THIRD EDITION

ASSEMBLER LANGUAGE PROGRAMMING

The IBM System/370 Family

GEORGE STRUBLE

THIRD EDITION

ASSEMBLER LANGUAGE PROGRAMMING

The IBM System/370 Family

GEORGE STRUBLE
WILLAMETTE UNIVERSITY

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts ■ Menlo Park, California
London ■ Amsterdam ■ Don Mills, Ontario ■ Sydney

SPONSORING EDITOR
James. T. DeWolf

PRODUCTION MANAGER
Herbert Nolan

PRODUCTION EDITOR
William J. Yskamp

TEXT AND COVER DESIGNER
Maria Bergner Szmauz

COVER ILLUSTRATOR
Tom Norton

ART COORDINATOR
Loretta M. Bailey

MANUFACTURING SUPERVISOR
Hugh J. Crawford

This book is in the Addison-Wesley Series in Computer Science.

Michael A. Harrison
Consulting Editor

Library of Congress Cataloging in Publication Data

Struble, George, 1932–
Assembler language programming.

Includes bibliographies and index.
1. Assembler language (Computer program language)
2. IBM 370 (Computer)—Programming. I. Title.
QA76.73.A8S87 1984 001.64'24 83-17912
ISBN 0-201-07815-5

DEFGHIJ-DO-898765

ASSEMBLER LANGUAGE PROGRAMMING

The IBM System/370 Family

PREFACE

Assembler languages occupy a unique place in the computing world. Because most assembler language statements are symbolic of individual machine language instructions, assembler language programmers have the full power of the computer at their disposal in a way that users of other languages do not. Because of the direct relationship between assembler language and machine language, assembler language is used when high efficiency of programs is needed, and especially in areas of application that are so new and amorphous that existing problem-oriented languages are ill-suited for describing the procedures to be followed. Assembler language programming has particular advantages in character and bit-operation tasks.

This is one of the reasons for studying assembler language. Another is that assembler language is a great vehicle for learning the structure and organization of the computer. Beyond that, a study of how to program standard—and unusual—tasks in assembler language gives us new insight into how the computer must do its tasks when we write our programs in higher-level languages such as Pascal and FORTRAN. With that insight, we know what is quick and what is time-consuming, and can make very good guesses at the computer's behavior in somewhat unusual situations. This makes us much more effective as programmers in our higher-level languages.

This text couples the study of assembler language programming and programming techniques to the study of a particular family of computers, which we call the *IBM System/370 family*. This family includes not only the several models of IBM System/370, and their predecessors the IBM System/360 models, but the IBM 4300 series, the IBM 3030 series, and the IBM 3080 series. These computers are all compatible in executing user programs written in assembler language. There are a few differences; almost

all of the differences are new features that can be used on newer models and we try to point these out. But we can take the programs written for the IBM System/360 fifteen years ago and run them without change on all of the newer models. This gives us some reason to expect that the assembler language (and machine language) features we learn to use now will still be applicable several years from now.

The structure and organization of the IBM System/370 have become an industry standard, and the same assembler language is also applicable to a number of "plug-compatible" computers produced by other companies, including Amdahl, Cambex, Control Data, IPL Systems, Magnuson, National Advanced Systems (NAS), and Trilogy. This text is therefore also applicable to the use of those computers, though there are some differences in the features used by the operating systems, which we explore in Chapter 17.

Study of assembler language *must,* by the nature of the language, be relative to *some* machine, and some particular machine, at that. The IBM System/370 series is a good choice for two principal reasons: (1) computers of the IBM System/370 family (including its plug-compatible competitors) are in use by the thousands, so thousands of computer users will find a study of the specifics of this computer immediately applicable; (2) the IBM System/370 exemplifies features of many other current computers, so concepts and techniques learned with respect to this computer will be applicable to other computer systems as well.

The aim of this text is therefore to introduce the detailed structure of the IBM System/370 and its instruction repertoire, programming in assembler language for this computer, and techniques useful in the applications of computers, especially those more easily programmed in assembler language than in higher-level languages.

It is assumed that you are already familiar to some extent with programming, probably in FORTRAN or Pascal. The particular language of your previous exposure is unimportant; what matters is experience in analyzing a problem, and in developing an algorithm for a computer solution. Many basic concepts are reviewed or presented in a framework that supports my development of further material, so with a basic background you should find this book suitable for individual study.

