



T H I R D   E D I T I O N

ASSEMBLER  
LANGUAGE  
PROGRAMMING

The IBM System/370 Family

GEORGE STRUBLE

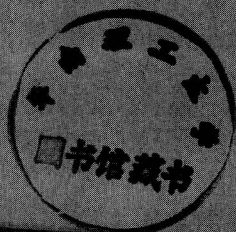
TP312  
S927  
E-3

9760704

T H I R D E D I T I O N  
A S S E M B L E R  
L A N G U A G E  
P R O G R A M M I N G

The IBM System/370 Family

G E O R G E S T R U B L E  
W I L L A M E T T E U N I V E R S I T Y



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

Copyright © 1984, 1975, 1969 by Addison-Wesley Publishing Company, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

DEFGHIJ-DO-898765

---

# ASSEMBLER LANGUAGE PROGRAMMING

The IBM System/370 Family



**A**ssembler 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

# C O N T E N T S

## CHAPTER 1

### INTRODUCTION TO COMPUTER STRUCTURE: THE IBM SYSTEM/370

---

- 1.1 Decimal, Binary, and Hexadecimal Numbers 1
- 1.2 Subsystems of a Stored-program Digital Computer 8
- 1.3 Structure of the IBM System/370 18
- 1.4 Representation of Information 23

MAIN IDEAS 28

PROBLEMS FOR REVIEW AND IMAGINATION 29

REFERENCES 30

---

## CHAPTER 2

### INTRODUCTION TO IBM SYSTEM/370 MACHINE LANGUAGE

---

- 2.1 The Nature of Machine Language 31
- 2.2 Operand Addressing in Machine Language 32
- 2.3 Machine Language Instruction Formats 36
- 2.4 An Example of a Program Segment 41

MAIN IDEAS 43

PROBLEMS FOR REVIEW AND IMAGINATION 43

REFERENCE 44

---

**CHAPTER 3****INTRODUCTION TO  
ASSEMBLER LANGUAGE**

---

- 3.1** A First Look at Assembler Language 46
- 3.2** Format of an Assembler Language Program 47
- 3.3** An Example 49
- 3.4** Addressing of Operands in Assembler Language 54

MAIN IDEAS 61

PROBLEMS FOR REVIEW AND IMAGINATION 61

REFERENCES 63

---

**CHAPTER 4****INFORMATION MOVE AND  
BINARY INTEGER ARITHMETIC**

---

- 4.1** General Structure 64
- 4.2** Information Move Instructions 66
- 4.3** Binary Integer Add and Subtract Instructions 72
- 4.4** Binary Integer Multiplication 74
- 4.5** Binary Integer Division 78
- 4.6** The LM and STM Instructions 81
- 4.7** The LA Instruction 83
- 4.8** Generation of Pseudo-random Numbers 85

MAIN IDEAS 88

PROBLEMS FOR REVIEW AND IMAGINATION 90

REFERENCES 91

---

**CHAPTER 5****WRITING A  
COMPLETE PROGRAM**

---

- 5.1** Introduction 93
- 5.2** Register Conventions 94
- 5.3** Definition of Constants in Assembler Language 95

<b>5.4</b>	The DS (Define Storage) and EQU (Equate Symbol) Statements	104
<b>5.5</b>	Assembler Control Statements	106
<b>5.6</b>	Completing a Program with ASSIST Facilities	108
<b>5.7</b>	Running a Complete Program: Batch Mode and Job Control Language	112
<b>5.8</b>	Running a Program under CMS	116
MAIN IDEAS		120
PROBLEMS FOR REVIEW AND IMAGINATION		121
REFERENCES		123

---

## CHAPTER 6

# CONVERSIONS AND SUBROUTINES

---

<b>6.1</b>	Introduction	124
<b>6.2</b>	The BR and BALR Instructions	125
<b>6.3</b>	An Implied Base Register: The USING Pseudo-operation	126
<b>6.4</b>	Subroutine Implementation	130
<b>6.5</b>	Passing Parameters to a Subroutine	135
<b>6.6</b>	Number Conversions	138
<b>6.7</b>	Examples of Complete Subroutines: Pseudo-random Number Generation	142
MAIN IDEAS		146
PROBLEMS FOR REVIEW AND IMAGINATION		147
REFERENCES		151

---

## CHAPTER 7

# ELEMENTARY CONTROL STRUCTURES

---

<b>7.1</b>	The Program Status Word and the Condition Code	153
<b>7.2</b>	Setting the Condition Code	153
<b>7.3</b>	The Compare Instructions	156
<b>7.4</b>	The BC and BCR Instructions	157
<b>7.5</b>	IF-THEN and IF-THEN-ELSE Instructions	159
<b>7.6</b>	Extended Mnemonics	161



- 7.7 Looping Structures 164
- 7.8 Example: Insertion in a Linked List 167
- 7.9 Style and Control Structure Summary 171

