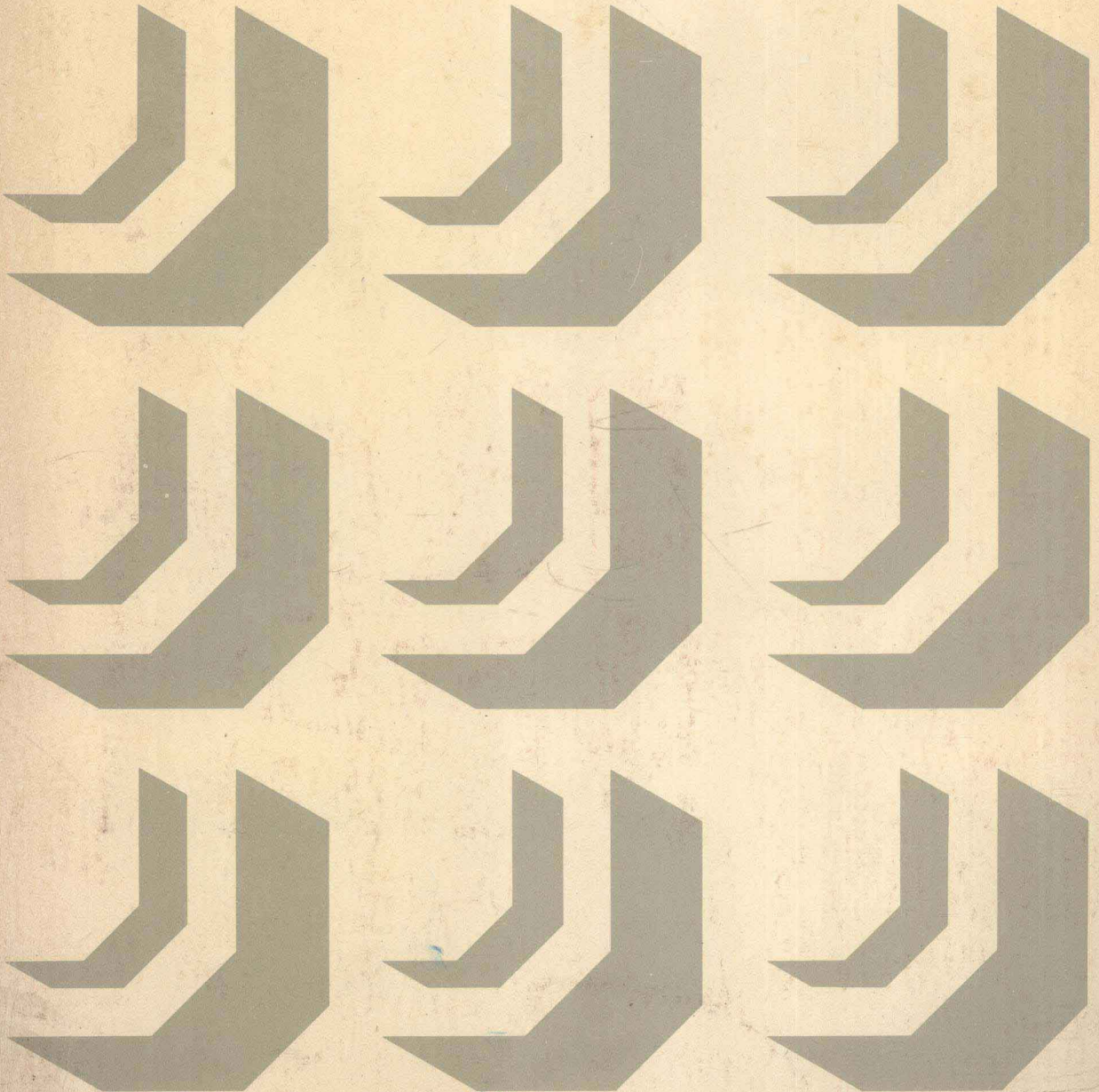


# Assembler Language Programming: Systems/360 and 370

Sharon K. Tuggle



SHARON K. TUGGLE *Vassar College*

---

# Assembler Language Programming: Systems/360 and 370



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

© 1975, Science Research Associates, Inc. All rights reserved.

