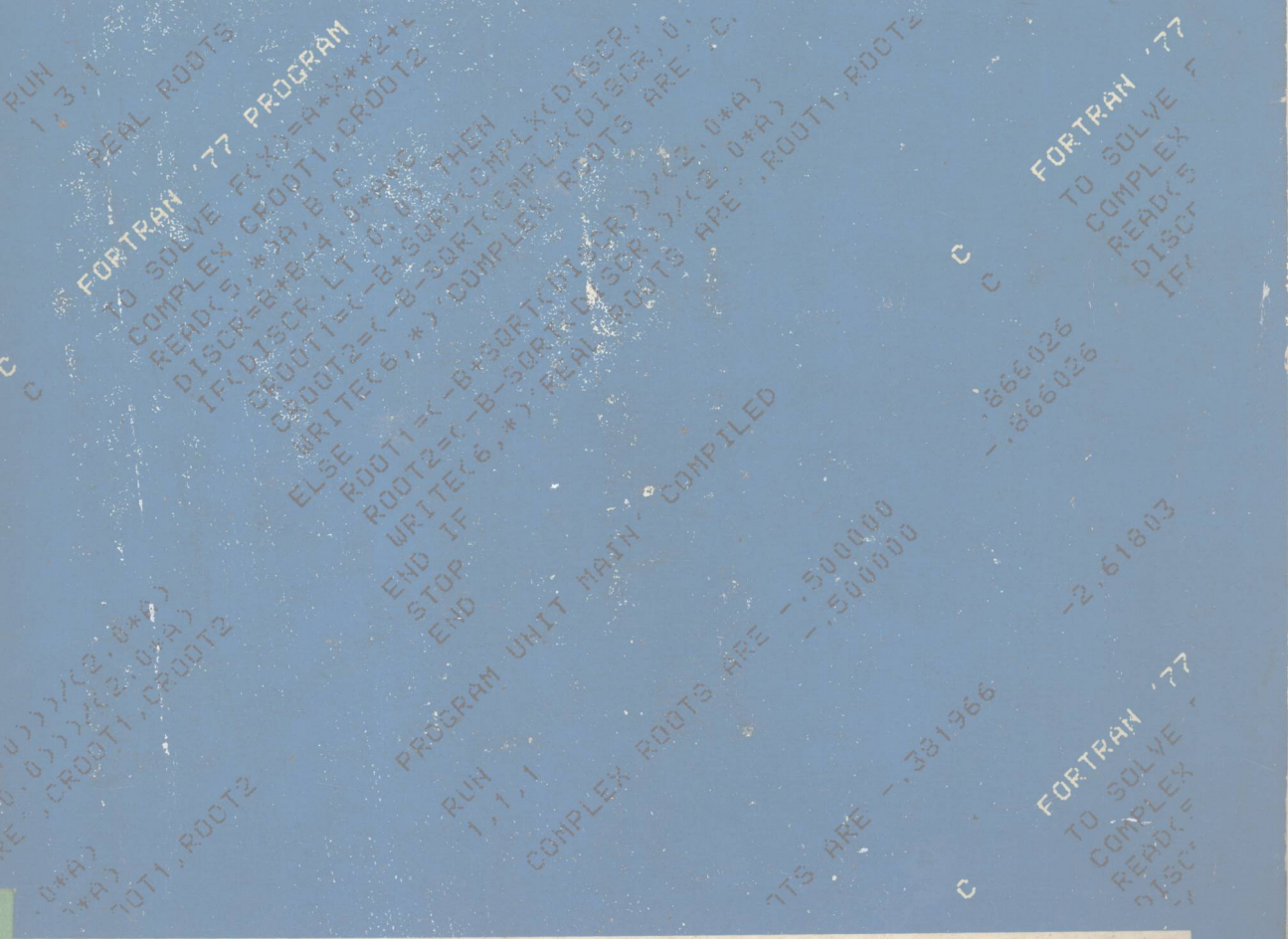
APPLIED FORTRAN

FOR ENGINEERING AND SCIENCE



James P. Schwar Charles L. Best

Applied FORTRAN

for Engineering and Science

JAMES P. SCHWAR
CHARLES L. BEST

Lafayette College
Easton, Pennsylvania

SRA ®
SCIENCE RESEARCH ASSOCIATES, INC.
Chicago, Palo Alto, Toronto
Henley-on-Thames, Sydney
A Subsidiary of IBM

Acquisition Editor	Alan W. Lowe
Project Editor	Judith Fillmore
Compositor	Graphic Typesetting Service
Illustrator	Diane Keller
Cover Designer	Joe di Chiarro
Text Designer	Judith Olson

Library of Congress Cataloging in Publication Data

Schwar, James P.
Applied FORTRAN for engineering and science.

