```
0134   RAL    65   0142   0148
0148   AV     10   8001   0105
0105   MULT   19   0108   0189
0189   STV    21   0144   0147
0147   RAV    60   0100   0155
```

```
12F0013   10   12F00
12F0014   01   27G00
12F0015   00   12F00
```

DIGITAL COMPUTER PROGRAMMING

D. D. McCracken

The first general introduction in book form, stressing actual work with computers.

```
0300   9000   .86   0530   1500
0301   9001   .80   9010   (9030)
0302   9002   .11   9010   9013
```

```
fa-all   c25   c16   b03     add
         c16   0     0       set-3
         b03   b11   14b03   div
```

```
01010   A200    REG   A001
01020           SRI   0,200
01030   RDATA   RCD   READER
```

John Wiley & Sons, Inc., Publishers

DIGITAL
COMPUTER PROGRAMMING

Library of Congress Catalog Card Number: 57–8891

Printed in the United States of America

PREFACE

This book is written for the person who needs to know how problems are solved on a modern stored program computer. The person seeking such information in the past has had to rely on printed materials which are directed either toward those who only want to know the end product of computing—what computers can do, how much money they can save, etc.—or toward those who want the details of operation of a particular machine. *Digital Computer Programming* provides a general introduction to the entire field, with emphasis on the basic principles. It is written for people with no previous knowledge of computing who want to know how to prepare the detailed "instructions" for the computer, as well as for people whose work is so closely related to computer applications that they need to know what is involved in programming.

The book begins with a rudimentary discussion of the elements of a computer and their relationships. It presents the fundamental ideas of programming with detailed examples and explanations. These examples are written for a mythical computer called TYDAC, which stands for TYpical Digital Automatic Computer. This "paper" computer is intended primarily as an aid to learning rather than as a compilation of all the features of available equipment. It is generally representative of the major trends in present computer building. The examples are written in a form which makes it possible, if desired, to study them without detailed knowledge of the characteristics of the illustrative computer. The book presents many of the programming techniques which must be known to make efficient use of the equipment, and thus helps to answer the question, "Now that I know how the machine works, how do I solve my problem?"

It is anticipated that the book will be useful, in different ways, to two main groups of readers. Those who read it without having an actual machine to practice with will find that what they learn can easily be applied to a real situation later. This is because the primary concern is not with details and peculiarities of a particular machine, which must be the concern of a machine manual, but with the *principles* of programming. In fact, much of the textual matter is a general discussion of ideas which apply equally to any computer, without direct reference to TYDAC.

Those who have a computer at hand while they read the book will find several desirable features. Possibly the most important is that the chapters which do not apply to a particular situation can be omitted without loss of continuity: at least half of the chapters may be omitted or included at will. This group of readers will find little difficulty in applying the illustrations to their particular machine, partly because of the format of the programs and partly because TYDAC is an uncomplicated machine. In a classroom situation, the instructor can fairly easily rewrite the illustrations. The many exercises are in no way dependent on the features of TYDAC.

Both groups of readers will find that the text is self-contained. If necessary, it may be read without an instructor or reference material, either to provide a general background knowledge of computer programming or as a supplement to a manual.

Practically none of the technical material of the book is original with me. I am indebted to all those who have developed and made available the material presented here. All the computer manufacturers were most helpful in supplying material on their equipment.

I wish to acknowedge my appreciation to P. M. Thompson, W. C. McGee, and Dr. H. R. J. Grosch of General Electric Company, for encouragement in the very early stages of this effort; to F. G. Gruenberger of General Electric, who read the manuscript and made many valuable suggestions; to R. C. McGee of General Electric, who supplied most of the material for Chapter 7; to members of my staff who assisted in the clerical work; and to my wife, for her patience during the writing.

D. D. McCRACKEN

Phoenix, Arizona
March 1957

CONTENTS

1 COMPUTING FUNDAMENTALS

1.0 Introduction

Programming a problem for solution on a digital computer is basically a process of translating from the language convenient to human beings to the language convenient to the computer. The language of the problems to be solved is mathematics or English statements of decisions to be made; the language of the computer is simple arithmetic and elementary choices, expressed in coded numerical form. By and large, we are at present required to present problems to the computer in *its* language.

