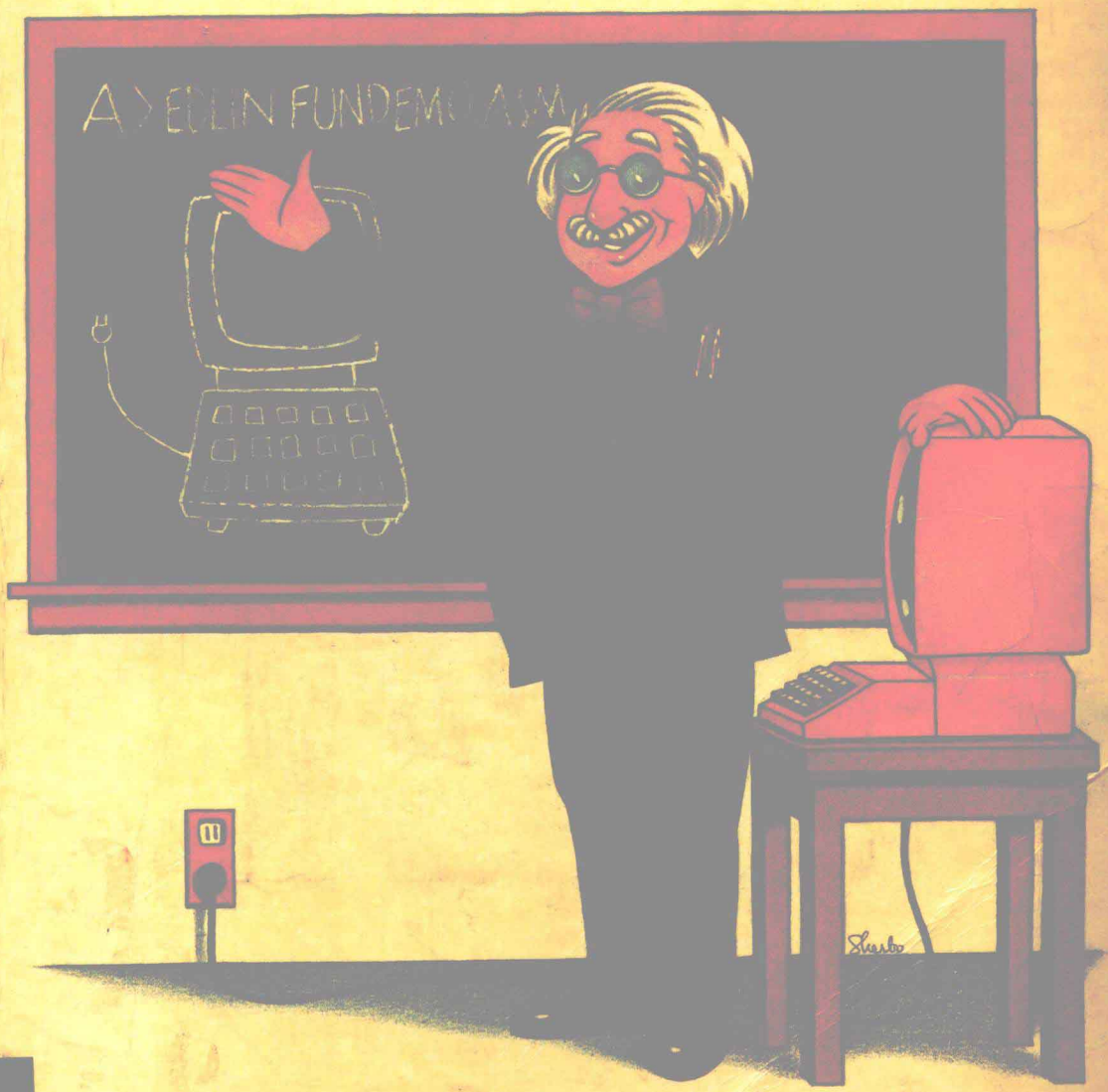
IBM-PC® Assembly Language is Fun & Easy



Samuel A. Solomon

IBM-PC® ASSEMBLY LANGUAGE IS FUN AND EASY

Samuel A. Solomon

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

Library of Congress Cataloging in Publication Data

Solomon, Samuel A.
IBM-PC® assembly language is fun and easy.

IBM Personal Computer—Programming. 2. Assembler language (Computer program language) I. Title. II. Title: I.B.M. P.C. assembly language is fun and easy.
QA76.8.I2594S64 1984 001.64'24 83-26965
ISBN 0-8359-3038-6

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

ACKNOWLEDGMENTS

The author wishes to acknowledge the following persons whose help was invaluable during the writing of this book: James Rusten, for editorial and artistic contributions; Linda Rusten, for artistic talent; Richard Nielsen, for assistance in helping to obtain the hardware for this book; and Dave Dixon, for assistance in helping to obtain the software for this book.

INTRODUCTION

Confusion. Fright. Complexity. Chips, bits, binary, RAM. Are those the thoughts that enter your mind when you think about assembly language? If they do, you are one of millions of new computer users who will be facing such thoughts in the coming years as more and more people delve deeper into their IBM PCs and PC compatibles.

If the thought of assembly language overpowers you, this book is for you. If you know nothing about exotic numbering systems, or if you have never spoken a word of BASIC or any other computer language, or if this is the first book about your IBM PC that you have picked up, this book is for you. If you have dabbled in BASIC or can count to four in binary, that's great but not a prerequisite.

