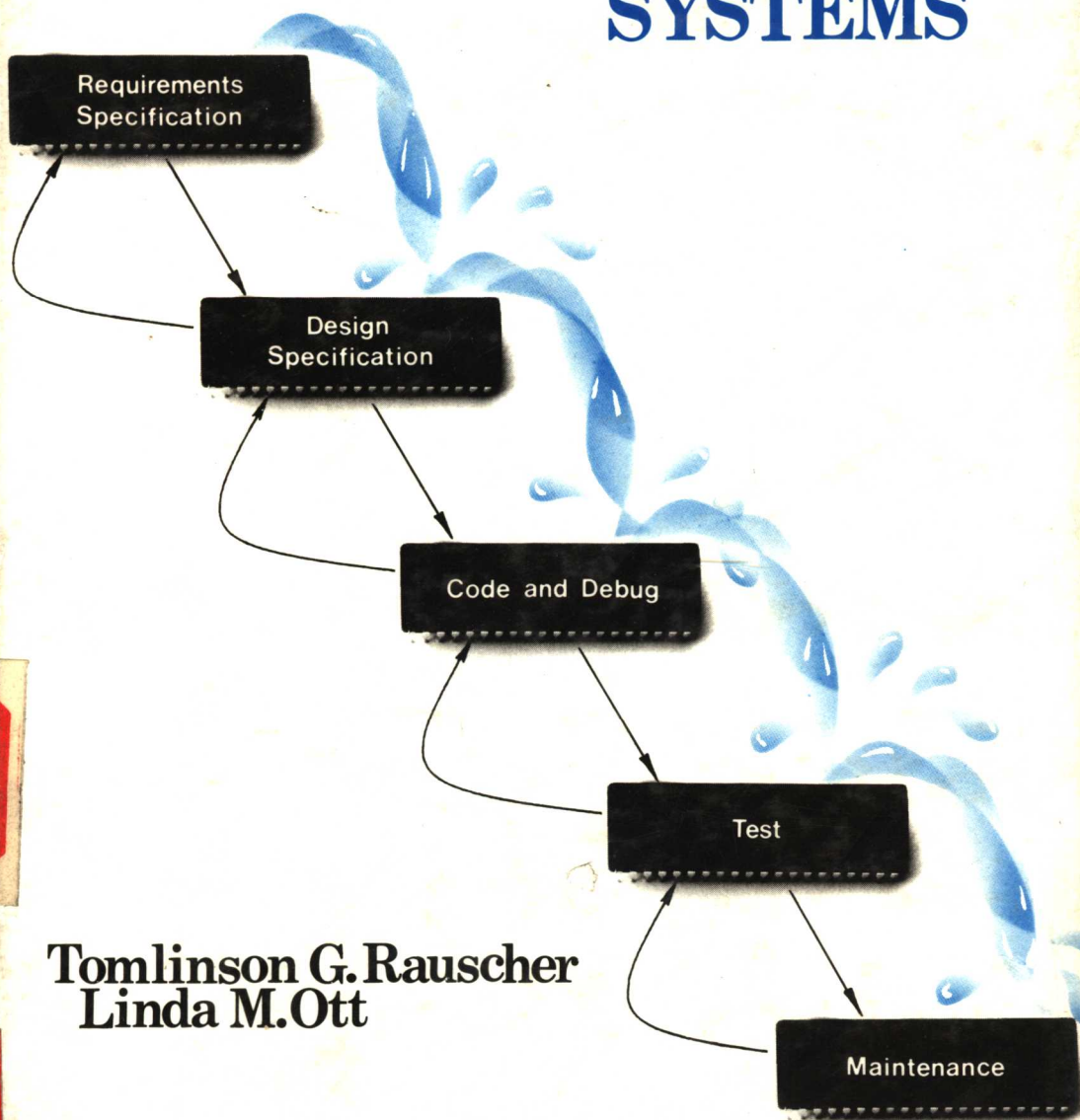


SOFTWARE DEVELOPMENT and MANAGEMENT for MICROPROCESSOR-BASED SYSTEMS



Tomlinson G. Rauscher
Linda M. Ott

**Software Development
and Management For
Microprocessor-Based
Systems**

Tomlinson G. Rauscher

*Xerox Corporation
Rochester, New York*

Linda M. Ott

*Michigan Technological University
Houghton, Michigan*

Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

RAUSCHER, TOMLINSON G.