In order to put the problem in the required form, we must learn in some detail the functions of the various parts of a computer, and the precise manner in which orders are given to the machine. This chapter presents the framework of the subject; later chapters will provide the details. Section 1.1 discusses the over-all picture and defines some of the basic terms. Section 1.2 gives an initial description of the mythical computer used for illustration in the text. The succeeding sections trace the development of present equipment, sketch the steps in computer solution of a problem, and list some typical computer applications.

1.1 Computer Organization

A modern digital computer usually consists of several boxes or racks of mechanical and electronic equipment, connected together by electric cables. In this array we find five distinct functions being performed: *input, memory, arithmetic, control,* and *output*. Figure 1 is a block diagram of these functions, showing the relationships among them.

The input section of a computer ordinarily consists of devices which take information from punched cards, paper tape, or magnetic tape, and place it in memory. In technical language, this is called *reading*. The function of the input device(s) is essentially to translate from the external form in which the information is represented, such as a punched card, to the form in which the same information is stored in memory. The *information* in question may be anything which can be stored in memory: numbers to be used in the calculation, *instructions* which tell the machine what to do, numbers or letters to be used later as column headings on the output, etc. The single arrow from the input box to the memory box in Figure 1 implies that the information goes only *to* memory—further operations must take the information from memory to other sections of the machine.

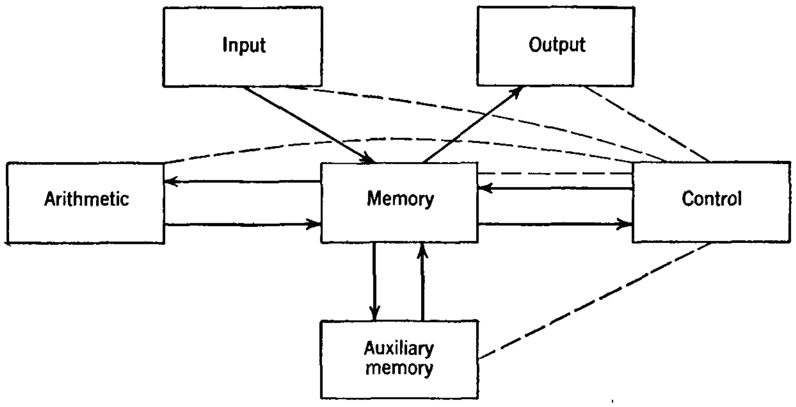

Figure 1. Functional parts of a digital computer and their relationships. The solid lines represent information flow, the dashed lines control signals.

It is difficult to find good analogies between large computers and things more familiar, and the analogies are apt to be misleading. Nevertheless, it may be helpful to characterize the input function as equivalent to the keyboard of a desk calculator. Of course, the difficulty with this analogy is that a desk calculator has no real internal memory.

The memory or storage of a computer is the nerve center of the machine. All information must travel through it. All numbers must be in it before any arithmetic manipulations can be carried out. All the instructions which tell the machine what to do must be in memory before they can go over to the control section. The memory needs to be large and fast, i.e., it should be able to hold many numbers or instructions—from 1000 to 30,000 in present equipment—and be able to send these to the arithmetic or control sections with a minimum delay—as short as about 10 microseconds in the fastest machines at the time of writing. If it is not technically or economically feasible to build a high-speed memory large enough to hold all the information required, a solution is to store the part not currently needed in a larger, but slower, auxiliary device. As indicated in Figure 1, the auxiliary memory "communicates" only with the main memory.

The present trend is for the main memory to be built around magnetic cores in large machines, and magnetic drums in the smaller. Auxiliary memory is almost always magnetic tape, with magnetic drums also being used in the large computers. Electrostatic and mercury-delay storage are still employed in some machines, but are being superseded in the newer ones. Descriptions of the operation of these devices will be found in works listed in the bibliography.

The arithmetic section of the computer does what its name implies. It is here that the actual work of problem solution is done. In addition to the four arithmetic operations, this section can *shift* numbers right and left, and assist in certain operations which make it possible for the computer to make *decisions*. It corresponds in a desk calculator to the wheels and gears and shafts that actually do the calculation.

Register is a term commonly used in connection with several of these basic functions; this is simply a device for temporarily storing a piece of information while or until it is used. A register corresponds quite closely to the dials on a desk calculator, which are wheels that temporarily store the numbers on which arithmetic is done. In our case, it is not only numbers which may be stored in a register but also instructions.

The control section of a calculator has the function of *interpreting* or *decoding* the instructions stored in memory, and then sending signals to the rest of the parts telling them what to do. In the diagram we see two solid lines, implying that instructions are sent to and from (usually *from*) memory to control; the dashed lines imply electric signals sent to the rest of the machine, based on these instructions.

The control section is equivalent to the buttons which are pushed to start the various arithmetic operations on a desk calculator, but the analogy is quite incomplete. The arithmetic and control sections are the hardest functions to point to in looking at a machine. The input and output devices are usually separate frames, as are memory and the magnetic drums and tapes if any. The arithmetic and control sections, on the other hand, are made up of ordinary-looking electronic components, and the equipment constituting the two functions is usually in the same cabinet. Incidentally, there are usually one or more boxes to which no reference has been made here: the power supply. This omission simply points up that we are looking at a computer from the standpoint of what it does and how the information flows, not from the standpoint of electrical engineering.

The output section has the obvious purpose of recording in convenient form the answers to the problem or anything else in memory. The media may be punched cards, printed pages, or paper or magnetic tapes. The chart shows that information may be recorded (or *written,* in the jargon) only from memory. For our purposes this is true, although *electronically* the arithmetic unit may be involved.

The word *instruction,* which has been used repeatedly, should be amplified. Anyone who has used a desk calculator realizes that it is necessary to have some sort of pattern to the operations so that the operator can get into a routine. This pattern consists of a sequence of specified arithmetic operations on specified quantities. Analyzing or breaking down the process further, we see that doing a desk calculation consists of doing a series of distinct steps, each step involving one arithmetic operation and one new piece of information.

The situation in the electronic computer is not so different. For a problem to be solved on a computer, it must be broken down into a series of precise steps, each involving one arithmetic operation and one piece of information (two or three in some machines) in addition to the result of the previous step. The difference between this situation and the desk calculator is that with the desk calculator the sequence of operations is in the operator's head, whereas to satisfy the computer the sequence must be written down in a rigidly defined form. The appearance of these instructions will be elaborated in the next chapter; we may say here that they are usually stored in memory as ordinary numbers.

After defining several more terms, we shall look at the flow of information as a typical instruction is executed. The first term is *program.* A program is simply a collection of instructions which carries out some purpose such as solving a particular problem. We speak of modern computers as being *stored program* machines. As a verb, to program means to write the instructions necessary to tell a computer how to solve a problem, along with the planning necessary before the detailed instructions can be written. The word *code* is sometimes used almost as a synonym for program, but often it implies a lower level of activity which involves a smaller amount of planning. *Word* is used in computing as a generic term to cover either a number or an instruction or a group of characters to be used for some other purpose. It is roughly equivalent to *piece of information* as used previously.

Suppose now as a very simple example that two numbers are to be added. The two numbers, and, in the type of machine to be considered in this book, three instructions, have to be loaded into memory by the input device(s). Actually, many other instructions have to be in memory to instruct the machine to bring these in, but we can without great inconvenience ignore this fact.

The first instruction moves from memory to the control unit, which analyzes the coded instruction to determine what operation is called for and where in memory to locate the first number. After this analysis or interpretation, the control unit sends out signals to the appropriate units, calling for the specified number to move to one of the arithmetic registers in preparation for the next operation. The second instruction is similarly interpreted and the control unit calls for the second of the two numbers to move from memory to the arithmetic unit and be added to the first number. The third instruction sends the sum back to memory. Finally, the sum is written on an output device; this also requires many more instructions, which fact can be temporarily ignored.

