Preface

There are three main reasons for writing this book. While several assembly language books are on the market, almost all of them cover only the 8086 processor—a 16-bit processor Intel introduced in 1979. A modern computer organization or assembly language course requires treatment of a more recent processor like the Pentium, which is a 32-bit processor in the Intel family. This is one of the main motivation for writing this book.

There two other equally valid reasons. The book approaches assembly language programming from the high-level language viewpoint. As a result, it focuses on the assembly language features that are required to efficiently implement high-level language constructs.

Performance is another reason why people program in assembly language. This is particularly true with real-time application programming. Our treatment of assembly language programming is oriented toward performance optimization. Every chapter ends with a performance section that discusses the impact of specific set of assembly language statements on the performance of the whole program. Put another way, this book focuses on performance-oriented assembly language programming.

Intended Use

This book is intended as an introduction to assembly language programming using the Intel 80X86 family of processors. We have selected the assembly language of the Intel 80X86 processors (including the Pentium processor) because of the widespread availability of PCs and assemblers. Both Microsoft and Borland provide assemblers for the PCs.

Assembly language programming is part of an undergraduate curriculum in computer science, computer engineering, and electrical engineering departments. This book can be used as a text for those courses that teach assembly language. It can also be used as a companion text in a computer organization course for teaching the assembly language. Because of the performance-oriented assembly language programming style advocated by the book, the book is especially useful in real-time programming courses in engineering.

In addition, it can be used as a text in vocational training courses offered by community colleges. Because of the teach-by-example style used in the book, it is also suitable for self-study by computer professionals

and engineers.

Prerequisites

The student is assumed to have had some experience in a structured, high-level language such as C. However, the book does not assume extensive knowledge of any high-level language—only the basics are needed. Furthermore, it is assumed that the student has rudimentary background in the software development cycle, as is obtained in a typical high-level programming course. Of course, the student is assumed to be familiar with using the PC and its operating system.

Features

Here is a summary of the special features that sets this book apart:

- The book is self-contained and does not assume background in computer organization. All necessary background material on computer organization is presented in the book.
- A methodical organization of chapters for a step-by-step introduction to the assembly language.
- The book covers all processors in the Intel 80X86 series (from 8086 to Pentium).
- Extensive examples are used in each chapter to illustrate the points discussed in the chapter. Our objective is not just to explain how an instruction works but to provide the rationale as to why the instruction has been designed the way it is. This is the best way of understanding the strengths and weaknesses of the Intel 80X86 series of processors.
- Procedures are introduced early to encourage modular programming in developing assembly language programs.
- A set of input and output routines is provided so that the student can focus on developing assembly language programs rather than spending time in understanding how input and output can be done using the basic I/O functions provided by the operating system.
- This book does not use fragments of code in examples. All examples are complete in the sense that they can be assembled and run giving a better feeling as to how these programs work.

- All examples and other required software are available either on a disk that accompanies the text or online (check my home page for information) to give opportunities for students to perform hands-on assembly programming.
- Most chapters are written in such a way that each chapter can be covered in two or three 60-minute lectures by giving proper reading assignments. Typically, important concepts are emphasized in the lectures while leaving the other material in the book as a reading assignment. Our emphasis on extensive examples facilitates this pedagogical approach.
- Since performance is one of the two main objectives in using assembly language, each chapter contains a "Performance" section that discusses the performance implications of the topics/instructions discussed in that chapter. This aspect is important in those courses that deal with real-time programming. However, the performance section is completely independent and can be omitted altogether.
- Inter-chapter dependencies are kept to a minimum to offer maximum flexibility to instructors in organizing the material. Each chapter clearly indicates the objectives and provides an overview at the beginning and a summary at the end.
- Each chapter contains two types of exercises—review and programming—to reinforce the concepts discussed in the chapter.
- The appendices provide special reference material that contains a thorough treatment of various topics.

Overview and Organization

The book is divided into four parts. Part I presents introductory topics and consists of the first three chapters. Chapter 1 provides introduction to assembly language and presents reasons for programming in assembly language. Chapter 2 presents basics of computer organization with a focus on the Intel 80X86 family of processors. In particular, this chapter gives sufficient details on the 16- and 32-bit Intel processors so that the student can effectively program in the assembly language. Chapter 3 gives an overview of the assembly language. After covering these three chapters, one can write simple stand-alone assembly language programs without requiring further information from the other chapters. These three chapters should be covered in the sequence presented in the book.

The amount time spent on this part can vary depending on the background of the students.

Part II provides basic topics and consists of five chapters—Chapters 4 through 8. To emphasize the importance of modular programming, procedures are introduced early on (in Chapter 4). The other chapters in this part expand on the overview given in Chapter 3. Chapter 5 presents the addressing modes supported by the Intel 16- and 32-bit processors. This chapter also contains a detailed discussion on the motivation for providing the various addressing modes. Chapter 6 discusses the arithmetic instructions and the use of the flags register. Chapters 7 and 8 present conditional and bit manipulation instructions. A feature of these two chapters is that they relate how high-level language statements can be implemented using the instructions discussed in these two chapters. The first two chapters of this part—Chapters 4 and 5—should be covered in some detail for proper grounding in assembly language programming. However, the remaining three chapters can be studied in any order. As well, the depth that these three chapters are covered can be varied without sacrificing the effectiveness depending on the time available and importance to the course objective.

The remaining five of the thirteen chapters constitute Part III. These chapters deal with advanced topics. Chapter 9 discuses the string processing instructions in detail. Macros and conditional assembly directives are discussed in Chapter 10. ASCII and BCD arithmetic instructions are presented in Chapter 11. Chapter 12 takes a detailed look at the interrupt mechanism and input/output interface. This is an important chapter in computer organization and real-time system programming courses. The final chapter deals with high-level language interface, which allows mixed-mode programming in more than one language. We use C and assembly language to cover the principles involved in mixed-mode programming. The chapters in this part can be covered in any order the instructor wishes. While most of the topics of this part are optional, a good, well-rounded course should cover some aspects of macros (Chapter 10), interrupts (Chapter 12), and high-level language interface (Chapter 13).

The five appendices provide a wealth of reference material needed by the student. Appendix A primarily discusses the number systems and their internal representation. Appendix B gives information on the use of I/O routines provided with this book and the assembler soft-

ware. Debugging aspect of assembly language programming is discussed in Appendix C. Selected Pentium instructions are given in Appendix D. Finally, Appendix E gives the standard ASCII table.

Acknowledgments

Several people have contributed, either directly or indirectly, in writing this book. First and foremost, I would like to thank my family for enduring my preoccupation with this project. My wife Sobha has taken care of our daughter so that I could spend more time on the book. My daughter Veda, who will be four in January 1999, contributed in her own way by allowing me to proofread some of the chapters while playing in her favorite park in Ottawa.

I want to thank Martin Gilchrist, former Senior Editor at Springer-Verlag, for his positive feedback on the initial proposal and Bill Sanders, Senior Editor, for his continuous and enthusiastic support for the project. I would also like to express my appreciation to the staff at the Springer production department for converting my LaTeX files into the book in front of you. Some people directly involved with the book are Fred Bartlett, Anthony Guardiola, and Chrisa Hotchkiss.

I also express my appreciation to the School of Computer Science and Carleton University for providing a great atmosphere to complete this project.

Feedback

Works of this nature are never error-free, despite the best efforts of the authors and others involved in the project. I welcome your comments, suggestions, and corrections by electronic mail.

Ottawa, Canada
July 1998

Sivarama Dandamudi
sivarama@scs.carleton.ca
http://www.scs.carleton.ca/~sivarama