Introduction

In the summer of 1992 my school did what so many schools were doing: they removed their mainframe computer. I fully understood their reasons for doing so: all administrative processing had been moved to another platform, and this machine was being maintained solely for our Assembler Language Programming and Advanced COBOL classes. It simply was not cost effective to keep that machine. Nevertheless, I was committed to continuing to teach assembler programming. But now it appeared as though I was without a means for doing so.

Several years earlier I had heard about PC/370, a shareware product which allows the user to write and execute mainframe assembler code on the PC. Skeptical, I tried it. And loved it! The differences between mainframe assembler and PC/370 are trivial. PC/370 has the look and feel of the mainframe. But you don't need a fancy PC to run PC/370: every program in this book was written on a Compaq 286LTE using DOS 3.4!

I have used PC/370 in my assembler classes in school and in industry since that time and have never been disappointed. The only problem I had was the lack of a textbook to support PC/370. That was the original impetus for writing this book.

But I wanted this book to be something more than a user's guide to PC/370. I wanted to write an easy-to-read assembler text which would emphasize business applications, based on my experience as an assembler programmer in industry. I believe I have been successful in doing so.

Currently, there is renewed interest in mainframe programming. This is largely due to the year 2000 problem (aka "Y2K" and "the millenial bug"). Many "legacy systems" were written in assembler, and there simply isn't the time and/or money to rewrite them in another language. But they must be made century compliant. Still more of these systems are written in COBOL, and I subscribe to the view shared by many in this industry; that is, learning assembler will make you a better COBOL programmer. A knowledge of assembler demystifies the COBOL compiler which is, after all, just an assembler programmer.

I realize few of the readers of this text will become assembler programmers. But the skills learned from completing the exercises in this text will benefit the reader regardless of the language used.

This text is written in a very non-technical style. You'll be writing assembler programs as early as the first chapter. Execute the sample programs. Complete the exercises. And have fun while you learn!

How this Book is Organized

In chapter 1 you will learn the mechanics of writing, assembling, and executing assembler programs using PC/370. You will also learn how to create the datasets used in this text.

In chapter 2 you will learn the fundamentals of defining and moving character fields. You will then be able to produce simple formatted reports.

In chapter 3 you will learn how to do IF's in assembler. It will probably be unlike anything you've done before. But you can do it: just think very low level!

In chapter 4 you will learn how to structure an assembler program. Many programmers think "structure" and "assembler" don't belong in the same sentence. But if you follow some very simple rules, you'll soon be writing programs which are very easy to modify and maintain.

In chapter 5 you will learn more about data representation, including the binary and hexadecimal number systems and the EBCDIC and ASCII collating sequences. Many assembler texts put this information in the first chapter, but I have put it off until now in favor of enabling you to write programs as soon as possible.

In chapter 6 you will learn about some of the assembler instruction formats. Upon completion of this chapter you will actually "be" the assembler, determining the object code instructions that would be produced by the assembler for your source code.

In chapter 7 you will learn about packed decimal arithmetic. Most arithmetic in assembler is done with packed decimal numbers or binary numbers. Packed decimal numbers are the more important of the two since binary numbers must be converted to packed decimal in order to be printed. So this is a very important chapter.

In chapter 8 you will learn about page break logic. This is one of many applications of the arithmetic you learned in chapter 7.

In chapter 9 you will learn about the EDIT instruction, which enables you to print a packed decimal number in just about any form you need (for example, with embedded commas and a decimal point). At this point you will be producing professional caliber reports.

In chapter 10 (Control Break Logic), chapter 11 (More than One Input File), and chapter 12 (Sequential File Update) you will learn the logic necessary to complete some very common business applications. There are no new assembler instructions here: the emphasis is on logic.

In chapter 13 you will learn more about packed decimal arithmetic, specifically how to multiply and divide. Division in assembler is particularly unusual, but I'll show you some absolutely fool-proof tricks for doing so.

In chapter 14 you will learn about binary, or register, arithmetic. Specifically, you will learn about addition and subtraction. Things start to get a little more confusing here, but the payoff makes it worthwhile. This chapter is particularly important for computer science (vs. information systems) majors.

In chapter 15 you will learn about table processing. Table processing could not be done in assembler without register arithmetic.

In chapter 16 you will learn more about binary, or register, arithmetic. Specifically, you will learn how to multiply and divide.

In chapter 17 you will learn about bit level operations. There are some interesting applications in this chapter, including changing a letter from upper case to lower case (and vice-versa), removing the sign from a numeric field, encryption, swapping two fields, multiplication and division by powers of two, and accessing the system date and time.

In appendix A you will find a description of the datasets used in this text.

In appendix B you will learn how to convert your PC/370 programs to run on the mainframe.

A Note to Instructors

Perhaps you lost your mainframe, just as I lost mine. You recognize the value of assembler programming skills for all mainframe programmers, but haven't been able to teach your course for want of a mainframe. Take heart: with this book and PC/370 you are back in business!

I have found that I can cover the entire text in one semester long course at Waubonsee Community College. Most of these students have already had a semester of BASIC, a semester of Programming Logic, and a semester of COBOL. I have to hurry towards the end of the semester, but it can be done!

As you plan your course, you may want to keep the following in mind:

Performance based objectives are found at the start of each chapter and will be most beneficial to you in planning your course.

Most chapters contain "You Try It" exercises within the text of the chapter. I usually use these as in-class exercises which prompt good discussion and help to keep me on track.

I have put off data representation until chapter 5 in favor of getting my students writing assembler programs as early as possible. I realize that many instructors would prefer to cover this material in the first week of the course. This chapter can be discussed sooner, but the discussion of the sample programs contained therein will need to be deferred.

Chapter 10 (Control Break Logic), Chapter 11 (More than One Input File), and Chapter 12 (Sequential File Update) discuss programming techniques commonly found in business applications. I believe these chapters are very important for future applications programmers, but since none of these chapters introduce any new assembler instructions, and since these concepts may have been discussed in other courses, you can skip any or all of them as necessary to fit the available time.

The programming assignments throughout this text make use of the "databases" contained in Appendix A. You will notice the recurring "Small Town" theme. The Small Town Community College database is used in most of the example programs. You may want to pick one or two of

the other databases, such as the Small Town Hardware Store or the Small Town Blood Bank and choose programming exercises which utilize these databases. In this way your students' programs will evolve in complexity over the duration of the course.

None of the date fields in any of the files in this text include the century. As you are probably aware, this is usually the case with legacy systems. Challenge your students to make their programs "century compliant".

If your students still have access to a mainframe, consider having them develop their programs using PC/370, then upload their programs to the mainframe and make the necessary changes to run their programs there. I did this while teaching this course at De Paul University and it was very successful. (The changes are trivial and are discussed in Appendix B.)

About PC/370

When you assemble and link a PC/370 program, you will see a reminder that it is shareware and that you should send forty-five dollars to become a registered user. You do *not* need to register, as indicated below (letter on file):

> Per your request I am sending this letter confirming that I, Donald S. Higgins, author of the PC/370 shareware package *no longer accept registrations*, and that you are free to distribute the shareware version without charge. The last version of the PC/370 shareware is 4.2 dated January 1988. There is a copy of this version available for download from the Compuserve PC Programmers forum library under the name PC370.ZIP.
>
> Sincerely,
>
> (signed) Donald S. Higgins

Thanks to Don for developing this great product!