In Chapter 2 we shall discuss the same example in terms of the *details* of machine characteristics.

1.2 TYDAC

Much of the material of later chapters will be illustrated by writing codes for TYDAC, which, in the tradition of naming computers by acronyms, stands for TYpical Digital Automatic Computer. This machine is a compilation of representative characteristics of present computers, and of course exists only in this book. This section is a description of the major features of TYDAC, showing the relationship to the material of the previous section.

The input of TYDAC is assumed to include punched cards, a special typewriter, and a paper-tape reading device on the typewriter. The memory is taken to be 2000 words, each holding ten decimal digits and sign. Each word may be either a number to be used in the calculation or a (coded) instruction. No assumptions or statements are made about the physical type (whether magnetic cores, drums, etc.) or the speed of the memory. Four magnetic tapes are assumed as auxiliary memory.

The arithmetic unit comprises two registers: the *accumulator* and the *multiplier-quotient* or *MQ*. The accumulator does all the addition and subtraction and participates in multiplication, division, and most other operations. It can hold eleven digits and sign. The MQ is involved in multiplication, division, some shift operation, and a few others. It holds ten digits and sign. TYDAC is assumed to be able to do floating decimal arithmetic (Chapter 10), i.e., keep track of decimal points during a calculation, if desired.

The control section has four registers: the current instruction register, the location counter, and two registers called index registers. As discussed in the previous section on general computer organization, each instruction from memory has to be placed in the control section before being interpreted and executed. The temporary storage in which each instruction is held after being brought from memory, and while it is being decoded, is called the *current instruction register.* The register which keeps a running record of the "location" in memory of the instruction of current interest is called the *location counter.* (The notion of location in memory has not been discussed yet; it will be clarified early in the next chapter.) The *index registers* have to do mostly with the automatic modification of instructions. Chapter 8 is devoted to their operation and use.

The output equipment of TYDAC is assumed to be punched cards, the special typewriter, and a paper-tape punch attached to the typewriter.

We may now draw a diagram of TYDAC, Figure 2, which is an expansion of the general diagram, Figure 1. The over-all information flow paths are the same as in Figure 1; the details will be presented in succeeding chapters.

1.3 History of Computing

The characteristics of present computers have been arrived at through a process of development, most of which has occurred since 1945. It may be instructive to trace, in broad outline, the course of these developments.

Devices to assist in working with numbers have been in existence as long as there have been numbers. The first was the abacus, which made use of the bi-quinary number system (page 49) some 5000 years before its application in several modern computers. The first mechanical computer was built by Pascal; a better device was built by Leibnitz in 1673. The first large computer was started in 1812 by Charles Babbage, a British mathematician. The machine was called

