Introduction: Pushing the Envelope

So you want to be a PC guru? You've set yourself an ambitious and difficult goal, with no guarantee of success. There's no sure-fire recipe for becoming a guru, any more than there's a recipe for becoming a chess grand master. There is, however, one way you can greatly improve your chances: become an expert assembly language programmer. Assembly language won't by itself make you a guru--but without it you'll never reach your full potential as a programmer.

Why is assembly language so important in this age of optimizing compilers and program generators? Assembly language is fundamentally different from all other languages, as we'll see throughout The Zen of Assembly Language. Assembly language lets you use every last resource of the PC to push the performance envelope; only in assembly language can you press right up against the inherent limits of the PC.

If you aren't pushing the envelope, there's generally no reason to program in assembler. High-level languages are certainly easier to use, and nowadays most high-level languages let you get at the guts of the PC--display memory, DOS functions, interrupt vectors, and so on--without having to resort to assembler. If, in the other hand, you're striving for the sort of performance that will give your programs snappy interfaces and crackling response times, you'll find assembly language to be almost magical, for no other language even approaches assembler for sheer speed.

Of course, no one tests the limits of the PC with their first assembler program; that takes time and practice. While many PC programmers know something about assembler, few are experts. The typical programmer has typed in the assembler code from an article or two, read a book about assembler programming, and perhaps written a few assembler programs of his own--but doesn't yet feel that he has mastered the language. If you fall into this category, you've surely sensed the remarkable potential of assembler, but you're also keenly aware of how hard it is to write good assembler code and how much you have yet to learn. In all likelihood, you're not sure how to sharpen your assembler skills and take that last giant step toward mastery of your PC.

This book is for you.

Welcome to the most exciting and esoteric aspect of the IBM PC. The Zen of Assembly Language will teach you how to create blindingly fast code for the IBM PC. More important still, it will teach you how to continue to develop your assembler programming skills on your own. The Zen of Assembly Language will show you a way to learn what you need to know as the need arises, and it is that way of learning that will serve you well for years to come. There are facts and code aplenty in this book and in the companion volume, but it is a way of thinking and learning that lies at the heart of The Zen of Assembly Language. Don't take the title to mean that this is a mystical book in any way. In the context of assembly-language programming, Zen is a technique that brings intuition and non-obvious approaches to bear on difficult problems and puzzles. If you would rather

think of high-performance assembler programming as something more mundane, such as right-brained thinking or plain old craftsmanship, go right ahead; good assembler programming is a highly individualized process.

As the subtitle of this book indicates, The Zen of Assembly Language is about assembly language for the IBM PC (and, by definition, compatible computers). In particular, the bulk of the book will focus on the capabilities of the 8088 processor that lies at the heart of the PC. However, many of the findings and almost all of the techniques I'll discuss can also be applied to assembly-language programming for the other members of Intel's 808X processor family, including the 8086, 80186, 80286, and 80386 processors. This book doesn't much apply to computers built around other processors, such as the 68000 family, the Z80, the 8080, or the 6502, since much of the Zen of assembly language in the case of the IBM PC derives from the highly unusual architecture of the 808X family.

While I will spend a chapter looking specifically at the 80286 found in the AT and PS/2 Models 50 and 60 and the 80386 found in the PS/2 Model 80, I'll concentrate primarily on the 8088 processor found in the IBM PC and XT, for three reasons. First, there are about 10,000,000 8088-based computers around, ensuring that good 8088 code isn't going to go out of style anytime soon. Second, the 8088 is far and away the slowest of the processors used in IBM-compatible computers, so no matter how carefully code is tailored to the subtleties of the 8088, it's still going to run much faster on an 80286 or 80386. Third, many of the concepts I'll present regarding the 8088 apply to the 80286 and 80386 as well, but to a different

degree. Given that there are simply too many processors around to cover in detail (and the 80486 on the way), I'd rather pay close attention to the 8088, the processor for which top-quality code is most critical, and provide you with techniques that will allow you to learn on your own how best to program other processors.

WHAT YOU'LL NEED

The tools you'll need to follow this book are simple: a text editor to create ASCII program files, the Microsoft Assembler (MASM) or a compatible assembler to assemble programs, and the Microsoft Linker or a compatible linker to link programs into an executable form. I used version 2.1 of the Brief text editor, MASM version 5.0, and the Microsoft Linker version 3.60 to prepare the programs in this book.

There are several types of reference material you should have available as you pursue assembler mastery. You will certainly want a good general reference on 8088 assembler. IBM's hardware, BIOS, and DOS technical reference manuals are also useful references, containing as they do detailed information about the resources available to assembler programmers.

If you're the type who digs down to the hardware of the PC in the pursuit of knowledge, you'll find Intel's handbooks and reference manuals to be invaluable (albeit none too easy to read), since Intel manufactures the 8088 and many of the support chips used in the PC. There's simply no way to understand what a hardware component is capable of doing in the context

of the PC without a comprehensive description of everything that part can do, and that's exactly what Intel's literature provides.

Finally, keep an eye out for articles on assembly-language programming. Articles provide a steady stream of code from diverse sources, and are your best source of new approaches to assembler programming.

By the way, the terms "assembler" and "assembly-language" are generally interchangeable. While "assembly-language" is perhaps technically more accurate, since "assembler" also refers to the software that assembles assembly-language code, "assembler" is a widely-used shorthand that I'll use throughout this book. Similarly, I'll refer to "the Zen of assembler" as a shorthand for "the Zen of assembly language."

THE PATH TO THE ZEN OF ASSEMBLER

The Zen of Assembly Language consists of four major parts, contained in two volumes. Parts I and II are in this book, Volume I, while Parts III and IV are in Volume II, The Zen of Assembly Language: The Flexible Mind. While the book you're reading stands on its own as a tutorial in high-performance assembler code, the two volumes together cover the whole of superior assembler programming, from hardware to implementation. I strongly recommend that you read both.

Part I introduces the concept of the Zen of assembler and details the tools we'll use to delve into assembler code performance.

Part II covers various and sundry pieces of knowledge about

assembler programming, examines the resources available when programming the PC, and probes fundamental hardware aspects that affect code performance.

Part III (in Volume II) examines the process of creating superior code by combining the detailed knowledge of Part II with varied and often unorthodox coding approaches.

Part IV (also in Volume II) illustrates the Zen of assembler in the form of a working animation program.

The four parts together teach all aspects of the Zen of assembler: concept, knowledge, the flexible mind, and implementation. Together, they will take you down the road to mastery of the IBM PC.