Printed in the United States of America.

**Library of Congress Cataloging in Publication Data**

Tuggle, Sharon K

Assembler language programming, systems/360 and 370.

Includes index.

1. Assembler language (Computer program language)
2. IBM 360 (Computer)—Programming. 3. IBM 370 (Computer)—Programming. I. Title.

QA76.73.A8T83      001.6'424      74-84276  
ISBN 0-574-19160-7

Appendixes 3, 5, and 6 and form X28-6509 are reprinted by permission of International Business Machines Corporation.

# PREFACE

This text covers Systems/360 and 370 Assembler Language, as used on both OS- and DOS-based operating systems. Where important, the differences in the two families of operating systems are discussed; for example, job control language and input and output instructions. Where possible, the tone of the language used in the text is conversational, allowing the student to interact with the author as if attending a lecture. Questions that are likely to arise in the reader's mind are carefully anticipated, and answers or explanations discussed at that point.

At the end of each chapter is a series of questions that can be used as a self-testing device. Following the questions, most chapters have a series of exercises that can be assigned as formal problems or done as inclass exercises to solidify the information presented. In addition, at the end of most chapters is a quick-reference list for easy location of specific items, terms, or instructions.

The book is divided into three parts: Introduction to Computers and Computer Programming, Introduction to Assembler Language, and Program Management and the Assembly Process. Part I contains an explanation of punched cards, flowcharts, and loops. It provides a good introduction for students taking their first course in programming. (For those who have had some previous computer experience, such as a course in one of the higher-level languages, this part may be reviewed, or skipped all together.)

Part II contains all that a student needs to know to code problems of simple to average complexity. Chapter 3 contains an explanation of the basic components of Systems/360 and 370, chapter 4 explains the representations of data within the machine, and chapter 5 describes how to define storage and con-

stants. Chapter 6 presents the student with his first machine instructions, the move instructions. This is followed by chapters 7 through 10, which cover the three different types of arithmetic instructions—fixed-point binary, packed decimal, and floating point—along with the conversion instructions necessary to make input data ready for calculations and prepare output data for printing. In part II, transfer of control—branches and jumps—is also explained.

If further detail is desired and more time is available, part III contains many topics of interest to the individual who wants to understand more about assembler language and its more advanced capabilities. Topics covered are: the assembler's scanning process and the two passes of the assembler, the handling of absolute and relocatable values, and the fundamentals of addressing. Chapter 14 goes into collecting data into tables and how they are referenced (index registers) and some of the more complicated branch instructions (BAL, BALR, BXH, and BXLE). Chapter 15 covers program sectioning and linking—the whole area of handling subroutines.

Chapter 16 treats the areas of debugging and dump reading. Chapters 17 and 18 cover the remainder of the machine language instructions—those involved in the manipulation of bytes and bits within bytes, and the powerful special-purpose instructions (ED, EDMK, EX, TR, TRT) and the instructions found only on System/370.

The last four chapters, 19 through 23, each contain topics that can enrich any course. Each can be covered in depth or just introduced. The topics are: DSECTS, Job Control Language, Input and Output, macros and the macro language, and virtual storage. There is very little interdependence in part III. These chapters can be discussed in any order or interleaved with the chapters in part II.

The contents of this book comply with Course B2, Computers and Programming; and the majority of Course B1, Introduction to Computing, as set forth in the 1968 Course Curriculum, recommended for Academic Programs in computer science by the ACM Curriculum Committee on Computer Science. It can be used in either a one-semester course or a two-semester course, through expanded use of the topics in part III.

I want to express my sincere thanks and appreciation to all who have been so helpful in the completion of this text: Shirley Mayewski and Pat Caswell, for their patience in typing the manuscript; to my reviewers, whose comments were so helpful in making this a better book: G. G. Casper, Weber State College; Carl Eckberg, California State University; Marilyn Bohl and William Lewis, of IBM; as well as Sallyann Hanson and Eric Weiss; also to Jim Budd, whose attention to detail has made this a better book.

To the computer-center staff at Vassar College, and to my many students, who suffered through the preliminary drafts, I give my appreciation. Most of all, I wish to thank my parents, Eleanor and Cully Krennek, who forged in me the habits and attitudes to accomplish this task, and my husband, Mike, who provided moral support and editorial comment.

