The Computer Comes of Age

The People, the Hardware, and the Software

R. Moreau

The Computer Comes of Age

The People, the Hardware, and the Software

R. Moreau

Translated by J. Howlett

The MIT Press
Cambridge, Massachusetts
London, England

English edition © 1984 by The Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

Original edition copyright © 1981 by Bordas, Paris, published under the title *Ainsi naquit l'informatique.*

This book was set in Baskerville by The MIT Press Computergraphics Department and printed and bound by The Murray Printing Co. in the United States of America.

Library of Congress Cataloging in Publication Data

Moreau, René.
The computer comes of age.

(The MIT Press series in the history of computing)
Translation of: Ainsi naquit l'informatique.
Bibliography: p.
Includes index.
1. Computers—History. 2. Programming languages (Electronic computers)—History. I. Title. II. Series.
QA76.17.M6713 1984 001.64'09 83-9424
ISBN 0-262-13194-3

Series Foreword

René Moreau's book inaugurates a new series devoted to the history of computer and data processing. Future volumes will deal with various aspects of the development of systems, hardware, and software, and there will be both general works and specialized monographs. Some works being planned for the series will be of a biographical and even autobiographical nature, and others will concentrate on either a particular development, such as magnetic memory, or the technical history of an industrial company.

The first book in the series is by René Moreau, manager of scientific development for IBM France. He is the author of a general computer-oriented work on the theory of languages (1975) and of computer-based analyses of the language of General De Gaulle and of the language used in the election campaigns by Giscard D'Estaing and Mitterrand.

The history of computers and data processing is a relatively new subject. There are numerous specialized historical presentations within this area, but this is the first book to attempt to make a synthesis of the whole subject. René Moreau has given us a technical history that centers on problems and their solutions, stage by stage, from the beginnings of this subject until the year 1963, when the IBM Series 360 computers introduced a new age in the history of technology.

In this book each of the major concepts and devices is explained or described in its historical context. Accordingly, this book should be of significance for any reader wishing to understand the development of information processing and information theory (or informatics). Rene Moreau has enlarged our understanding of the development of technology and of the role of machines in modern society. All readers will be stimulated by his fresh point of view and his deep insight into the significance of information technology.

I. Bernard Cohen
William Aspray

此为试读,需要完整PDF请访问: www.ertongbook.com

Preface

It is not uncommon to read that the first computer was the MARK I, or the Harvard-IBM machine, or perhaps ENIAC; but it is seldom stated that the IBM Selective Sequence-Controlled Electronic Calculator (SSEC) alone can claim this distinction. There is the same uncertainty in dating the first formulations of the main concepts of what we now call computer science—multiprogramming, operating systems, teleprocessing, programming languages, and so on. There are at least two reasons for this. The definitions of the concepts can vary from one writer to another, and it is often difficult to select from the multitude of facts relating to these concepts the one or the few that indicate the origins of the most significant developments.

It seemed to me that it would be interesting to give an account of the history of the computer and of computer science that, while addressed to the nonspecialist, would define these concepts as precisely as possible and, taking the majority view of the knowledgeable writers, would assign dates to first formulations. Thus the book is not simply a history of one particular technology; it is also an exposition of the main ideas and concepts to which this technology has given rise. As a work of popularization, it may give to the reader who does not work in the field of computers some familiarity with the subject. As a history, it may give the specialist a better feeling for the origins of this subject.

The bibliography of so broad a subject could fill several volumes; the bibliographical entries given here are works actually cited in or having a direct bearing on the text.

Translator's note: Certain conventions have been adopted for this edition. First, author's notes are cited by superscripts 1, 2, 3, . . . ; translator's notes, by superscripts a, b, c, . . . ; both author's and translator's notes appear at the back of the book. Second, text in square brackets is an insertion by the translator.

Before concluding this preface, I should like to thank all those who have given me so much help, whether in reading and criticizing my text or in providing me with documents.

Contents

Introduction

Twenty years ago very few people, apart from the small number actually in the field, had ever heard of computers, let alone seen one. But in 1980 it is estimated that about 50 percent of the venture capital invested during the past few years has gone into computer hardware and software companies. In France, according to the Accountant General (Cour des Comptes) the total value of the computers and associated equipment installed was over $10,000 million. The computer has already invaded most human activities; who among us has never called upon its services, usually without knowing it, whether in running one's washing machine, checking one's wristwatch, or using the automatic cash dispenser at one's bank? Who for that matter has never seen a computer, at least in its physically smallest form, now called the microprocessor, which can be bought in the big stores or the specialist shops for a few hundred dollars? The transformation has been so rapid that it seems to have been brought about by some magician, more or less diabolical, waving a mysterious magic wand.