The book is intended primarily for use as a class text, in a "second" course in computing (though a teacher who wishes to use the book as an introductory text should be able to do so, with a reasonable amount of supplementary explanation). It may be used in a course in assembler language programming. It may also be used in conjunction with a more general or theoretical text on computer organization or information structures or as a stand-alone text. There is more than enough material for a semester, and sufficient opportunity for an instructor to omit or substitute other material. The text is also flexible in that the order of presentation of material after Chapter 11 can be rearranged at will, and parts of Chapter 12 can be introduced earlier without any difficulties.

The most frustrating problem for teachers—and students—of assembler language for IBM System/370 is that a student must learn a fairly large amount of information before submitting a first lab exercise to the computer. It is a problem inherent in teaching powerful and flexible systems. In this edition we introduce two facilities that make the job easier for the student. One is the ASSIST system, which is designed to be helpful to students with easy input, output, data conversion, linkage, and debugging facilities, and manages also to be significantly more efficient of computer time than the standard OS. Both OS and DOS versions of the ASSIST system are available to educational and commercial computer installations; for product and distribution information, write to

Program Librarian
214 Computer Building
Pennsylvania State University
University Park, Pa. 16802

The second facility new to this edition is CMS (Conversational Monitor System); CMS also is easier to use than the standard batch system, partly because it is interactive, but also because of some of its easy-to-use input, output, and debugging facilities.

Also available (but too new to include in this book) is ASSIST/I, an interactive system that interpretively runs IBM System/370 family programs on a variety of microcomputers and minicomputers. The ASSIST/I system is complete with full-screen editor and interactive debugging; it is available for IBM PC computers with MSDOS, UNIX systems, VAX VMS systems, and a number of other systems. ASSIST/I, or more information about it, is available from

Overbeek Enterprises
P.O. Box 726
Elgin, Ill. 60120

We are not comprehensive in our descriptions of either ASSIST or CMS, but in Chapter 5 we introduce enough information about each so that students can get started on their own programs. The second edition of this text mentioned READATA and PRINT subroutines, and had an appendix on linkage between assembler language and FORTRAN routines. Copies of READATA and PRINT and the appendix are still available from me to instructors.

Chapter 1 introduces the structure of a computer, especially the structures of the IBM System/370 and the ways information is represented in them.

Chapter 2 is an introduction to the machine language of the IBM System/370, and Chapter 3 follows immediately with the introduction of assembler language. Throughout the remainder of the book assembler language is used, though you should understand the machine language produced by the assembler language.

Chapter 4 begins the study of the characteristics and uses of individual IBM System/370 instructions, with the binary integer arithmetic and information move instructions.

Chapter 5 introduces a variety of topics—what they have in common is that they should be sufficient to get you started on the computer. Definition of constants and storage areas, assembler control statements, and register conventions are important here. Chapter 5 also shows how to use ASSIST and CMS.

Chapter 6 consolidates some of the Chapter 5 topics by discussing subroutine linkage in depth and introducing the machine language instructions that perform data conversions. Portions of Chapters 5 and 6 may be omitted or postponed, depending on what facilities are available to you.

Chapter 7 introduces control structures and their implementation in assembler language. Our approach to structured programming is to write Pascal-like pseudo-code structures in a standard form in remarks statements, and then follow the pseudo-code structures quite closely in the assembler language. We show easy and standard ways of implementing the structures through assembler language; the result is quite readable programs, with the added benefit that following the pseudo-code structures yields programs with fewer bugs. Chapter 8 continues with control structures but adds address modification as well.

Chapter 9 concentrates on debugging, showing debugging facilities of the standard system, ASSIST, and CMS. It also includes a discussion of debugging strategies, and some notes on programming methodology that may help minimize the need for debugging.

Chapters 10 and 11 deal with byte (character) and bit operations. In assembler language these functions are simple, direct, and efficient; this is one of the areas in which the power of assembler language, as compared to most other languages, is realized.

Chapter 12 introduces manipulation of data sets, both within a program by means of the input and output macros, and external to the program by means of job control statements. Basics of the OS operating systems are illustrated and explained.