S. K. T.

# TABLE OF CONTENTS

## **PART I: INTRODUCTION to COMPUTERS and COMPUTER PROGRAMMING**

### **Chapter 1 Introduction to Computers 3**

- Data Processing Systems
- Standardization
- The Use of Punched Cards
  - Punched Cards as Input to a Data Processing System
- Stored Program Concept

### **Chapter 2 Basic Computer Programming Techniques 12**

- Problem Solving
- Program Flowcharts
- Programming Techniques
  - Program Loops

## **PART II: INTRODUCTION to ASSEMBLER LANGUAGE**

### **Chapter 3 Fundamentals of Assembler Language 29**

- Introduction to Programming Languages
  - Machine Language Programming
  - Symbolic Language Programming
  - Assembler Language Programming
- Basic Components of Systems/360 and 370
  - The Byte as the Basic Building Block
  - Fixed- and Variable-Length Data
  - Registers

- Assembler Language Instruction Format
  - The Four Fields of Information
  - Full-Line Comments
- Types or Classes of Instructions
- The Coding Form
- Overview of the Programming Job

**Chapter 4 Representation of Data**

**46**

- Positional Notation
  - Decimal Number System
  - Binary Number System
  - Conversion of Decimal to Binary
  - Hexadecimal Number System
  - Conversion of Decimal to Hexadecimal
  - The Use of Conversion Tables
- Arithmetic in Different Number Systems
  - Binary Addition and Subtraction
  - Hexadecimal Addition and Subtraction
- Internal Data Representation
  - Character Information
  - Fixed-Point Binary
  - Two's Complement Notation

**Chapter 5 Definition of Storage and Constant Description**

**63**

- Introduction to Defining Constants
  - Examination of the Subfields
  - Character Constant (C)
  - Fixed-Point Constants (F, H)
- Submitting Your First Job to the System
  - Necessary Job Control Language
  - Information Available in a Listing
  - Syntax Error Examples
- Introduction to Defining Storage Areas

**Chapter 6 Information Move Instructions**

**87**

- Three Basic Groups of Instructions to Move Data
  - Register-to-Register Move Instruction
  - Instructions to Move Data Between Registers and Storage
  - Storage-to-Storage Move Instructions
- Submitting Your Job for Assembly and Execution

**Chapter 7 Fixed-Point Binary Arithmetic Instructions**

**107**

- General Requirements for Binary Arithmetic Operations
- Binary Addition and Subtraction
- Binary Multiplication and Division

**Chapter 8 Arithmetic Conversions**

**123**

- Introduction to the Input and Output of a Data Processing System
  - Zoned-Decimal Format
  - Packed-Decimal Format



- The Input Conversion Process
  - The Pack Instruction (PACK)
  - The Convert to Binary Instruction (CVB)
- Statements Necessary for Providing the Input Function
- Preparing to Write Output
  - Convert to Decimal Instruction (CVD)
  - The Unpack Instruction (UNPK)
  - The Move Zones Instruction (MVZ)
- Statements Necessary to Perform the Write Operation

**149**

## **Chapter 9 Packed-Decimal Arithmetic**

- Advantages
- The Packed-Decimal Format
- General Structure of the Decimal Instructions
- Decimal Constants (P)
- The Decimal Instructions
  - Zero and Add Instruction (ZAP)
  - Add and Subtract Decimal (AP and SP)
  - Multiply Decimal (MP)
  - Divide Decimal (DP)
- Printing the Results of Decimal Arithmetic Operations

**160**

## **Chapter 10 Floating-Point Arithmetic**

- The Floating-Point Format
- Floating-Point Constants
- Floating-Point Instructions
  - Normalization Process
  - Unnormalized Addition and Subtraction
- The Halve Operations

**174**

## **Chapter 11 Transfer of Control: Branches and Jumps**

- Introduction to Condition Codes
- Deciding Among Alternate Groups of Instructions
  - Extended Mnemonics
  - The Overflow Condition
- Repetition of Groups of Instructions
  - Loop Structure
  - Flowchart Examples of Loops
  - Programming the Decision for Exit from the Loop
- Setting the Condition Code Using the Compare Instructions
  - Algebraic Compares
  - Decimal Comparison
  - Floating-Point Comparisons
  - Logical Comparisons
- Other Instructions that Set the Condition Code
  - The Load and Test (LTR)
  - Floating-Point Load and Tests (LTDR and LTER)
- Testing the Condition Code Using the BC Instruction



