# COMPUTER ORGANIZATION AND THE MC68000

Prentice-Hall International Editions

PANOS E. LIVADAS
CHRISTOPHER WARD

# Computer Organization and the MC68000

Panos E. Livadas
*University of Florida*
Christopher Ward
*Auburn University*



**Prentice-Hall International, Inc.**


A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632



Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-175712-1

Prentice-Hall International (UK) Limited, London
Prentice-Hall of Australia Pty. Limited, Sydney
Prentice-Hall Canada Inc., Toronto
Prentice-Hall Hispanoamericana, S.A., Mexico
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Simon & Schuster Asia Pte. Ltd., Singapore
Editora Prentice-Hall do Brasil, Ltda., Rio de Janeiro
Prentice-Hall, Inc., Englewood Cliffs, New Jersey

# Preface

## BASIC RATIONALE

In the past, the subjects of **computer organization** and **assembly language programming** have been taught as separate entities. This decision has been justified based on the observation that the computer science discipline *itself* has evolved rapidly from hardware and beyond. Thus, it was natural that during the 1970s and early 1980s computer scientists were expected to be thoroughly familiar with the internal organization of a computer and be proficient assembly language programmers. As computer science continues to grow, a wealth of new material has appeared in high-level software developments. Current trends in this arena may be found in window management software, computer aided software engineering tools, and networked applications. This abundance of new material clearly places pressure on the computer science curriculum with the result that the lower levels are squeezed. Often this results in material at the lower levels being dropped from the curriculum. This is quite appropriate, to a point. We believe, however, that some introduction to both the areas of computer organization and assembly language programming are *essential* to all computer science students. Computer organization enables computer scientists to talk intelligently about computers with engineers and appreciate the limitations imposed by conventional computers; likewise, assembly language programming is necessary for computer scientists to visualize the relationship between higher-level programming and the computer.

With these observations in mind, we have composed a single text that provides a thorough, yet brief, introduction to both computer organization *and* assembly language programming. In doing this, our goal was to effectively reduce two separate courses into one. We have attempted to harmonize the two topics by using a particular architecture, the MC68000, as a model for the hardware aspects of the text as well as the programming

component. We note at the outset that to provide a hardware realization of the MC68000 is quite beyond the scope of this introductory text. Instead, a subset referred to as SIM68 is used to demonstrate how a computer might be designed using register-level components, and this subset is also used to introduce MC68000 assembly language programming. Although this coupling is not perfect, it does provide students with the necessary insight that we seek. In achieving this goal, the task of teaching a combined course can become difficult; these issues are addressed in later paragraphs. In writing this *introductory* textbook we have attempted to address the following issues:

**1.** *Suitability for classroom use and self-instruction.* An introductory text in any subject is primarily intended for novices; therefore, it must be their first, not last, book on the subject. As a first book, it must present the topics in a complete and "detailed" fashion; issues that may seem "trivial" to the experts are not so to novices. This obviously should *not* be accomplished at the expense of an accurate and thorough treatment of the subject.

With such an approach the book becomes lengthy, but the responsibility of the instructor is to *navigate* the student through the book; therefore, he or she may cover what seems appropriate. In addition, computer science is no longer a bag of tricks; it has evolved into a highly technological *science*. Therefore, instruction should not terminate with the end of class. We hope that this text will help the student clarify and understand the answers to the "trivial" issues that may not be presented during classroom time.

**2.** *Study of computer organization.* The principal difficulty associated with many textbooks using the title *Computer Organization* is that they are not targeted at an introductory level. There are many texts suitable for students with a broad knowledge of the area. These texts implicitly assume that students are familiar with arithmetic in different bases, the concepts of bytes, bits and word alignment, and the principal components of a computer. Often, however, this is not the case. Typically students at an early stage have had only high-level programming experience, or less! It is to these students that this course is directed. Another problem with current computer organization texts is that although machine code and assembly language programming are introduced, it is in a highly artificial context. Texts frequently assume that the material will be covered elsewhere (i.e., in an assembly language programming course) and cover the concepts far too quickly for students without experience to grasp.

