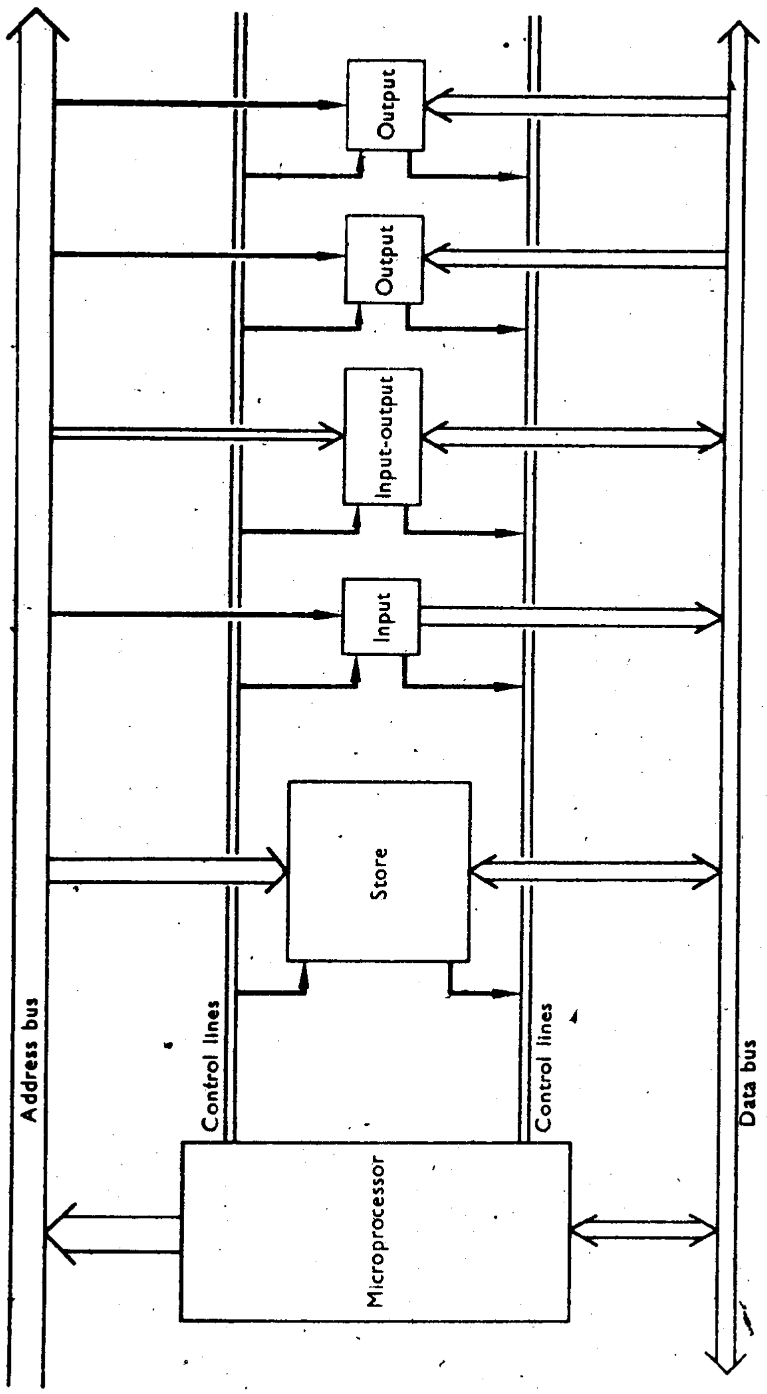


Figure 1.1. A typical microprocessor system.

The sequence of instructions required to cause the processor to accomplish some specific task is a *program*. The instructions of the program being executed are held in a *store* (conceptually a large array of bistable elements). As each instruction is required by the processor its *address* (the set of bits defining its location within the store) is generated by the processor and presented to the store. A copy of the instruction is then input to the processor; the contents of the store remain undisturbed. Now an address can be regarded as a binary number, and normally the *next instruction* is that whose address *follows* that of the current instruction in the numerical sense of having the next greater value. However, a given instruction may require to be followed in execution by an instruction which does not occupy the next address in store. The possibility of such *jump* instructions means that certain sequences of instructions may be executed many times while others may not be executed at all during some execution of the program, depending on the *data* with which the program is working. (The number of instructions stored is no clue to the number of instructions executed or to the time which a program will take to run.)

The microprocessor and store together form the basic *microprocessor system*. In general there will also be other system components, called *peripheral devices*. Typically there will be at least one *input device*, by means of which numbers or other quantities (for example, instructions or instrument readings) can be taken into the store or processor, and at least one *output device*, by means of which numbers (sets of bits) can be taken from the store or processor to the outside world (for example, to be printed, or to control the operation of a machine). A typical microprocessor system is illustrated in Figure 1.1.

The components are connected by sets of wires:

(i) the *data bus*, which may at any given time be carrying,
  (a) an instruction from the store to the processor;
  (b) a number from the store or an input device to the processor;
  (c) a number (result) from the processor to the store or an output device;
  (d) a number between one part of store and another, between store and a peripheral device, or between two peripheral devices;

(ii) the *address bus*, which may at any given time be carrying
  (a) the address of the next instruction required to be executed by the processor;
  (b) the address of a number in store required for computation by the processor;
  (c) the address into which a number computed by the processor is to be written;
  (d) the address (or *device number*) of an input device from which the processor is to receive a number;
  (e) the address (or device number) of an output device which is to receive a number from the processor;
  (f) an address concerned in a transfer as in (i) (d) above;

(iii) *control lines*, carrying timing and control signals which ensure the synchronization and coordination of the computer, in order to ensure the correct operation of transfers of information as listed under (i).

Some microprocessors require variations of this basic scheme. The integrated circuit may be so organized that the address bus and data bus have to share a common