Testing the Condition Code Using the BCR Instruction  
Introduction to the Load Address Instruction (LA)  
The Branch on Count Instructions (BCT, BCTR)

### **PART III: PROGRAM MANAGEMENT and THE ASSEMBLY PROCESS**

<b>Chapter 12 The Assembler and the Assembly Process</b>	<b>209</b>
The Assembler Program	
The Assembler's Scanning Process	
Introduction to the Assembler's Two Passes	
The Location Counter	
The Symbol (Cross-Reference) Table	
The First Pass in Detail	
The Second Pass in Detail	
Handling and Addressing Operands	
Absolute Values—Self-Defining Terms	
Symbols: Relocatable and Absolute	
Relocatable Values: Location Counter References	
Relocatable Values: Literals	
The Evaluation of Expressions	
<b>Chapter 13 Fundamentals of Systems/360 and 370 Addressing</b>	<b>232</b>
The Concept of Addressing	
The Need for Relocation	
The Means for Relocation	
The Process of Establishing Addressability	
The Programmer's Responsibilities	
Fulfilling the Programmer's Responsibilities	
Execution Time, Calculation of Effective Addresses	
Using the FIRST and LAST Macros	
<b>Chapter 14 Data Structures, Looping, and Address Modification</b>	<b>245</b>
Referencing Collections of Data	
Introduction to Tables	
Using Index Registers in Address Modification	
Search Reference and Direct Reference Tables	
Search Reference Tables	
Direct Reference Tables	
Explicit Addressing	
Special Uses for Instructions	
The LA Instruction	
The BCTR Instruction	
Special Branching Instructions	
The Branch and Link Instructions (BAL, BALR)	
Branch on Index Low or Equal (BXLE)	
Branch on Index High (BXH)	
Introduction to Chained Lists, or Queues	
Construction of a Chained List	
Adding and Removing Data Items	
Double-Threaded Chained Lists	

<b>Chapter 15 Program Sectioning and Linking</b>	<b>275</b>
Introduction	
The Terminology Used	
Need for Standard Linkage Conventions	
The Use of Save Areas	
Register Usage	
Differences Under DOS-Based Systems	
Communications Between Control Sections	
Branching Tables	
Symbolic Linkage	
Passing Control and Handling Parameters	
Review of Subroutine Responsibilities	
<b>Chapter 16 Dumps and Debugging</b>	<b>289</b>
Importance of Debugging	
Three Times for Debugging	
Desk Checking	
Assembler Diagnostics	
Errors During Execution	
The Interrupt Procedure	
Information Found in a Dump	
Locating Cause of an Error at Execution Time	
Debugging Errors in Logic	
Different Types of Dumps	
The PDUMP Macro (DOS-Based Systems)	
The SNAP Macro (OS-Based Systems)	
<b>Chapter 17 Coding at the Bit and Byte Level</b>	<b>319</b>
Logical Operation on Fullwords	
The Meaning of the Word Logical	
Arithmetic on Unsigned Numbers	
Fullword Logical Compare Instructions	
Character or Byte Manipulation	
Byte Transfer or Move Instructions	
Comparison of Characters	
Bit Manipulations	
The Shift Operations	
Bit Manipulations Based on the Functions of Logic	
The Test Under Mask Instruction	
<b>Chapter 18 Powerful Special-Purpose Instructions</b>	<b>334</b>
Editing of Output Fields	
The Edit Instruction (ED)	
The Edit and Mark Instruction (EDMK)	
The Execute Instruction (EX)	
Translation of Character Strings (TR)	
Character Scan or Search (TRT)	
Instructions Available Only on System/370	
The Move Long Instruction (MVCL)	
The Compare Logical Long Instruction (CLCL)	
Character Operations Executed Under Mask (CLM, ICM, STCM)	