This book discusses computer organization at several levels thereby permitting different issues to be offered to different people. In particular, note the following topics that are presented:

**a.** *Gate level.* For students that have had previous exposure to introductory engineering the text discusses several transistor technologies used today. The purpose is to demonstrate that the mechanism underlying modern computers may be traced from the most abstract level down to the materials level. There was some deliberation whether the text should include an introduction to VLSI techniques; however, even though this material is quite interesting, it will add no essential detail to concepts already presented.

**b.** *Register level.* Although the gate level is provided, it is expected that in general the starting point for this course from the computer organization text perspective will be

at the register level. In this text, as with several others, we introduce the building blocks of the computer. We have organized the material so that students with no experience in this arena should be able to follow the presentation without difficulty. We have included sufficient material for students to design new register level components using those provided as building blocks.

**c.** *System level.* It is at the system level that we have merged the components of computer organization and assembly programming. As we shall see shortly, SIM68 is introduced as an abstract model for assembly programming. However, in the sections where computer organization is discussed, we show how portions of such an abstract computer could be realized using *only* the components discussed in earlier sections. We use SIM68 as an example to design a simple central processing unit (CPU). During this phase, we further introduce the notion of clocked sequential control and microprogramming. Because students will already be familiar with SIM68, there is a greater appreciation of the material covered.

**d.** *Network level.* We have extended the computer organization into networked computers with a discussion of the devices used to connect computers and some broader issues of networking in general. We have used the ISO/OSI model to explain the difficulties that must be overcome when networking computers; we have also provided an example network application using Internet. We consider this to be the highest level of computer organization.

**3.** *Study of assembly language programming.* A similar criticism may be leveled at most texts on assembly language programming as was discussed for computer organization. That is, they assume a greater breath of knowledge than the students possess. Specifically texts will frequently discuss device control and interrupts in later chapters without describing the relationship between these components and the rest of the system (i.e., they assume a computer organization course). There was also an emphasis during the 1980s on explaining how high-level programs are supported at the assembly language level using frames, modules, and so on. Although this is useful to some degree, the current trend toward simpler instruction sets (i.e., the RISC architectures) suggests that such support is unnecessary at the processor level. With these comments in mind, we note the following topics covered in assembly language programming:

**a.** *Computer system basics.* The material presented early in the text is common to both computer organization and assembly programming. In these sections we have attempted to explain how to manipulate numbers in different bases; an overview of a computer system, how data, code, and instructions are represented in a computer, and notions such as a **byte** of storage and **word alignment** are some of the subjects presented. The material is expressly written to be easily understood by novices.

**b.** *Simple programmers model—SIM68.* Assembly language programming is introduced using a simple subset of the Motorola MC68000 instruction set known as SIM68. The programmer's model contains sufficient detail to permit students to write programs using the SIM68, which may be demonstrated using the SIM68 simulator freely available from the authors. Through use of SIM68, many of the concepts associated with both machine code organization and assembly language programming

may be discussed without the complexities inherent in the full MC68000 model. In addition to the programming aspect of SIM68, the computer architecture sections use the same instruction set to demonstrate how, using components already defined, SIM68 could be realized in hardware. This coupling provides additional interest for both computer organization and assembly language programming.

c. *Programming on the MC68000.* Once SIM68 has been introduced, it becomes an easy transition to move to the MC68000. Students have already had exposure to different addressing modes and instruction types via SIM68. The MC68000 is simply an extension of these concepts (albeit a large extension). Later in the text we also introduce the notion of subroutines, the frame concept, and exception processing. The latter issue clearly ties into computer organization.

d. *Controlling devices.* An excellent demonstration of reasons why we have coupled computer organization and assembly language programming may be observed in the sections on device control. We present these devices from the programmers perspective; however, the diagrams of the devices and a notion of how they could be implemented will be carried by the students from the organization section.