Twenty years ago, however, the main concepts of computer science had already been formulated, and there is no mystery about this; they were the culmination of a slow intellectual and technological evolution that began at a very early period in the history of mankind. Today's computers, of whatever shape or form, do not differ in essence from those of that time; moreover, the main kinds of uses had already been defined and the main fields of application sketched out. By the end of 1963 or the beginning of 1964 computer science had in some sense come of age.

From the point of view of the early 1980s, the period ending in 1963 can be regarded as belonging to the history of the technology with which this book is concerned. It is now possible, because of the passage of time, to see papers that up to now have been held as confidential, and thus to give a more exact account than had previously been possible of the more recent developments in computer science, still too closely linked with people still alive. This historical period is described here, together with accounts (to put it as simply as possible) of the main concepts forming the skeleton of computer science as we now know it. This history can serve therefore as an introductory course on the subject.

The first chapter deals with the "gestation" period of computer science, which stretches from the earliest times to the appearance of the first computer early in 1948. In this account of the linking of the various concepts and attempts at construction leading to the first calculating devices, then to the true calculators, and finally to the computer, I do not have the space to describe all the abaci, calculating instruments, and other machines that mathematicians, astronomers, and military engineers have built over the years for the evaluation of mathematical functions. I have dealt only with those that contribute directly to the understanding of the historical account. In the second chapter I speak of the "infancy" of computer science. Before it could be considered to be adult, the subject had to grow, and its essential device, the computer, had to develop from the research laboratories where it was born into a manufacturable product. This period lasted from 1949 to 1959; it is not fortuitous that computer specialists when describing the technology of the computer call this period the "first generation." I have chosen that title for this second chapter. The third period, from 1959 to 1963, is that of "adolescence," during which computers started to invade all human activities. It has been called the "second generation" of computers, the title and subject of the third chapter.

This division into three periods can be regarded as corresponding to three main phases of the evolution of the technology: realization of the computer in the first, development of supporting and ancillary equipment for storing information in the second, and during the third the development of operating systems for the control and exploitation of the computer's resources—systems intended for the replacement of the human operator, who was becoming incapable of reacting to the ever increasing speed of the machine.

The same division into three periods corresponds equally well to the three main phases in the development and use of the principal languages of communication between man and machine. Before 1948 the languages used were very primitive; the first of the more highly developed languages appeared during 1948–1959; and 1959–1963 saw the birth of all the languages now most in use. But because the development of languages is linked less closely to that of the machine

此为试读，需要完整PDF请访问：www.ertongbook.com

than is the development of other technologies, I have dealt with this in a separate chapter, the fourth.

The history of this whole period is in every aspect so complex that in the interest of clarity it has been necessary to select particular events, particular pieces of hardware, and particular manufacturers. An attempt to cover everything would have been very difficult and led to a far too bulky book. It was therefore necessary to decide either not to mention this or that machine or manufacturer or not to say very much about them. Since no such choice can be made purely objectively, there will inevitably be some criticisms of those made here. It is not only the choice of facts that poses problems; there is also that of dates, for different dates for the same event can often be found in the literature. There are at least three phases in the development of a computer product: conception, announcement, and installation in a user's premises. I have usually chosen the date of announcement, which has also been done most often in the relevant literature.

1

The Birth of the Computer

Ever since the invention of numbers, humanity has tried to make instruments to help in performing calculations. There were tablets for calculating earlier than 3000 B.C., and the well-known Chinese bead frame existed well before the birth of Christ; but tablets, bead frames, and abaci are all completely nonautomatic. The idea of mechanizing calculation is very old, although it was not a practical possibility until mechanical engineering (whose finest applications had been in the design and making of clocks) became sufficiently highly developed. In the early nineteenth century the British astronomer and mathematician Charles Babbage described what could have been a machine with the ability to perform any calculation whatever; but unfortunately the mechanical-engineering technology of the time provided neither the reliability nor the speed that were necessary for the realization of his dream. The construction of the first calculators had to await the arrival of electromechanical technology, and it was the development of electronics that led to the first computers.