Chapters 13 and 14 introduce two other modes of arithmetic available in the IBM System/370: floating point and decimal. These chapters should help you to understand what your programs written in higher-level languages do.

Chapter 15 describes five sophisticated and powerful instructions: Translate, Translate and Test, Edit, Edit and Mark, and Execute; these in-

structions are typical of efforts by computer manufacturers to extend the standard instruction repertoire.

Chapter 16 introduces the facility for defining macros in assembler language. This facility is really powerful, and probably the most intellectually exciting in the book.

Chapter 17 introduces virtual storage concepts, program status word formats and manipulation, input and output instructions and channel programming, interrupt handling and the storage protection system—the pieces of computer structure that are used by operating systems programmers, not applications programmers. The chapter is intended as an introductory survey to give the reader some understanding of a part of the computer structure used only implicitly by applications programs; even a little understanding can relieve anxiety and promote more efficient use of the facilities of the operating system.

An important feature of the book is that it discusses a number of common information processing problems, introduces significant computing techniques, and illustrates implementation of these techniques in assembler language. Thus while you are learning to handle certain features of the IBM System/370, you are also learning valuable techniques that are useful in many situations. Among the techniques discussed are generation of pseudo-random numbers, operations on linked lists, binary search, scatter storage and hashing methods, and a sort–merge algorithm.

This book, then, describes thoroughly the instruction repertoire usable by applications programmers of the IBM System/370. Functions reserved to the operating system are introduced, because an applications programmer needs some understanding of what lies behind supervisor functions, and because budding systems programmers must start somewhere too. However, the functions used only in supervisor state are treated in much less detail than the others, and some of the more specialized are not even mentioned. Assembler language is introduced and used extensively, but several advanced features of the assembler language are not mentioned. Instructions for use of facilities of the operating system are even less complete; users must learn more details from appropriate IBM manuals and must get information on the configuration in use at their own computing centers. I have tried to follow the terminology of IBM manuals to facilitate transition from this book to the manuals. It will be helpful if a good set of manuals is available to you for reference while you are studying this text.

I would like to express my deep appreciation to the many people who contributed to the development of both the original and this new edition. Most of the material, of course, is derived from various IBM manuals; the IBM Corporation has been generous in allowing me to use excerpts from the manuals listed in the bibliographies at the end of each chapter, and individuals within IBM have been encouraging and helpful in their explanations of further points. Robert Heilman, John MacDonald, Thom Lane, Michael

Harrison, Tim Hagen, Norman Beck, Sally Browning, Gordon Ashby, Kevin McCoy, Ed Rittenhouse, Jerzy Wilczinski, Darrell Jones, and Gary Bello made valuable suggestions and criticisms. Sharon Burrowes, Jenny Brown, Barbara Korando, and my daughters Jennifer and Laura did a masterful job of typing the manuscript in its several editions.

Finally, I wish to acknowledge the contributions of users of the book who have helped to debug and strengthen the ideas and presentation as they endeavored to learn about assembler language and the IBM System/360 and 370 from preliminary versions and the first edition of this book.

Salem, Oregon
March 1984

G.W.S

CONTENTS

CHAPTER 3

INTRODUCTION TO ASSEMBLER LANGUAGE

CHAPTER 4

INFORMATION MOVE AND BINARY INTEGER ARITHMETIC

CHAPTER 5

WRITING A COMPLETE PROGRAM

CHAPTER 6

CONVERSIONS AND SUBROUTINES

CHAPTER 7

ELEMENTARY CONTROL STRUCTURES

CHAPTER 8

LOOPING AND ADDRESS MODIFICATION

CHAPTER 9

DEBUGGING

CHAPTER 10

CHARACTER OR BYTE OPERATIONS

CHAPTER 11

BIT OPERATIONS

CHAPTER 12

INPUT AND OUTPUT THROUGH THE OPERATING SYSTEM

CHAPTER 13

FLOATING-POINT ARITHMETIC

CHAPTER 14

DECIMAL ARITHMETIC

CHAPTER 15

TRANSLATE, EDIT, AND EXECUTE INSTRUCTIONS

CHAPTER 16

MACRO DEFINITION AND CONDITIONAL ASSEMBLY