Includes index.
1. FORTRAN (Computer program language) I. Best, Charles L. II. Title.
QA76.73.F25S34 001.64'24 81-13609
ISBN 0-574-21365-1 AACR2

Printed in the United States of America.

10 9 8 7 6 5 4 3

Applied FORTRAN

for Engineering and Science

preface

This book is designed for a one-semester introduction to FORTRAN 77 programming for students in engineering and science. FORTRAN, the acronym for FORmula TRANslation, is the prevailing computer language in engineering and science. Since the introduction of FORTRAN in the mid-1950s, the language has undergone several generations of development. An unfortunate consequence of this development has been the introduction of various dialects peculiar to particular machines. FORTRAN 77 is an attempt to update and standardize the language of FORTRAN. It is the standard proposed by the American National Standards Institute (ANSI) and thus can be considered official FORTRAN. To quote ANSI, however, "An American National Standard is intended as a guide to aid the manufacturer, the consumer and the general public. The existence of an American National Standard does not in any respect preclude anyone, whether he has approved the standard or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standard." Hence, although FORTRAN 77 can be considered official, you may encounter some variations of language on your computer.

Chapter 1 introduces the basic concepts of programming and their machine implementation. Chapter 2 discusses data types and data structures, and the internal (binary) representation of data, all of which are helpful in understanding the limitations and rules associated with FORTRAN 77. (Chapter 2 can be omitted, however, without interrupting the continuity of the text.) So that you can start programming as soon as possible, Chapter 3 presents the elementary syntax of FORTRAN. Because formatting of input/output (I/O) is highly formalized in FORTRAN, its discussion is postponed until Chapter 7. Instead Chapter 3 includes what is called *list-directed* I/O. List-directed I/O is machine-controlled; hence, you can concentrate on the logical and syntactical flow of program writing without special regard for the details of form of the input and output. Chapter 4 discusses flowcharting and problem solving. This chapter includes details on such important concepts as

flowchart symbols and programming structures such as SEQUENCE, DOUNTIL, IFTHENELSE, and DOWHILE.

The most important capabilities of a computer are carrying out repetitive calculations and handling large arrays of numbers. The concepts of looping and arrays are closely related. Looping, however, is an important concept in itself. Hence, looping is presented as an idea separate from numerical arrays in Chapter 5, followed by a discussion of numerical arrays in Chapter 6.

Chapters 1, 3, 4, 5, and 6 contain the essentials of FORTRAN programming. A thorough understanding of those five chapters will enable the beginning student to solve a host of computer problems.

Chapter 7 discusses formatted I/O and Chapter 8 discusses the important specification statements: INTEGER, REAL, COMPLEX, DOUBLE PRECISION, LOGICAL, IMPLICIT, DATA, and PARAMETER. The CHARACTER specification statement for the manipulation of character strings is treated extensively in Chapter 8.

Intrinsic functions, user-defined functions, and subprogramming are discussed in Chapter 9.

A special feature of this book appears in Chapter 10, where some applications to numerical methods and matrices are introduced. The computer is of prime importance in solving problems of numerical integration, root search, simultaneous equation, and the like; without it, hundreds of hours would be expended in hand-done calculations.

In learning computer programming you must always remember that the actual *writing* and *running* of programs is essential. For this reason numerous exercises are provided at the end of each chapter. Computer programming is fun as well as a valuable experience in logical problem solving. You will find it a real thrill when your program runs smoothly and efficiently and, above all, provides the correct answers. Remember, a computer will do only what it is told to do!

ACKNOWLEDGMENTS

We and the publisher would like to thank the following reviewers for their helpful comments: Robert H. Dourson, California Polytechnic State University (San Luis Obispo); William E. Lewis, Arizona State University; Leon E. Winslow, University of Dayton; and Charles Wortz, Vanderbilt University. To our colleague at Lafayette College, Dr. Chester Salwach, we give special thanks for his critical eomments during the preparation of this manuscript. Appreciation also goes to three generations of Lafayette College students who labored valiantly in the vineyard doing the exercises and illustrative problems as well as identifying errors great and small. Finally, to Mrs. Elizabeth A. Bullock, who not only typed the manuscript but endured the authors' endless revisions, we say bless you.