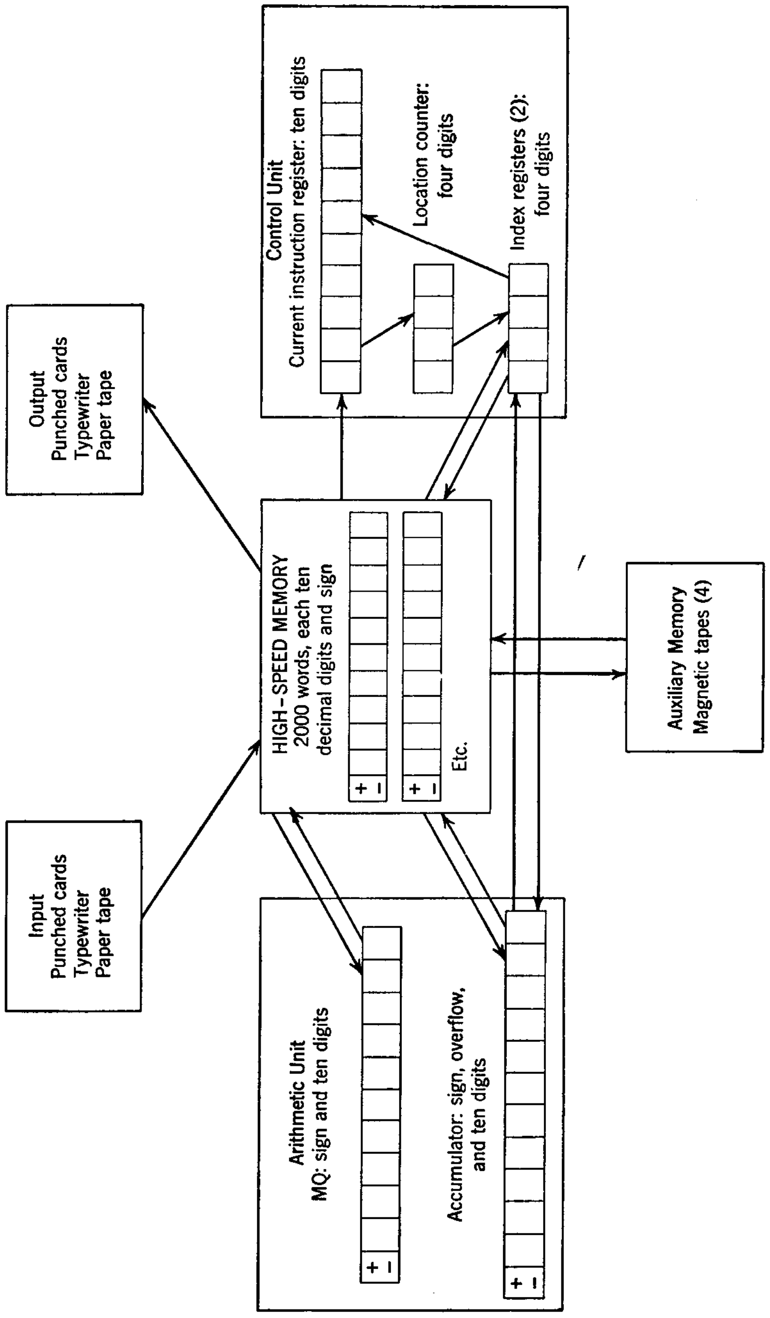

Figure 2. Functional diagram of TYDAC.

the Difference Engine, from the mathematics it employed to calculate tables of mathematical functions. Babbage did not complete his machine, but others built a computer from his plans.

In 1833 Babbage conceived the Analytical Engine, which is the ancestor of all automatic computers. This machine can fairly be called a general-purpose computer, since it was to have flexible sequential control over the arithmetic operations it performed. *Sequential control* means that it was to be possible to specify in advance a sequence of arithmetic operations and the numbers to be operated on. Once the sequence had been specified by a punched card mechanism developed earlier for use on the Jacquard loom, the machine would carry out the operations automatically. The sequence could be changed by altering the punched cards. It was to store numbers in mechanical wheels and use mechanical arithmetic elements. The input was to be either punched cards or hand-set dials, and the output was to be punched cards, a printed page, or a mold from which type could be set. Unfortunately, this brilliant conception was never translated into a working machine, due partly to financial difficulties and partly to engineering problems which were at the time insurmountable.

The present application of punched cards began in 1889 when Dr. Herman Hollerith patented the Hollerith punched card. The equipment he invented and constructed was used in his work for the U. S. Census Bureau, and later became the basis for the International Business Machines Corporation which was organized in 1911.

The first modern machine to use Babbage's principle of sequential control was described subsequently by Dr. Howard Aiken of Harvard University in the 1930's. Called the Automatic Sequence Controlled Calculator, or more commonly the Mark I, it is remarkably similar in principle to the Analytical Engine. It does, however, make use of electromagnetic relays, and uses punched paper tape for sequence control rather than punched cards. It was completed in 1944 after several years' work by Harvard University and IBM. It is still in use.