This book is the outgrowth of four years of class notes and lectures, and has been designed primarily for a course in computer science to address the more elementary issues in *computer architecture* as described by Peter Denning et al. in "Computing as a Discipline," *Communications of the ACM,* vol. 32, no. 1, January 1989, pp. 9–23. The text does not break new ground; it does, however, permit the entire lower-level material in architecture and assembler programming to be taught from a single text, in either one or two semesters depending on the material covered.

## ORGANIZATION OF THIS BOOK

Chapter 1 serves as an introduction to the topics that are covered in the later chapters of the book. First we identify, by means of examples, the issues involved in computer organization. We then proceed to present the basic foundation of computer system material as the three "onions" that correspond to the **hardware, software**, and **user** views. It is the combination of these views around which the remaining text is based.

Chapter 2 contains the "foundations" for the remaining chapters and includes the following sections:

1. *Number systems.* A thorough discussion is given, including addition, subtraction, multiplication, and division in different bases and one's complement, two's complement, and sign-magnitude representations.
2. *Boolean logic.* This section includes boolean algebra, operator representations, normal forms, and expression simplification using Karnaugh maps.
3. *Gate implementation.* This provides information regarding how gates are implemented in different technologies and includes the characteristics of different technologies and packing methods.

4. *Clocks and timing issues.* This introduces the notion of a clock and why one is necessary.
5. *Combinational and sequential circuits.* This section includes all the components used later in the hardware design of SIM68.

The material covered in Chapter 2 will be determined by the student's previous exposure. In the interest of completeness, however, we have included everything that we think is necessary to teach the remaining material.

Chapter 3 discusses the principal components of a computer system. It is written so that it may be taught without reference to the hardware considerations discussed in Chapter 2.

Chapter 4 describes the programmer's view of SIM68 and introduces the student to machine language programming and assembly language programming using a very simple instruction set. This material, along with the fundamentals, provides the necessary motivation for the next chapter.

Chapter 5 describes the register-level building blocks used in computers including registers, encoders, decoders, multiplexers, memory, and arithmetic and logic units. The chapter then proceeds to demonstrate how SIM68 might be constructed from these components while introducing the student to register description languages, clocked sequence logic, and microprogramming. The chapter concludes with a discussion of the MC68000, MC68010, MC68020, MC68030, MC68040, and MC88100 (RISC) processors and is intended to be used for supplemental (or "advanced") reading.

Chapter 6 introduces addressing modes as a general issue and then introduces the MC68000 from the programmer's perspective. We have attempted to first explain addressing modes without specific mention of MC68000. It is hoped that this "preview" of addressing and a discussion of why addressing modes are useful will allow an easier grasp of the plethora of addressing modes provided by the MC68000. After discussing the MC68000 we conclude the chapter by examining the more advanced addressing modes as found in the MC68020, MC68030, MC68040, and RISC architectures. This material is intended as an "advanced" section. Again this chapter is written so that architecture is not a required component.

Chapters 7 to 9 contain a full treatment of the various instructions, modes, and operand sizes of the MC68000. Chapter 7 presents the majority of the instructions and is most detailed. The chapter introduces instructions as follows: binary integer arithmetic, moving data around, branching and looping, logical and bit operations, and decimal arithmetic. In Chapter 8 we extensively discuss subroutines, the frame concept for high-level programming support, introduce the concepts of static and dynamic libraries, and explain macroprocessing. Chapter 9 discusses exception processing including internal and external interrupts. These chapters contain the bulk of the assembly language component of the text. As with earlier chapters, we also include some "advanced" topics related to the MC68020, MC68030, MC68040, and RISC processors as appropriate.

Chapter 10 contains device control and memory management. We provide a detailed explanation of a simple serial port controller, a parallel port controller, and a timer. These explanations include code examples. Also discussed are component-level diagrams of more complex DMA devices including disk-drive and Ethernet controllers. Through these

demonstrations and discussions, we hope that the student will at least be able to understand the concepts of device drivers and device control, which they might encounter in later courses (e.g., operating systems). The section on virtual memory, although not really an external "device" is included as an "advanced" topic.