<b>Chapter 19 Additional Facilities of the Assembler</b>	<b>363</b>
Controlling the Assembler Program	
Listing Control	
Changing Statement Formats	
Alteration of the Location Counter Setting (ORG)	
Pooling Literals (LTORG)	
Use of Multiple Base Registers	
Control Section Longer than 4096 Bytes	
Dummy Control Sections (DSECT)	
<b>Chapter 20 Job Control Language</b>	<b>372</b>
Structure of a Job	
Job Control Statements for DOS-Based Systems	
Using Tapes on DOS-Based Systems	
Label Processing	
Using Direct-Access Storage Devices on DOS-Based Systems	
Job Control Statements under OS-Based Systems	
The JOB Statement	
The EXECute Statement	
Using Cataloged Procedures	
The Data Definition Statement (DD)	
Overriding Statements in Cataloged Procedures	
Using Standard Cataloged Procedures	
The Linkage Editor	
OS-Based Systems	
DOS-Based Systems	
<b>Chapter 21 Input and Output through the Operating System</b>	<b>399</b>
Generalities about Data Sets	
Describing Physical Characteristics of Data—DOS	
Describing Sequential Data Sets Under DOS	
Describing Physical Characteristics of Data—OS	
Opening and Closing a Data Set	
Accessing Records in a Data Set	
Comprehensive Examples	
<b>Chapter 22 Macros and the Macro Language</b>	<b>414</b>
Introduction	
Concept of Macros	
Elements of Macro Definitions	
Placement of the Definition	
The Prototype Statement	
Positional and Keyword Parameters	
Conditional Assembly Features	
Sequence Symbols	
The Branching Statements	
Exit From a Macro Definition	
Concatenating Symbolic Parameters	

Use of Assembler Attributes	
Set Symbols	
Defining Set Symbols	
Assigning Values to Set Symbols	
Adding Macro Definitions to a Library	
<b>Chapter 23 Coding for Virtual Storage</b>	<b>437</b>
Why Virtual Storage?	
The Concept of Relocation	
Types of Relocation	
Static Relocation	
Dynamic Relocation	
Virtual Storage	
Coding Practices for Virtual Storage	
 <b>APPENDIXES</b>	
<b>Appendix 1 The FIRST, LAST, MREAD, and MRITE Macros</b>	<b>453</b>
<b>Appendix 2 Answers to Selected Chapter Exercises</b>	<b>460</b>
<b>Appendix 3 System/370 Reference Summary</b>	<b>468</b>
<b>Appendix 4 Possible Program Interruptions</b>	<b>475</b>
<b>Appendix 5 Comparison of Assemblers</b>	<b>477</b>
<b>Appendix 6 Hexadecimal and Decimal Fraction Conversion Table</b>	<b>481</b>
 <b>GLOSSARY</b>	<b>483</b>
 <b>INDEX</b>	<b>503</b>

# **Introduction to Computers and Computer Programming**



# Introduction to Computers

Since the beginning of time, man has been forced to adapt to an ever-changing environment. Today's environment is changing rapidly, in part because of the development and expanded uses of the electronic digital computer. And today, as often in the past, some people fear that advances in technology will encroach on their friendly, familiar environment. Man's first reaction to a new and powerful invention is frequently fear and apprehension. He fears for his job, his safety, and his security. It is understandable, then, why many people fear the computer.

And yet the computer, when reduced to its simplest components, is easy to understand. It is made up of relatively uncomplicated parts. Men build into the computer a language—that it understands and follows; to communicate with the computer, we need only to learn that language. Anyone knowing the computer's language can tell it exactly what to do, step by step. Built into the computer is the ability to perform any number of operations when given a sequence of orders. Change the sequence of orders, and the computer performs a different sequence of operations.

Give the computer no orders and it just sits there—an inanimate box of wires, circuitry, and switches. It needs you, the programmer, to tell it what to do.

Note: The remainder of this chapter, as well as all of chapter 2, may be review if you can already program in a higher-level language such as FORTRAN or PL/1. If so, you may wish to skim them.