J. P. S.
C. L. B.

Applied FORTRAN

for Engineering and Science

contents

chapter 1

Introduction to Hardware and Software

The modern electronic digital computer is a high-speed device capable of serving many users performing varied tasks. *Time-sharing* is the ability of the computer to serve many users concurrently, and *multiprogramming* permits the computer to handle several tasks or programs.

A digital computer can serve many users concurrently through the interaction of the machine hardware (the physical components of the computer) and the software (the programs or instructions that cause the hardware to function in the desired way).

1.1 COMPUTER HARDWARE

Computer hardware consists of the *central processing unit* (CPU) and the *input/output units* (I/O). The CPU can be further subdivided into a *memory unit*, an *arithmetic-logic unit*, and a *control unit*. The components of a CPU are diagrammed in Figure 1.1.

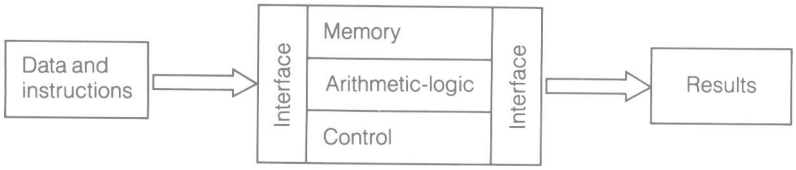


Figure 1.1 Central processing unit

Instructions and/or data are prepared by the user and entered into computer memory through an input device. The control unit decodes these instructions and directs the arithmetic-logic unit to the action to be taken in processing the data. The central processing unit will send the results, output data stored in memory, to an output unit. These instructions, when arranged in some logical sequence, are called a *computer program*.

Information, which may be data in, results (data out), or instructions, enters or leaves computer memory via an interface called a *controller*. The function of a controller is to interface the central processing unit to a specific physical input/output device.

Computer memory is the holding area for instructions and/or data. The arithmetic-logic unit performs actual data manipulations. The control unit decodes instructions; it directs and synchronizes those actions of the arithmetic-logic unit that are required to implement the instruction.

Input/output units are of many types. A widely used input/output device is the *computer terminal*. A terminal has a typewriterlike keyboard for input and either a printing mechanism to produce hard copy or a cathode-ray tube (CRT) for visually displaying output. A typical terminal and its keyboard layout are shown in Figure 1.2.

A computer terminal is commonly used to access a digital computer in an interactive (conversational), time-shared environment. Other input/output devices include magnetic tapes and magnetic disks. These magnetic storage units are capable of holding large amounts of information.

Certain units function only as input (card reader) or output (line printer) devices. Card input and printer output are not interactive. The amount of and speed with which information can be transferred to or from the computer is usually limited by the input/output device itself. Table 1.1 summarizes several types of input/output devices and some of their important operating characteristics.

TABLE 1.1 Input/Output Device Characteristics

Device	Characters per line	Transfer speed
printing terminal	80 to 132	10 to 120 characters/second
CRT terminal	80	10 to 960 characters/second
card reader	80	300 to 1000 cards/minute
line printer	80 to 132	300 to 2000 lines/minute
magnetic tape	user defined	36,000 to 562,500 characters/second
magnetic disk	user and/or system defined	100,000 to 3,000,000 characters/second

