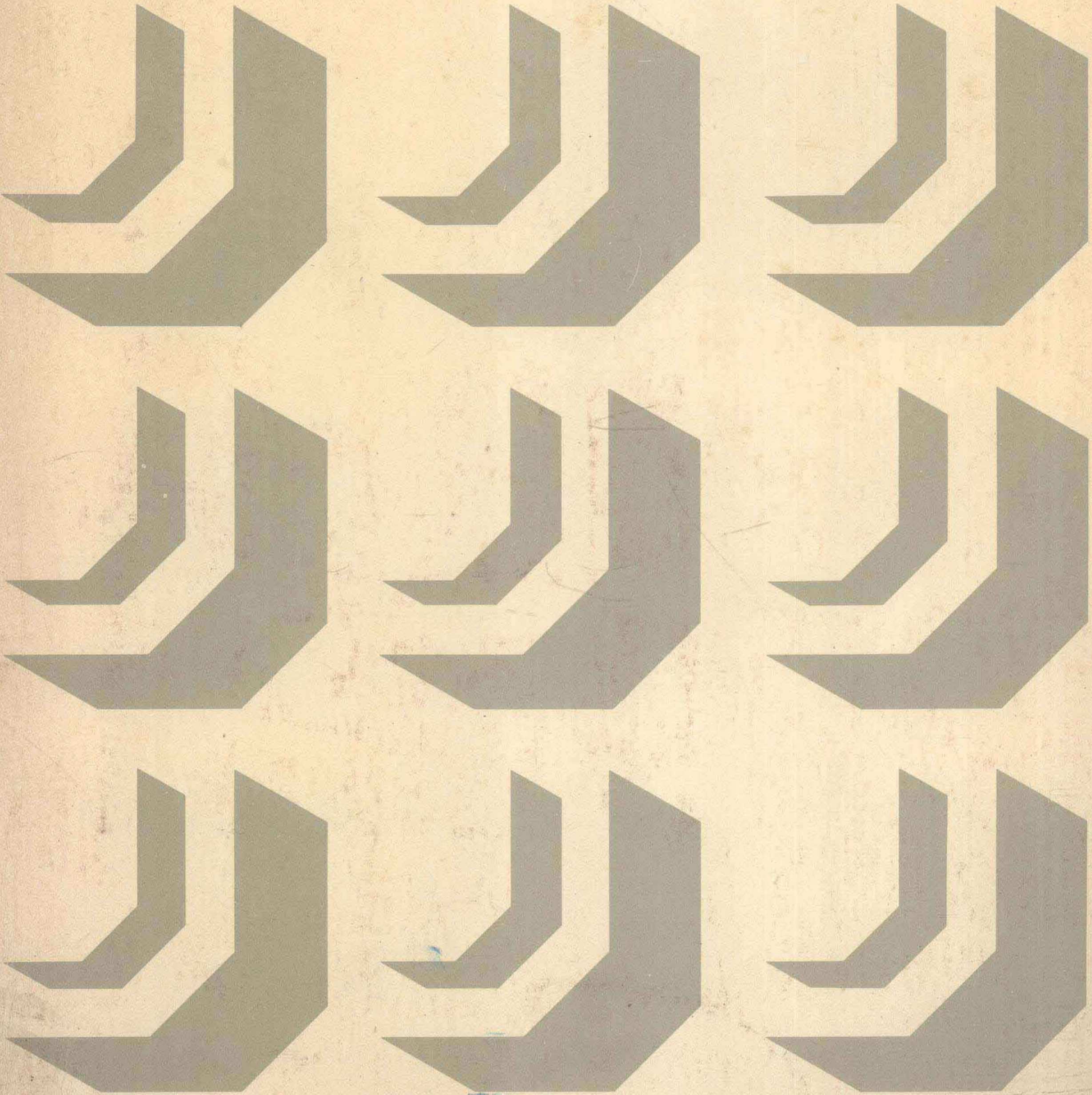
Assembler Language Programming: Systems/360 and 370

Sharon K. Tuggle



SHARON K. TUGGLE *Vassar College*

Assembler Language Programming: Systems/360 and 370

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

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

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 constants. 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 Krenek, 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

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*

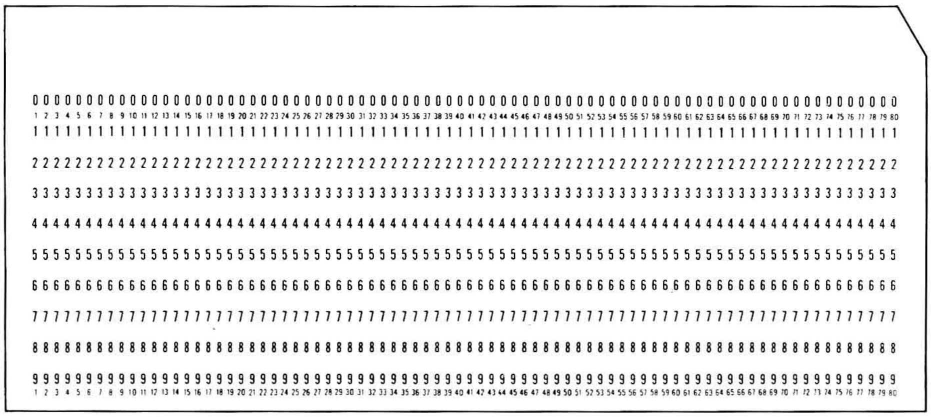


Fig. 1-1 The 80-column punch card