Software development and management for
microprocessor-based systems.

Includes index.

1. Computer software—Development.
2. Microprocessors—Programming. I. Ott,

Linda M., 1950-19 II. Title.

QA76.76.D47R38 1987 005.26 86-30477

ISBN 0-13-822933-3

Cover design: *Photo Plus Art*

Manufacturing buyer: *Gordon Osbourne*

To David, Tasha, April, and Nathan

© 1987 by Prentice-Hall, Inc.

A Division of Simon & Schuster

Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-822933-3 025

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*

PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*

EDITORIA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

PREFACE

The purpose of our book is to describe a set of techniques for developing software for large microprocessor-based systems, and for managing such development activities. The book is practical in its orientation; it describes a particular set of techniques that have demonstrably improved the effectiveness of microprocessor development and management.

The topics of microprocessor software development and management have grown in interest for several reasons:

- o Microprocessors are being used in a large and increasing number of applications to provide greater functionality at a lower cost: consumer products, appliances, automobiles, industrial control, personal computers, office equipment, and so forth.
- o The software systems for microprocessors in such applications are increasing in size, complexity, and sophistication, thus creating a demand for more effective software development and management.
- o Many software developers of microprocessor systems have little training in developing the large software systems that typify modern microprocessor-based products.
- o Many managers of microprocessor-based system products have little knowledge or experience in managing large software projects.
- o The software development cost for microprocessor-based systems is large (often tens or hundreds of man-years), typically larger than the cost of developing the electronic hardware on which the software runs, and almost always larger than it should be.

Our book is intended for two principal audiences:

- o Students at the upper class or graduate level who study microprocessor software in electrical engineering, computer

engineering, computer science, or related programs of study. Such students seldom have the opportunity to work on large projects, and thus lack the knowledge of a practical development environment to which many of them will graduate.

- o Industrial practitioners who want to develop and manage significant microprocessor software activities. These professionals may have experience writing small programs on microprocessors or writing programs for large data processing systems. This basis coupled with the information presented in our book provides strong capabilities for developing and managing microprocessor-based systems.

We have used material in this book with hundreds of people in these two groups, effecting substantial improvements in personal and corporate capabilities.

We approach the subject of microprocessor software development assuming that the reader has some knowledge of microprocessors and programming. Readers should have a working knowledge of microprocessors, have written small assembler language programs for a particular microprocessor, and have done small programming projects with a higher level language to understand the fundamentals of programming concepts. There are a number of books that address such topics; our book is a second book addressing advanced topics for large microprocessor-based systems. Using introductory knowledge as a starting point, we describe in Chapters 1 through 10 specific techniques for engineering significant microprocessor software systems. The techniques cover the phases of software engineering – requirements specification, design specification, coding and debugging, testing, and maintenance – concentrating on microprocessor applications with emphasis on real-time aspects. To illustrate techniques, the examples used are short and address typical microprocessor applications, so that they can be grasped quickly. The book describes specific techniques that address the entire software development life cycle for microprocessor applications, in contrast to surveying a large number of techniques concentrating on one phase of software development, or orienting the techniques toward business data processing applications.

In a similar manner, we approach the subject of microprocessor software management assuming that the reader has some knowledge of management activities such as general project planning, PERT methods, task assignments, and delegation of authority. In the second part of the book, Chapters 11 through 19, we address software management for microprocessor-based systems by extending relevant aspects of traditional managerial concepts such as planning, staffing, organizing, and reviewing

to facilitate developing software for microprocessor-based products. Facilities required for developing microprocessor software are described. The special problems of managing embedded microprocessor software are examined in detail. Recommendations are given for developing software technology from both the practical aspects of architecture, tools, and techniques, and the personal aspects of professionals who constitute a staff. The book concludes with software management lessons that address problems typically encountered when microprocessors play a significant role in the product development repertoire in a company. The software development of a real microprocessor-based product in a large company is reviewed and critiqued.