MAIN IDEAS 172

PROBLEMS FOR REVIEW AND IMAGINATION 173

REFERENCES 178

---

## CHAPTER 8

# LOOPING AND ADDRESS MODIFICATION

---

- 8.1 The Anatomy of a Loop: Address Modification 179
- 8.2 Address Modification: Changing and Testing Contents of a Base Register 184
- 8.3 Address Modification: Use of Index Registers 187
- 8.4 The BXH and BXLE Instructions 190
- 8.5 The Programming Process: A Sequential Search 195
- 8.6 The BCT and BCTR Instructions 197
- 8.7 Ordered Lists and Binary Search 199

MAIN IDEAS 203

PROBLEMS FOR REVIEW AND IMAGINATION 204

REFERENCES 207

---

## CHAPTER 9

# DEBUGGING

---

- 9.1 Exceptions and Interrupts 208
- 9.2 Indicative Dumps 214
- 9.3 Error Messages 216
- 9.4 Fuller Dumps 217
- 9.5 Advance Preparation 222
- 9.6 Partial Dumps 223
- 9.7 Trace Features 225

<b>9.8</b>	Interactive Debugging in CMS	226
<b>9.9</b>	A Last Few Hints on Programming and Debugging	230
MAIN IDEAS		232
PROBLEMS FOR REVIEW AND IMAGINATION		232
REFERENCES		233

---

## CHAPTER 10

# CHARACTER OR BYTE OPERATIONS

---

<b>10.1</b>	Byte Transfer or Move Instructions	234
<b>10.2</b>	Character Compare Operations	238
<b>10.3</b>	An Example: Searching for a Name	241
<b>10.4</b>	Control Sections	242
<b>10.5</b>	An Example: Character Set Conversion	244
<b>10.6</b>	An Example: Counting Digits	246
<b>10.7</b>	An Example: Generating a Symbol Table	247
MAIN IDEAS		254
PROBLEMS FOR REVIEW AND IMAGINATION		255
REFERENCES		257

---

## CHAPTER 11

# BIT OPERATIONS

---

<b>11.1</b>	Logical Instructions: Arithmetic on Unsigned Numbers	258
<b>11.2</b>	Shift Instructions	260
<b>11.3</b>	An Example: Hexadecimal Conversion	267
<b>11.4</b>	Taking a Square Root	268
<b>11.5</b>	Logical Instructions: AND and OR	269
<b>11.6</b>	Generating Moves in Checkers	278
MAIN IDEAS		285
PROBLEMS FOR REVIEW AND IMAGINATION		285
REFERENCES		289

---

**CHAPTER 12****INPUT AND OUTPUT THROUGH THE OPERATING SYSTEM**

- 12.1** Basic Structure of Input and Output Processing 291
- 12.2** Organization of a Data Set 295
- 12.3** Buffering and Exit Options 299
- 12.4** Using QSAM Macros in a Program 302
- 12.5** Survey of Job Control Language 312
- 12.6** Examples of Specific Data-set Operations 321
- 12.7** Sorting by Merging 329

MAIN IDEAS 334

PROBLEMS FOR REVIEW AND IMAGINATION 336

REFERENCES 340

**CHAPTER 13****FLOATING-POINT ARITHMETIC**

- 13.1** Representations of Floating-point Numbers 343
- 13.2** Floating-point Registers; Load and Store Instructions 345
- 13.3** Floating-point Add, Subtract, and Compare Instructions 350
- 13.4** Floating-point Multiply and Divide 353
- 13.5** Unnormalized Add and Subtract Operations 356
- 13.6** Exceptions and Interrupts 358
- 13.7** An Example: A Regression Calculation 359
- 13.8** Extended-precision Instructions 365
- 13.9** Conversions 367

MAIN IDEAS 371

PROBLEMS FOR REVIEW AND IMAGINATION 372

REFERENCES 376

**CHAPTER 14****DECIMAL ARITHMETIC**

- 14.1** Packed Decimal Representations: Internal and Assembler Language 378

<b>14.2</b>	General Structure of the Decimal Instruction Set	379
<b>14.3</b>	Add, Subtract, and Compare Instructions	380
<b>14.4</b>	Moving a Packed Decimal Number	382
<b>14.5</b>	Multiply and Divide Instructions	384
<b>14.6</b>	Exceptions	386
<b>14.7</b>	Examples	386
MAIN IDEAS		392
PROBLEMS FOR REVIEW AND IMAGINATION		393
REFERENCES		395

---

## CHAPTER 15

# TRANSLATE, EDIT, AND EXECUTE INSTRUCTIONS

---

<b>15.1</b>	Translate Instructions	396
<b>15.2</b>	Editing	399
<b>15.3</b>	The EXecute Instruction	405
MAIN IDEAS		407
PROBLEMS FOR REVIEW AND IMAGINATION		408
REFERENCE		410

---

## CHAPTER 16

# MACRO DEFINITION AND CONDITIONAL ASSEMBLY

---

<b>16.1</b>	Outline of Facilities	412
<b>16.2</b>	Definition and Use of a Macro	413
<b>16.3</b>	Set Symbols, System Variable Symbols, and Attributes	418
<b>16.4</b>	Conditional Assembly	425
<b>16.5</b>	Substrings and Other Macro Features	429
MAIN IDEAS		431
PROBLEMS FOR REVIEW AND IMAGINATION		432
REFERENCES		435

---