1 Problem Solving and Its Mechanization

If it is to carry out a calculation, a machine must know the route it has to follow. But a machine has no intrinsic knowledge and knows nothing about the external world; therefore it has to be given instructions on how to proceed in the minutest detail. The details of the method of solution of a problem that can be performed by a machine are in the form of a statement of a sequence of operations that the machine has to carry out. We call this the *process*. This way of describing a process is in fact common practice, and we use it whenever we follow what we may call an *algorithmic procedure*.

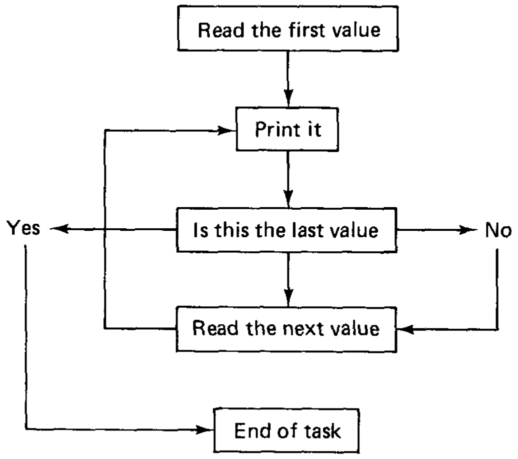

Figure 1.1
Reading a table of values.

Algorithmic Procedure
Among the problems continually presenting themselves for solution are a very large class for which the process of solution can be given in advance in terms of a finite number of exact statements of operations to be used in order to produce a specific result. Figure 1.1 illustrates this type of procedure by showing the steps that must be followed in order to read a table of values. We say that a procedure for solving a problem is *algorithmic* when it can be expressed as a sequence of statements of operations to be performed and when no knowledge or intelligence is required beyond what is strictly necessary in order to perform these operations. The statements must therefore be sufficiently clear and precise as to present no difficulties of interpretation. The diagrammatic representation of an algorithmic procedure is called a *flowchart.*

The statements in an algorithmic procedure are of two types. The first type is illustrated in Figure 1.1 by "Read the first value" or "Go from the part of the flowchart where the operation required is to print the value read to that where the operation is to check whether the last value has been read." These are what we might call imperative commands and we describe such statements as *unconditional.* The second type can be put in this general form: "*If* such-and-such a result is observed, *then* do this *else* do that." Such a statement in figure 1.1 is "*If* the value in the table that has been read is the last value, *then* the task is ended *else* read the next value." These are called *conditional* or *logical* statements.[1]

Thanks to the conditional statement there is the possibility, at any stage in a procedure, of choosing between different routes to follow from there on depending upon the conditions at that stage: for example, whether or not to read another value from the table. A procedure with no conditional statements permits of no variations at all and therefore can be used only to solve a single problem: for example, "Read a value from the table." The inclusion of conditional statements, on the contrary, allows a very much broader class of problems to be solved by making it possible to vary the procedure according to the requirements of the individual problems.

Algorithms

From now on we shall be concerned only with problems for which the process of solution, expressed as an algorithmic procedure, can be given to a machine and followed step by step by the machine until the result is obtained. To put it another way, we shall be studying only those types of algorithmic procedures corresponding to classes of problems that can be attacked with the help of a machine. We shall speak of *operations* to be performed rather than of statements, and of *algorithms* rather than of algorithmic procedures.

In general, an algorithm is constructed for the solution of a problem when it has been found, by reasoning or by experience or as a result of teaching, that a process for the solution can be described as a sequence of operations that can be performed without any need to attach meaning to them. We learn such step-by-step processes for solving certain problems from our earliest school days: for example, the addition or multiplication of numbers, finding square roots, and finding the volume of a sphere. These methods are all algorithms; they all lead from the use of a finite number of operations required to solve a particular problem (for example, the addition of 21 and 33) to their use in solving a broad class of problems (for example, the addition of *any* two numbers).

Algorithms have a long history. Those for addition, multiplication, and division were produced in prehistoric times. The Babylonians are a case in point; they devised algorithms for the solution of decidedly complex arithmetical problems posed by their studies of the movements of the stars and the planets. The word itself, however, comes from the name of the Persian mathematician Abu Ja'far Mohammed Ibn Musa Al Khowarismi, whose writings on arithmetic circa 825 A.D. influenced for centuries the development of mathematics.