In writing the book we have incorporated several concepts to increase its utility to students and professionals. The book uses a practical technique orientation to software development and management for microprocessors, based on real product development experiences; it is not a survey of theoretical concepts. The book concentrates on microprocessor examples; it is not a general data processing book. It addresses life cycle aspects of software development and management, not just the details of a methodology to support one phase of product development. It describes techniques that are integrated with one another throughout the development process; it is not a collection of independent articles addressing specific issues within the development process. The book is organized to present material in the order engineers and managers will be comfortable reading, building on coding and debugging knowledge and using this as motivation for design, requirements, testing, and maintenance. As the coding and debugging phase of software development is the phase with which engineers and managers are the most familiar, we present it first, even though it is neither the first phase nor the most important phase in microprocessor software development. With a thorough understanding of coding and debugging, readers will better understand the need for the design specification phase and the requirements specification phase, the most important phase in software development. The book also includes a chapter that summarizes techniques for microprocessor development in chronological order (Chapter 10) and a chapter that summarizes specific management directions (Chapter 19).

Many people assisted us in the preparation of the book. Steve Palumbo, Dan Auman, and John Kemp of Xerox Corporation helped with the preparation of several figures on software design. William Stumbo of Xerox assisted in the preparation of the index. Andrea and Roger Hopkins of Publishase prepared many of the figures. Jeanne Paige of Xerox Corporation typed several drafts of the book over an extended period; her

patient competence made preparation of the manuscript enjoyable. Marie Vasapolli and Nancy Taylor of Xerox made noteworthy contributions as well. Finally we would like to thank our colleagues at Xerox Corporation and Michigan Technological University for their encouragement and support during this work. Their reviews and comments improved not only the presentation but the substance of the book as well. Michael Nekora of Xerox made several insightful comments, which we hope we have conveyed to the reader.

CONTENTS

CHAPTER 1 – INTRODUCTION TO MICROPROCESSOR SOFTWARE SYSTEMS DEVELOPMENT	1
1.1 The Role of Software in the Microprocessor Revolution	1
1.2 The Software Development Life Cycle	4
1.3 Microprocessor Software System Management	9
1.4 Anatomy of Software Systems for Microprocessors	10
1.5 Introducing the Rest of the Book	14
CHAPTER 2 – CODING MICROPROCESSOR SOFTWARE	15
2.1 Introduction	15
2.2 Selecting a Programming Language	15
2.3 Coding Techniques – Introduction	19
2.4 Coding Techniques – Presentation	19
2.5 Coding Techniques – Algorithm Implementation	26
CHAPTER 3 – DEBUGGING MICROPROCESSOR SOFTWARE	37
3.1 Introduction	37
3.2 Debugging Techniques	38
3.2.1 Instrumentation	38
3.2.2 The Debugging Process	39
3.2.3 Debugging Perspectives	46
3.3 Coding and Debugging Tools	47
CHAPTER 4 – DESIGN PRESENTATION FOR MICROPROCESSOR SOFTWARE	52
4.1 Introduction	52
4.2 The Role of the Design Phase	52
4.3 Phases of Software Design	54
4.4 Design Phase Output	57

4.5 A Note on Terminology	57
4.6 Design Presentation Overview	59
4.7 Graphical Displays	59
4.7.1 System Flow Diagrams	59
4.7.1.1 Data Flow Diagrams	60
4.7.1.2 Finite State Machines	64
4.7.2 System Structure Diagrams	66
4.7.2.1 Hierarchy Charts	66
4.7.2.2 Structure Charts	69
4.8 Textual Documentation of Design	72
4.8.1 Program Design Languages	73
4.8.2 The PDL Language and Processor	73

CHAPTER 5 – DESIGN TECHNIQUES FOR MICROPROCESSOR SOFTWARE 91

5.1 Introduction	91
5.2 Design Analysis	92
5.2.1 Analysis Using the Finite State Machine	92
5.2.2 Analysis Using the Data Flow Diagram	94
5.2.3 Analysis in Perspective	96
5.3 Design Synthesis – High Level Design	98
5.3.1 The High Level Design Process	98
5.3.2 Transform Analysis	100
5.3.3 Transaction Analysis	101
5.3.4 Coupling	104
5.3.5 Cohesion	106
5.3.6 Fan-Out and Fan-In	107
5.4 Design Synthesis – Detailed Design	108
5.4.1 The Detailed Design Process	108
5.4.2 Developing the Physical Program Structure	109
5.4.3 Describing the Top Modules in the Physical Program Structure	110
5.4.4 Completing Module Descriptions	112
5.5 Design in Perspective	113

CHAPTER 6 – BASIC CONCEPTS OF REQUIREMENTS SPECIFICATION AND ANALYSIS FOR MICROPROCESSOR SOFTWARE 116

6.1 Introduction	116
------------------	-----

6.2	Characteristics of Requirements Specifications	118
6.2.1	Fundamental Characteristics of Requirements Specifications	118
6.2.2	Readability Characteristics of Requirements Specifications	119
6.2.3	Ease of Writing Characteristics of Requirements Specifications	119
6.2.4	Feasibility Characteristics of Requirements Specifications	120
6.3	Users of Requirements Specifications	122
6.4	Requirements Specification Process and Its Results	124
6.5	Requirements Analysis	129

CHAPTER 7 – SYNTHESIS OF REQUIREMENTS SPECIFICATIONS FOR

	MICROPROCESSOR SOFTWARE	130
7.1	Synthesis of the Functional Requirements	130
7.2	Writing the Customer Requirements Specification	131
7.3	Writing the User Manual	134
7.4	Synthesis of the Software Requirements	136
7.4.1	Introduction	136
7.4.2	Stimulus-Response Sequences	141
7.4.3	Other Requirements Specification Techniques	144
7.4.4	Recommendations on Preparing Software Requirements	147
7.5	Synthesis of the Subsystem Software Requirements	147

CHAPTER 8 – TESTING MICROPROCESSOR SOFTWARE

8.1	Introduction	149
8.2	The Testing Process	151
8.3	Test Planning	151
8.3.1	Testing Phases	151
8.3.2	Approaches to Integration Testing	152
8.3.3	Documenting the Test Plan	155
8.4	Test Case Selection	156
8.4.1	Glass Box Testing	157
8.4.2	Black Box Testing	160
8.4.3	Testing by Program Modification	161
8.4.4	Test Case Documentation	162
8.5	Test Execution	162
8.6	Test Review	163

8.7 Test Termination Criteria	164
8.8 Testing Tools	166
8.9 Testing in Perspective	168
8.10 Summary of Testing Principles	168

CHAPTER 9 – MAINTAINING MICROPROCESSOR SOFTWARE 170

9.1 Overview of Software Maintenance	170
9.1.1 Maintenance of Software in a Traditional Environment	170
9.1.2 Maintenance of Software in a Microprocessor Environment	172
9.2 Reducing the Need and Time for Software Maintenance	172
9.3 The Maintenance Process	174
9.4 Managing Software Maintenance	176

CHAPTER 10 – SUMMARY OF MICROPROCESSOR SOFTWARE DEVELOPMENT 178

10.1 Introduction	178
10.2 Requirements Specification	178
10.3 Design Specification	180
10.4 Coding	183
10.5 Debugging	185
10.6 Testing	188
10.7 Maintenance	189

CHAPTER 11 – INTRODUCTION TO MANAGING MICRO-PROCESSOR SOFTWARE DEVELOPMENT 191

11.1 Preparation for Managing Microprocessor Software Development	191
11.2 Planning Microprocessor Software Development Projects	192
11.3 Organizing Microprocessor Software Development Projects	193
11.4 Actuating Microprocessor Software Development Projects	194
11.5 Controlling Microprocessor Software Development Projects	195

**CHAPTER 12 – PLANNING MICROPROCESSOR SOFTWARE
DEVELOPMENT PROJECTS 197**

- 12.1 Introduction 197
- 12.2 Initial Project Scheduling and Costing 198
- 12.3 Detailed Project Plan 208
- 12.4 Pitfalls to Avoid When Planning 214

**CHAPTER 13 – STAFFING MICROPROCESSOR
SOFTWARE DEVELOPMENT PROJECTS 217**

- 13.1 Introduction 217
- 13.2 Recruiting 217
 - 13.2.1 Sources 218
 - 13.2.2 Résumés 222
 - 13.2.3 Interviews 223
- 13.3 Development 225
 - 13.3.1 Motivation 225
 - 13.3.2 Accessibility of Technical Information 229
- 13.4 Evaluation 229

**CHAPTER 14 – ORGANIZING MICROPROCESSOR
SOFTWARE PROJECTS 231**

- 14.1 Organization Theory 231
 - 14.1.1 Functional Organizations 231
 - 14.1.2 Project/Product Organizations 232
 - 14.1.3 Matrix Organizations 233
- 14.2 Organization of Programming Teams 234
- 14.3 Software Organization Pitfalls to Avoid 237
- 14.4 Improvement of the Software
Organization Team Members 238

**CHAPTER 15 – REVIEWING MICROPROCESSOR
SOFTWARE PROJECTS 240**

- 15.1 Introduction 240
- 15.2 Phase Reviews 240
- 15.3 Formal Reviews 242
 - 15.3.1 Characteristics and Objects of Formal Reviews 242
 - 15.3.2 Roles of Review Team Members 243
 - 15.3.3 Selecting the Review Team 244
- 15.4 Informal Reviews 246
- 15.5 Role of Management in the Review Process 246

**CHAPTER 16 – FACILITIES FOR MICROPROCESSOR
SOFTWARE PROJECTS 248**

- 16.1 Introduction 248
- 16.2 Computer Facilities 250
 - 16.2.1 The Purpose of Computer Facilities 250
 - 16.2.2 Computer Hardware Facilities 252
 - 16.2.3 Computer Software Facilities 256
 - 16.2.4 Integrated Computer and Support Systems 260
- 16.3 Office Facilities 262
 - 16.3.1 The Purpose of Office Facilities 262
 - 16.3.2 Offices 263
 - 16.3.3 Office Complexes 264
 - 16.3.4 Perspective on Office Facilities 266

**CHAPTER 17 – INTEGRATING MICROPROCESSOR
SOFTWARE INTO EMBEDDED PRODUCTS 268**

- 17.1 Motivation for Integration 268
- 17.2 Technical Direction in Integrating
Microprocessor Software into Embedded Products 269
 - 17.2.1 Organizational Interfaces 269
 - 17.2.2 Requirements Specification 270
 - 17.2.3 Design Specification 271
 - 17.2.4 Coding and Debugging 271
 - 17.2.5 Testing 272
 - 17.2.6 Maintenance 274
- 17.3 Interaction with Management in Integrating
Microprocessor Software into Embedded Products 275
 - 17.3.1 Organizational Interfaces 275
 - 17.3.2 Requirements Specification 275
 - 17.3.3 Design Specification 276
 - 17.3.4 Coding and Debugging 276
 - 17.3.5 Testing 278
 - 17.3.6 Maintenance 278

**CHAPTER 18 – DEVELOPING MICROPROCESSOR SOFTWARE
ENGINEERING TECHNOLOGY 279**

- 18.1 An Historical Viewpoint 279
- 18.2 Developing Microprocessor Software Methodology 280
 - 18.2.1 The Fundamental Concept
of the Software Life Cycle 280

18.2.2	Creating the Demand for Knowledge of the Software Life Cycle	282
18.2.3	Developing Knowledge of the Software Life Cycle	283
18.2.4	Adapting the Software Life Cycle to Particular Applications	283
18.2.5	The Role of Documentation	286
18.3	Developing Microprocessor Software Architecture	287
18.3.1	The Hardware/Software Classification	287
18.3.2	The Software Breakdown – Operating System, Diagnostics, and Applications	289
18.3.3	Tradeoffs in Operating System Design and Development	292
18.3.4	Tradeoffs in Diagnostic Design	294
18.3.5	Tradeoffs in Application Software Design	297
18.4	Developing Microprocessor Software Tools	297
18.5	Developing Microprocessor Software Personnel	301

CHAPTER 19 – SOFTWARE MANAGEMENT LESSONS OR "WHY DOES SOFTWARE COST SO MUCH AND TAKE SO LONG?" 304

19.1	The Problem	304
19.2	The Causes	306
19.2.1	The Fundamental Tenet – Management Is the Problem	306
19.2.2	Managers Lack Software Knowledge	306
19.2.3	Managers Do Not Develop Viable Plans	307
19.2.4	Managers Do Not Staff Projects Properly	309
19.2.5	Managers Do Not Organize Projects Properly	311
19.2.6	Managers Do Not Provide Facilities That Optimize Productivity	312
19.2.7	Managers Do Not Monitor Projects Properly	313
19.2.8	Managers Do Not Select the Proper Technology	314
19.3	Project Z – A Project That Cost Too Much and Took Too Long	315
19.4	Improving Software Management	318