As a beginner you will enjoy the approach taken by *IBM-PC Assembly Language Is Fun and Easy*. You will giggle your way to understanding as you read how hexadecimal numbers were used to mark the runways of Stonehill Airport. You will smile with appreciation as the eccentric Dr. Bittwiddler explains the mysteries of bits, bytes, and structuring assembly language programs. You will admire Dr. Hacker's concise and lucid explanations of some of the nastier assembly language concepts.

This book is ideal for the assembly language beginner. It uses humor and metaphor to put every assembly language instruction into understandable terms. Besides explaining the instructions, it goes a step further and explains how the instructions can be made to fit together into programs. Concepts of structured programming are put into human terms, with input, process, and output compared to the evolution of life on Earth.

The book focuses around one program in particular to illustrate how assembly language programs are put together. This program is useful in its own right as an aid to writing assembly language programs, and programs similar to it are currently being sold for more than the price of this book. A listing of the program is included with this book, or you can save yourself the trouble of keying it in and assembling by purchasing the program on a supplementary diskette directly from the author.

So if you still don't believe that IBM PC assembly language is really fun and easy, you're in for a pleasant, as well as a fun and easy, surprise. Let's begin right now!

ABOUT THE AUTHOR

Samuel A. Solomon is co-owner of the Learning Methods Computer Training Center in San Jose, California. Mr. Solomon learned the art of making obtuse subjects comprehensible during his fifteen years' experience in all aspects of newspaper production.

In 1977 Mr. Solomon joined Radio Shack (a division of the Tandy Corporation) in Ft. Worth, Texas, where he designed and wrote the best-selling Scripsit series of word processing software. Each of the three programs in the series was written in Z-80 assembly language and took about one year to produce. In 1981 Mr. Solomon founded Executive Software of San Francisco, California, a company that engages in contract programming and consulting for microcomputers.

CONTENTS

CHAPTER TWO

CHAPTER THREE

CHAPTER FOUR

CHAPTER FIVE

CHAPTER SIX

CHAPTER SEVEN

CHAPTER EIGHT

CHAPTER NINE

CHAPTER TEN

CHAPTER ELEVEN

CHAPTER TWELVE

CHAPTER THIRTEEN

CHAPTER FOURTEEN

CHAPTER FIFTEEN

CHAPTER SIXTEEN

Chapter One

IS THERE LIFE AFTER BASIC?

Welcome to the fascinating world of assembly language programming on the IBM® Personal Computer. Our study of assembly language will be an adventure, complete with all the perils and satisfactions of a long journey. This book will not lecture you, nor will it bore you with unneeded technical detail. Instead, by the end of this chapter you will have written and executed a simple assembly language program. Then, together, we will design and study an assembly language program that you will use again and again as you continue to explore your IBM PC. You will not only know the 92 instructions that comprise the 8088 microprocessor, but how to express them as a program using the IBM Macro Assembler, and by the end of this book you will know something about programming. You will learn more than just the definitions of the words of assembly language—you will know how to use them to form sentences, paragraphs, and programs.

Familiarity with a programming language like BASIC is helpful, but not necessary to comprehend this book. Unlike many other books on assembly language which tend to be heavy on the technical side, this book includes explanations of programming concepts from the ground up. For those of you who do know BASIC, comparisons of assembler and your native BASIC are employed to aid in your learning. For the serious students among you, each chapter contains a glossary, a summary of what you learned, and a few review questions. The chapter–by–chapter glossary is combined at the end of the book to form a handy reference.

THE GOOD DOCTORS BITTWIDDLER AND HACKER

The study you are about to undertake will not be easy. However, we are extremely fortunate to have with us two distinguished computer scientists who will help us grasp fundamental concepts and supplement the teaching skills of your author. They are Dr. Boran Bittwiddler and Dr. Hammond Hacker. I will now introduce Dr. Bittwiddler, who will

give us a brief explanation of what assembly language programming is. But, before I do that let me warn you that I've been roundly criticized for allowing Dr. Bittwiddler's intelligent ramblings to grace these pages—you see, after many years of living locked in the basement with a computer, several cases of soda pop, and a truckload of corn chips, some would say he has a few bits loose here and there. Many of you are likely to agree with Dr. Hacker, one of Dr. Bittwiddler's most outspoken critics, that Dr. Bittwiddler has a personality that could be characterized as—shall we say—offensive. If that is your opinion, you are welcome to skip the Doctor's sections and pay attention to Dr. Hacker only. So without further ado, please welcome Dr. Boran Bittwiddler:

"Thank you ladies and gentlemen, and fellow scientists. Let me begin by saying that I am most honored to help guide you through this most exciting of journeys—an adventure into the guts of the IBM Personal Computer. It is with a confirmed sense of adventure that I accompany you on this important mission, and with trepidation that I——"

"Please, Dr. Bittwiddler, get to the point. We've heard your tired ramblings too many times before."

As I said, Dr. Hacker is not one of Dr. Bittwiddler's most enthusiastic fans. We will request him to be quiet and let the Doctor finish. Dr. Bittwiddler?

"Thank you. Now, as I was saying....The computer we have chosen to explore is the IBM Personal Computer. This machine features a state–of–the–art 16–bit 8088 microprocessor, which will be your main concern as you learn to make it do tricks that BASIC programmers could only dream of in those dark, primitive days before assembly language became a staple of the masses.

HOW THE COMPUTER "THINKS"

"We humans have vast memories in our brains which can store millions of words and images. But computers are unlike us. While they do, indeed, have memories, those memories are capable of storing nothing but numbers. A program is really just a series of instructions, like a recipe, reduced to numerical codes. Let us now construct a sample code to illustrate this principle: