PROGRAMMING ASSEMBLER LANGUAGE

PETER ABEL

PROGRAMMING ASSEMBLER LANGUAGE

Peter Abel

British Columbia Institute of Technology

RESTON PUBLISHING COMPANY, INC.
Reston, Virginia
A Prentice-Hall Company

Library of Congress Cataloging in Publication Data

Abel, Peter
Programming assembler language.

Includes index.
1. Assembler language (Computer program language)
2. IBM 360 (Computer)—Programming. 3. IBM 370
(Computer)—Programming. I. Title.
QA76.73.A8A23 001.6'424 79-255
ISBN 0-8359-5658-X

Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia 22090

10 9 8 7 6 5 4 3 2

Printed in the United States of America.

PROGRAMMING ASSEMBLER LANGUAGE

PREFACE

Assembler language is the fundamental "low-level" language of the IBM 360 and 370 computers. As such, it is directly translatable into machine language; thus, one Assembler instruction typically generates one machine code instruction. "High-level" languages like COBOL and PL/I are easier to learn. Why then an emphasis on learning Assembler language? An understanding of Assembler language can help the programmer in a number of ways:

— A knowledge of Assembler can facilitate learning of any other language, including "high-level" languages and other assembly languages. And with a background in Assembler, the user can more clearly understand what the computer is doing.

— A knowledge of Assembler can help the programmer become more efficient. High-level languages like COBOL and PL/I can be deceptive, and appear to execute in some mysterious fashion. A programmer familiar with Assembler can code high-level languages with an understanding of what machine code they generate, and what is the more efficient technique. For example, why in COBOL does the use of COMPUTATIONAL, COMPUTATIONAL-3, and SYNC have considerable effect on the program's efficiency? What is the significance in PL/I of Decimal Fixed, Aligned, and Defined? With knowledge of Assembler, a programmer can examine the generated code to determine more efficient ways to write certain routines.

— Although most high-level languages provide extensive debugging aids, there are times when the programmer needs to delve into the generated machine code or examine storage dumps.

— Programs written in Assembler may be considerably more efficient in storage space and execute-time, a useful consideration if such programs are run frequently.

— Some advanced areas, such as technical support and telecommunications, require an extensive knowledge of Assembler.

Although the material in this text has been used successfully as an introduction to programming, most educational institutes would not teach Assembler as an introductory language. Generally, the concepts of logic and programming style are easier to learn when there is less need for concern with rigorous rules and field sizes. The text does not, however, assume that the reader has had much, if any, programming experience. The approach of the text is to introduce simple processing using card

input and printer output, first with character data only, and then with simple decimal arithmetic, all by Chapter 4. In this way, packed (decimal) data and editing are introduced early, and the user is soon writing quite realistic programs.

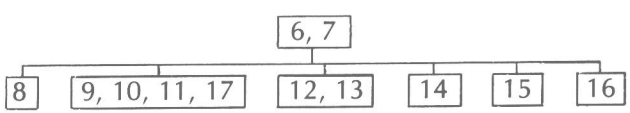
The book should provide both a practical guide for the Assembler student and also may act subsequently as a useful reference. These two objectives are accomplished by:

1. A step-by-step progression of material, from simple processing through to complex. There are many practical examples of complete and partial programs to illustrate concepts as they are introduced.
2. Chapters organized by logical topics, such as character data, packed, binary, input/output. The user can concentrate on mastering one programming area at a time, and most related material is contained in its own chapter.

The complexities of base/displacement addressing and file definition are delayed through use of several simple macros, similar to those used in many colleges. The text drops these macros by Chapters 6 and 7, where the technical material is covered in detail. Appendix E provides a listing of the macros for those who want to catalogue them on their own system.

The two major IBM operating systems are DOS and OS. The text covers the differences between them, giving examples for both.

It is possible to proceed through the text by several routes. Chapters 0 through 3 are fundamental, and the normal steps would be to continue sequentially with decimal arithmetic in Chapters 4 and 5. It is recommended next to cover the important material in Chapter 6 on base/displacement addressing and the instruction format, and then the elements of input/output in Chapter 7. By this point the user should be capable of coding some quite advanced programs. Chapter 8 on Programming Strategy could be covered in part or total, perhaps the sooner the better to get the user into subroutine logic. Following Chapter 8, the chapters need not all be covered sequentially. The following diagram indicates related chapters in boxes that may be taken in any sequence following Chapters 6 and 7:



Chapters 9, 10, and 11 develop related material on processing binary data. Chapter 12 on Magnetic Tape introduces basic material required for an understanding of Disk Storage in Chapter 13. To complete Chapter 14 (Macro Writing) would require some familiarity with the material in Chapter 9. Anyone interested in linking separately assembled programs could attempt Chapter 15 directly after Chapter 7, perhaps referencing Chapter 9 for some basic binary operations. Chapter 16 on Operating Systems is presented for general useful information, although not entirely related to Assembler programming as such.

Among the users of earlier versions of this text, many have worked ahead of the course, experimenting with binary operations, macro writing, and subprogram linkage. Such motivation is certainly commendable and should be encouraged.

The IBM manuals concerned with the material in this text require a bookshelf about five feet wide. Readers should not expect, therefore, that this or any other single book will provide all there is to know about the Assembler language and related topics. Eventually the IBM manuals have to be referenced for current detailed information. The following IBM manuals or their equivalent are especially recommended:

IBM FORM	TITLE
GA22-7000	IBM System/370 Principles of Operation. (370 system organization, machine instructions, input/output.)
GC33-4010	OS/VS-DOS/VS-VM/370 Assembler Language. (Assembler statements and macros.)
GC33-5373	DOS/VS Supervisor and Input/Output Macros.
GC28-6646	OS Supervisor Services and Macro Instructions.
GC26-3746	OS Data Management Services Guide.
GC26-3794	OS Data Management Macro Instructions.

Other useful manuals include those on Job Control, disk file organization, tape labels, disk labels, and the operating system.

Peter Abel

ACKNOWLEDGMENT

The author is grateful for the assistance from all those who contributed typing, reviews, and suggestions, and to IBM for permission to reproduce some of their copyrighted materials. The following materials are printed with permission, and with modifications from publications copyrighted in 1966, 1970, 1972, 1973, and 1974 by International Business Machines Corporation as IBM form numbers GA22-7000, GX20-1850 and GC20-1649: Figures 2-2, 13-2, 16-3, Appendix A, and Appendix C.

CONTENTS

APPENDICES

PART I

INTRODUCTION TO THE COMPUTER AND THE ASSEMBLER