Chapter 11 contains an introduction to computer networks. The authors view this area, in particular, as an extension of single-system computer organizations. Included in this chapter is a brief history of computers and how networks fit into this history; a simple taxonomy for networks; an introduction to the ISO/OSI model; an example using the Internet; and how computer networks are reaching into the domestic environment via ISDN.

It should be clear from the previous paragraphs that this text provides both an introduction to computer organization and to assembly language programming. By appropriate use of this text, an instructor may cover either one, the other, or both of these components depending on curriculum requirements and how familiar this material is to the student. We believe that this (re)integration of these subjects will satisfy CIS requirements without taking the time that completely separate courses would require.

## USING THIS BOOK AS A TEXTBOOK

This book has been used for the last four years as the main text in a three-credit semester course, Introduction to Computer Organization, at the Department of Computer and Information Sciences at the University of Florida. The students enrolled in this class are both graduate and undergraduate students in Engineering, Liberal Arts and Sciences, as well as Business students. The course is meant for those who have acquired respectable programming skills, knowledge of at least one modern high-level language (preferably Pascal or C). Exposure to courses such as Data Structures and Operating Systems would help to a certain degree, but, they are not prerequisites.

It is our experience that it is very difficult for an instructor to cover the entire book in class. Instead, we prefer to present the most important topics in class and assign other parts of the book as background reading. Certain sections can be omitted altogether. The issue of what may be considered an important topic depends both on the background of the students and what the instructor thinks is important. In our case we rapidly cover the material in Chapters 1 and 2 with the exception of some boolean algebra, gates, and minimization. We then cover Chapters 3 and 4 completely. At this point there is plenty of additional material that may be assigned if required. Also, the students are now in a position to begin writing programs using SIM68. We always assign one machine code program. After Chapter 4 we cover Chapters 6 to 8. This block represents the MC68000 component and allows us to assign successively more complex programs. We then cover Chapter 5, which overlaps with their assignments. Finally we discuss the computer connected to the outside world with Chapters 9 to 11. Again Chapters 9 and 10 allow us to assign simple device control exercises, and Chapter 11 is covered in class.

Because we believe that the students learn programming by writing assembly programs, we give three to four programming assignments during the semester. Each of these assignments deals with some aspect related to the course. The early assignments are directed toward table manipulation and simple arithmetic operations; the later assignments are "real projects" such as writing portions of a SIM68 assembler in MC68000 code. This

provides the students with a link between the hardware (computer organization) and the software (assembly language programming). We make use of Motorola's single-board computers in a lab; however, we have a complete portable software environment (described in Appendix F) that allows students to write, assemble, and "run" (via simulation) MC68000 programs on many platforms including the IBM PC, the Unix environment, and the Amiga. This environment is available at no cost from the authors.

## ACKNOWLEDGMENTS

Many people have been helpful and supportive during the period we have been writing this text. First, we would like to thank the following for their technical assistance: Andy Wilcox, for keeping things running smoothly(!); Steve Croll, for his assistance in writing the MC68000 assembler and several program segments; Wayne Wolfe, for developing the MC68000 simulator; Laura Allen, for including a disassembler and breakpoint support into the simulator; Tom Hain, for his work on the SIM68 hardware design; Debra Livadas and Joseph Vice for their excellent preliminary editing; and the many other consultants for their input. We would next like to thank our wives Debra and Kathleen who have given support and understanding throughout, have made many useful suggestions, and have stimulated extensive discussions about topics throughout the text. We would also like to thank the following reviewers for their thoughtful comments: James F. Peters III, University of Arkansas; Ron McCarty, Behrend College; Charles T. Zahn, Pace University; and John McCabe, Manhattan College. Finally, we would like to thank our many students (who used the first, as well as subsequent, drafts of this text) for their help in identifying areas of the book that needed more clarification and for their support.

# Contents