## DATA PROCESSING SYSTEMS

A *data processing system* is a network of machine components capable of accepting information, processing this information according to plan (a program), and producing the desired results. Regardless of what type of equipment is used, all systems perform the same five basic functions:

Input	makes data available to the system
Storage	provides devices into which data can be entered and held and from which it can be retrieved at a later time
Control	the order for performing basic functions
Processing	arithmetic operations or other manipulations on data
Output	the results

One example often used to illustrate the use of these terms is the preparation of an electric bill. Each customer of an electric company has a meter that registers the amount of electricity he uses. At periodic intervals a company representative reads the meter. This reading and the one taken for the previous billing period, make up the input. These two readings are stored, and the results of any calculations produce the customer's bill. The processing of data involves:

1. subtracting last month's meter reading from this month's reading to determine the amount of electricity used
2. multiplying the amount of electricity used by the rate charged for it
3. calculating any taxes or discounts that may be applicable

The output is the entry into the company's accounting records and the customer's bill. The control function is performed throughout the entire operation. It determines the sequence of the other functions. It ensures that subtraction is performed before multiplication by the rate, and it sees to it that taxes and discounts are applied before producing the bill.

The data processing system described above might involve only human processors, human and mechanical processors (such as adding machines), or human, mechanical, and electronic processors (computers). If only human processors are used in the system, the brain of a human being controls the entire operation. As long as the human can read the handwriting of the meter reader and has access to the accounting records, he can take the input, perform the proper calculations in his head or on a piece of paper, and write out a bill.

If the electric company expands, so that it is no longer feasible to perform all the basic functions by hand, the company could purchase adding machines. If, as time passes, the area the company services becomes a booming metropolis, hiring more people and buying more adding machines becomes impractical. The decision is then made to introduce a computer and let it perform the repetitive task of processing mountains of bills.

When a human being is processing data by hand, there is a great deal of latitude in the inputs and outputs that can be interpreted and produced. However, while input numbers scribbled on scraps of paper are fine for human processors, they have no meaning to a computer. A computer requires *standardization* of its input.

## STANDARDIZATION

The data used in conjunction with nonhuman processors, computers in particular, needs to be standardized in two ways:

1. standardization of the medium on which the data is recorded
2. standardization of the method by which data is recorded on the medium

The medium must be standard in size, quality, and composition. The basic and most frequently used medium in the data processing industry is the *Hollerith* punched card, which is shown in figure 1-1. It is rectangular in shape and measures  $7\frac{3}{8}$  inches by  $3\frac{1}{4}$  inches (18.6 by 8.3 cm). The corners of the card may be squared, rounded (to prevent wear), or cut off (to make sure that no cards in a deck are upside down or backwards).

Rectangular holes may be punched into a card. There are 12 horizontal rows and 80 vertical columns on the card, thus 960 positions where a hole may be punched. The rows are numbered 12, 11, 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 from top to bottom on the card. On an IBM punch card, the 80 columns may be identified by column numbers in two locations: at the bottom of the card and just below the zero row. The numbers 0–9 fill each of the 80 positions in each respective row. Punches in any of these 10 rows are referred to as *numeric*

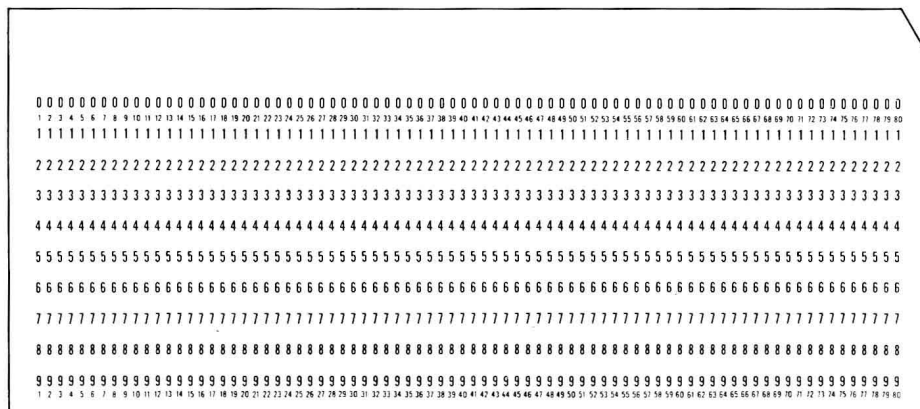


Fig. 1-1 The 80-column punch card