The ENIAC (Electronic Numerical Integrator and Computer) represented a considerable advance in the computer building technology, since it is entirely electronic in internal operation. Designed by J. P. Eckert and Dr. J. W. Mauchly of the Moore School of Electrical Engineering at the University of Pennsylvania, it was completed in 1946. It was of course much faster than any previous machine. Sequence control is effected by means of many external wires running between holes in plugboards, and by external switches.

Input and output are basically IBM cards, but dials may be used for the input of constants.

All these machines, and others along the same lines, use some external means of sequence control: punched cards, paper tape, wired plugboards. The memory is used only to store numbers. The fundamental idea of placing "instructions" in memory, which is basic to modern computers, did not emerge until 1945. This "stored program" idea, with which we shall have much contact, appeared in a report written by Dr. John von Neumann, proposing a computer quite different from the ENIAC. By storing the instructions internally and by using binary instead of decimal numbers (Chapter 3), much greater power could be achieved at considerably less expense of electronic equipment. The name EDVAC (Electronic Discrete Variable Automatic Computer) was suggested. In a further attempt to reduce the bulk of equipment, the memory of the EDVAC was built around the ultrasonic or mercury-delay type of memory. The EDSAC (Electronic Delay Storage Automatic Computer) was built along similar lines at Cambridge University. It first operated in 1949.

No radically new ideas, of the magnitude of the stored program principle, have appeared in the flood of computers designed and built since these early models. Great advances have been made, however, in speed, reliability, and ease of use.

The Univac, produced by what is now the Sperry Rand Corporation, was the first mass-produced computer placed on the market, in 1951. It is a decimal machine, has magnetic tapes, and uses the mercury memory. It and its successors are in wide use.

The IBM 701 appeared in 1953. It gained speed by using binary numbers and electrostatic storage.

The Whirlwind I, built at the Massachusetts Institute of Technology, was the first large machine to use magnetic cores for main memory. This development represented a gain of a factor of 2 or more in speed, and a great increase in reliability, over electrostatic memory. Production machines using magnetic cores include the Univac II and the IBM 704 and 705.

One of the problems plaguing computer designers for many years has been the great disparity in speed between the input-output equipment and the internal electronic circuitry. Significant advances have been made in improving the reading and writing devices, but no mechanical device can match arithmetic speeds of millions of operations per minute. A solution to this problem, which has been available since about 1954, is the use of separate high-speed tape reading and writing equipment. For instance, on a machine where punched cards are the primary input medium, it is highly uneconomical to tie up the entire machine while reading cards. What can be done is to read the cards in a separate machine and write the information onto magnetic tapes, at the usual card-reading speed; this, of course, while the main computer is doing something else. Then the information now on tape can be read by a tape reader connected to the computer, at the much higher tape-reading speeds. A similar saving can be effected on printing the output. Such equipment is available for the major production computers.

Many further advances are surely forthcoming. The foregoing is an outline of the trends of computers actually on the market, up to the time of writing. Computers now in development are said to be much faster, more flexible, and to have much larger memories.

1.4 Steps in Preparing a Problem for Computer Solution

There are several fairly distinct steps which must be carried out to solve a problem on a computer, some of which have been alluded to in previous sections. These steps are now outlined; details will be given later.

NUMERICAL ANALYSIS

In all but the simplest problems, a considerable amount of work must be done before much *detailed* consideration of the computer is brought in. This is because computers, in a single step, can do only simple arithmetic and make only simple logical decisions. Most scientific and engineering problems are expressed in terms a computer cannot handle directly: integrals, cosines, differential equations, vectors. A numerical method must be found to translate continuous functions to arithmetic: finite difference methods, infinite series, continued fractions, iterative procedures, etc.

Although numerical analysis (and in business problems, procedures analysis) is a highly important part of the computing field, it is outside the scope of this book.

PROGRAMMING

This is a classification which is often merged with the preceding or following. When interpreted strictly, it implies all the planning which comes after analysis and is related specifically to the computer. It involves primarily drawing a flow chart or block diagram (Chapter 7), planning memory allocation (Chapter 11), and planning for careful records of what is done during coding.