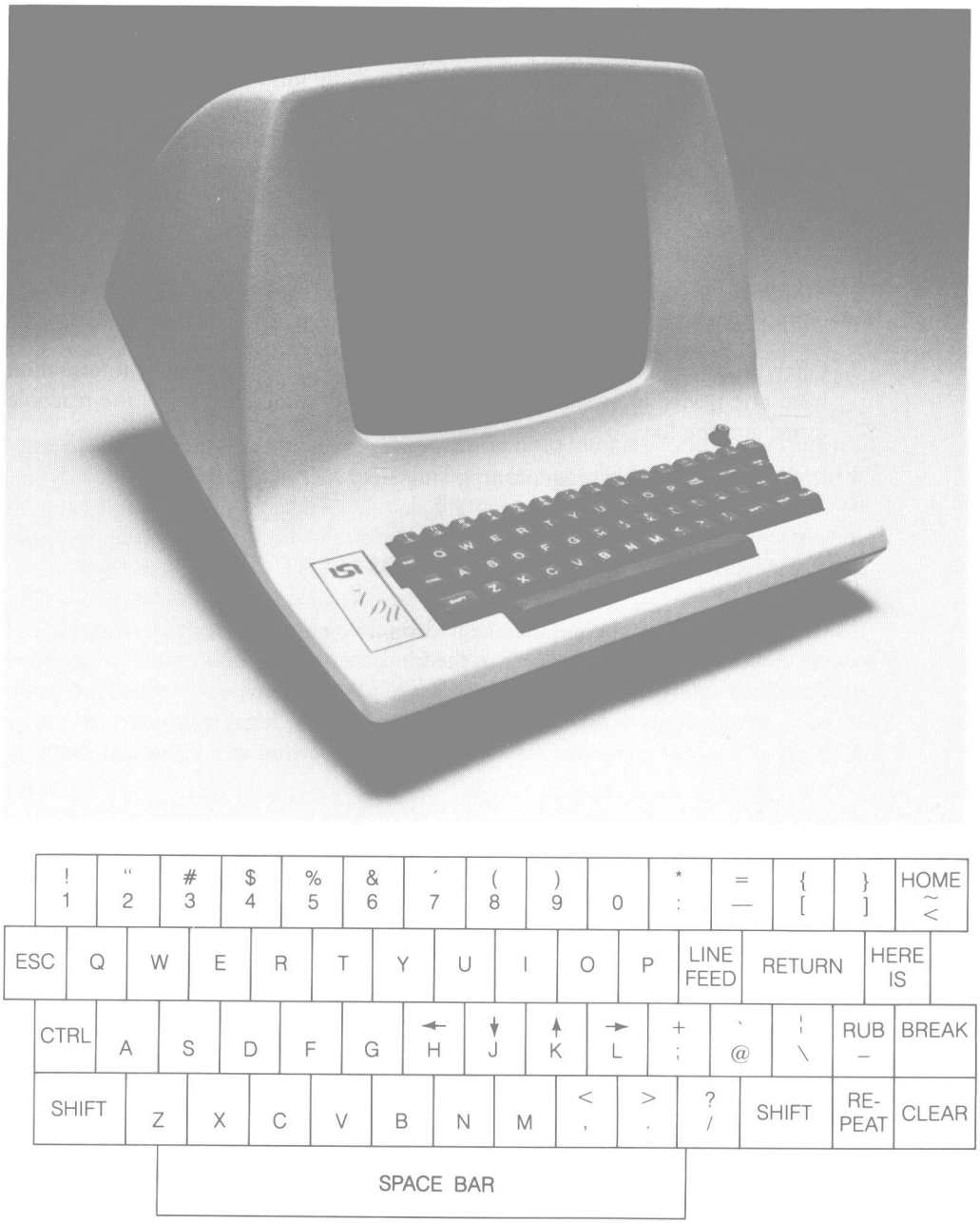


Figure 1.2 The ADM-3A interactive display terminal with a schematic of the standard keyboard *(Photo courtesy of Lear Siegler, Inc., Anaheim, California)*

1.2 COMPUTER SOFTWARE

Computer software is made up of user and system software. It consists of those instructions or programs that tell the computer what is to be done. They reside, along with data, in computer memory. *User software* consists of programs written by the user in a programming language that can be "understood" by the computer. *System software* consists of programs that are "built into" the computer system; these programs:

1. Monitor all system activities, such as computer logon, requests for other programs, execution of programs, and computer logoff.
2. Offer utility operations for listing programs and data, transferring information from one device to another, preparing programs and data in machine-readable form, and the like.
3. Implement computer programming languages such as FORTRAN.