Algorithmic Processing

As I have said, carrying out the sequence of operations represented by the statements forming an algorithm requires nothing more than what is necessary to understand these operations; therefore the process is essentially mechanical. Thus if a machine can be built for this special purpose, it will perform better than any human because it will follow the sequence precisely without any variation and will not tire. It is not surprising therefore that the idea of mechanizing algorithms is very old. (Rosenberg [1969], who gives attempts to mechanize algorithms before the year 1000 A.D., holds that the most typical were those made by Gerbert d'Aurillac, who became Pope Sylvester II in 999.)

What characteristics must a machine have in order to carry out such a sequence of operations, which we shall call *algorithmic processing*? At the outset, it must be able to deal with the two types of statement—unconditional and conditional respectively—defined earlier. For unconditional statements the machine must be able to perform the specific operations required by the algorithms; for example, it must have a device that can add two numbers. Such operations or transformations fall into two groups. In one are the operations of transferring information from one part of the machine to another. In the other are the operations having a single *operand*, called *unary operations* (such as the operation of changing the sign of a quantity, the quantity concerned being the operand), those having two operands, called *binary operations* (such as that of adding together two numbers, those numbers being the operands), and, further, operations having more than two operands, although these are much less common.[a] For conditional statements the machine must be able at least to perform simple tests, such as to decide whether one number is greater than another. Such operations, whether unary or binary, are called *logical operations.*

Operations of this type—transfer, unary, binary, logical—are called the *primitive operations* of the process. The elementary arithmetical and logical operations are of very great importance in any algorithmic process, as is easily seen from a consideration of a few examples. To answer the question "What is the sum of the first 10 numbers?," one needs only to make 10 additions—arithmetical operations—testing after each one to find whether that was the tenth. It is less obvious that a yes-or-no type of question can be answered by similar sequences of operations. Consider, for example, the question whether the word "greatness" occurs in a particular piece of writing of General de Gaulle's. To answer this, each word of the text must be given a code number, the same number for each occurrence of the same word and a different number for each occurrence of a different word; the text is thus

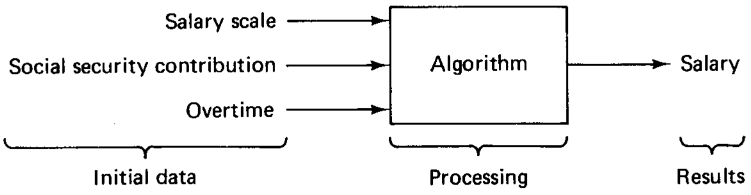

Figure 1.2
Algorithmic processing: calculating salaries.

represented by a sequence of numbers. The machine starts with the first number and subtracts from it the number representing the word "greatness"; if the result is zero, the two numbers are the same and "greatness" has been found as the first word of the text. If not, the machine moves on to the next number in the sequence and repeats the operation, and so on. If it arrives at the end of the text without recording a zero result, then the answer to the original question is no.

A machine that can carry out algorithmic processing must therefore have a number of basic operations. It must be able to perform the fundamental operations of arithmetic, and for this it must have what is called an *arithmetical unit*; similarly, it must have a *logical unit* for the logical operations. The two are often combined in one unit, called the *arithmetical-logical unit* (*ALU*). Finally, the actual performance of the operations as a sequence in time must be monitored and directed, and for this there is a special unit, the control unit.

The effect of any algorithm is to transform the set of elements supplied to it at the start into the set forming the results. For the calculation of a salary, for example, the starting, or initial, set can include salary scale, hours of overtime worked, social security contribution, and so on; Figure 1.2 illustrates this.

The quantities involved in an algorithmic process can be either *numerical*—numbers, financial accounts, statistical observations—or nonnumerical—letters, phrases, pieces of text, and also, for example, music; music as a sequence of sounds can be coded by a function of the amplitude of the sound, measured at regular intervals.[2] Thus algorithmic processing can be applied to an extremely wide range of types of elements. In fact, the fields of application of algorithms are so numerous that one can say that an algorithm can be provided for any problem for which one knows a method of solution. In what follows the word *data* will be used to describe any elements that can be coded and thus subjected to algorithmic processing.

The alphabet used most frequently for coding data inside the machine is the *binary* because this is the easiest to represent in physical equip-

此为试读，需要完整PDF请访问：www.ertongbook.com