1.3 INSTRUCTION WORDS AND DATA WORDS

The memory of a digital computer is divided into elementary units of storage called *words*. A word of computer memory is the fundamental unit of information accessed and operated on by the arithmetic-logic unit. A word of memory is itself made up of many *bi*nary digi*ts* or bits. A binary digit is a 0 or a 1. Most computers use either a 16-bit or a 32-bit *computer memory word*. The location of a computer memory

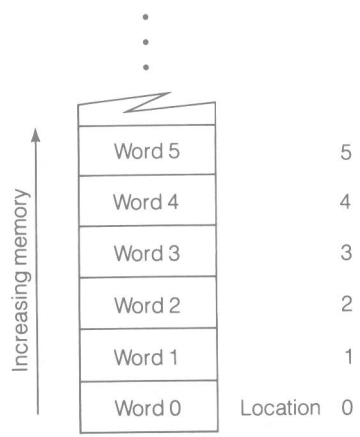


Figure 1.3 Representation of computer memory

word is called its *address*. Computer memory is made up of many contiguous locations; each location is directly addressable and each location is a memory word capable of storing 16 or 32 bits. Addressing starts at zero and proceeds serially upward. The maximum number of memory words available in a computing system is usually a power of 2. For example, 2^{20} or 1,048,576 memory words would be addressable beginning at location zero and ending at location 1048575. Figure 1.3 shows the serial nature of memory storage.

The binary nature of the computer is evident, since all information is stored in computer memory in one or more memory words. A single memory word is made up of a 16- or 32-bit (binary digit) number. This stored information may be either an *instruction word* or a *data word*. A single instruction may require part of a memory word, one memory word, or more than one memory word.

Figure 1.4 shows a partial program and its associated data that have been loaded into computer memory. Locations 1000 through 1002 contain instruction words, one instruction per memory word. These instructions tell the central processing unit to load the data word found at location 3000 into a special hardware register called the *accumulator*. "Add 3001" instructs the processor to add to the contents of the accumulator the data word at location 3001. "Store 3002" stores the contents of the accumulator at location 3002. After these three instructions have been executed, the data word at location 3002 will be decimal 30.

The instruction and data words shown in Figure 1.4 have been represented in an external format that is easy to read. Actually, these words are stored internally as binary numbers. The logical grouping of instruction words, in binary form, is called

Figure 1.4 Representation of machine-language program and data stored in computer memory

a *machine-language program*. Only a machine-language program can be executed by the computer and only binary data can be acted on by the central processing unit. It is necessary to ensure that instruction and data words are properly located in memory so that they are not intermixed or mistakenly used in the wrong context.

1.4 THE CENTRAL PROCESSING UNIT

Figure 1.5 shows the details of a typical central processing unit and its associated registers. A *register*, usually equal in size to a word of memory, holds information such as an instruction word or a data word.

The *data bus* is a pathway for data and instruction words as they move to and from memory. The *accumulator* (A) is a register that stores the data word loaded from memory and to be operated on by the arithmetic-logic unit. The *data counter* (DC) register stores the address or location of the data word that is to be loaded into the accumulator. The *program counter* (PC) register stores the address (location) of the instruction memory word that is to be loaded into the *instruction register* (IR). The IR stores the instruction word that is to be decoded by the control unit and used to direct the operation of the arithmetic-logic unit. Data words can be loaded from memory into the accumulator or from the accumulator into memory. The time needed to access a memory word and load it into the accumulator is in the range of nanoseconds (10^{-9} seconds). The time needed to execute an instruction is in the range of microseconds (10^{-6} seconds). Memory, whose access time is in the nanosecond range and whose words are each directly addressable, is termed *main*

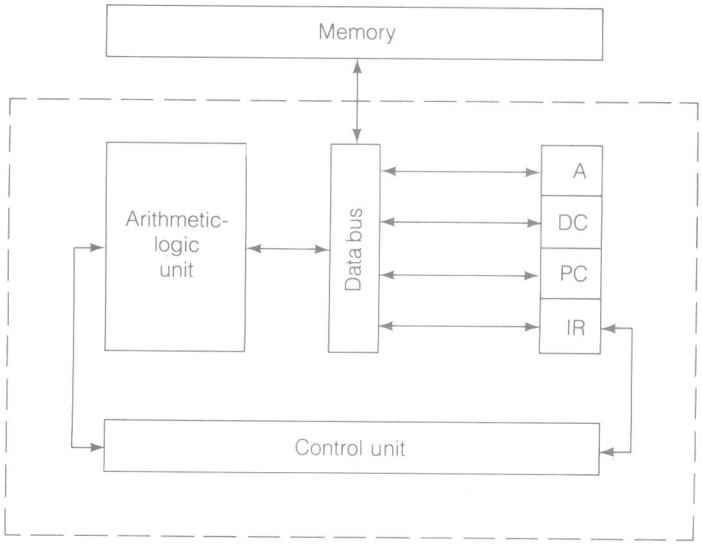


Figure 1.5 Representative CPU with registers