ANNOTATED BIBLIOGRAPHY 321

INTRODUCTION TO MICROPROCESSOR SOFTWARE SYSTEMS DEVELOPMENT

1.1 The Role of Software in the Microprocessor Revolution

The decade of the seventies will surely be remembered for witnessing the commercial introduction and utilization of microprocessors. Not only did their deployment revolutionize the use of electronics in a multitude of applications, but their continued improvement demonstrated incredible technological maturation. From the simple characteristics of the first four-bit microprocessors to the sophisticated features of modern sixteen-bit and thirty-two-bit microcomputers, the capabilities provided by these microelectronic marvels have increased dramatically. While capabilities have increased, the cost of employing these "computers-on-a-chip" has, surprisingly, decreased when examined on a cost-per-function basis. The increasing availability of capable, inexpensive microprocessors has facilitated the development of increasingly sophisticated systems, ranging from simple calculators to real-time systems that simultaneously support several activities and input/output devices.

The ability of microprocessors to support complex applications has been made possible not only by improved hardware capabilities such as greater functionality of instructions, larger on-chip memories, and faster instruction execution times, but also by larger, more powerful software systems. Whereas the size of early microprocessor programs ranged only up to a few thousand bytes, modern systems frequently reach fifty thousand

bytes and more. The effort involved in developing software systems of such capabilities requires technological knowledge, engineering discipline, and management direction. The development of these capabilities often proceeds at a slower rate than that of the microprocessor hardware technology. When this is coupled with the realization that developing large software systems for microprocessors is exponentially more complex than developing small systems (with which people typically experience some success in initial attempts), it is easy to see why software development has been criticized for products that are expensive, late, and unreliable.

Microprocessor software systems development need not deserve this reputation. With the appropriate application of engineering and management tools and techniques, microprocessor software systems development can be a rewarding and successful enterprise. The purpose of this book is to assist in the accomplishment of this goal.

It is appropriate to examine the historical background which has led to the poor opinion in which software is often held today. Early microprocessor software development activities tended to emphasize the differences from traditional software development, citing justifications such as

- o Close functional relationship to hardware being replaced
- o Small program sizes
- o Use of low-level languages, particularly machine and assembler languages

The current trend in microprocessor systems deviates from these viewpoints:

- o Microprocessors are being used where flexibility and expandability are required; specially designed VLSI chips and "programmable logic arrays" are now being used to replace hardware of moderate complexity where microprocessors were formerly used.
- o As mentioned earlier, large programs are as much the rule as the exception.
- o Higher level languages can, should, and are being used in most microprocessor applications.

It is useful to note, however, that microprocessor software often exhibits differences from the typical data processing programs, which constitute a significant portion of traditional software systems. These differences include the following:

- o Microprocessor software frequently represents only a small part of a system composed of several electrical, mechanical, and other components. The end user need not (and often is not) aware that a microprocessor is integrated in the system.
- o Microprocessor manufacturers seldom supply extensive support system software (although this trend is changing), and, as a result, developers frequently must design and implement all the microprocessor software for their system. This ranges from low-level operating system tasks such as interrupt and I/O handling to high-level application specific tasks.
- o Microprocessors often support real-time systems, wherein immediate response to asynchronous external events characterize normal requirements.
- o The principal memory for program storage in many microprocessor systems is read-only memory (ROM), due to its cost savings over other memory types, such as electrically programmable ROM (EPROM) and read/write random access memory (RAM). The cost per system aspect is especially important when considering that microprocessor products often have sales or usage volumes in the tens and hundreds of thousands (such as calculators, automobiles, cash registers, and games).
- o Another aspect of products with imbedded microprocessors is that companies often produce product families in which microprocessor use may vary in only minor ways among products in the same family.

These variations on traditional software systems result in the use of tools and techniques that are peculiar to microprocessor software development. Much of the development activity for microprocessor software, however, adapts techniques used in traditional software development.

It has been interesting to follow the technology for developing software for microprocessor systems because history has, to a large extent, repeated itself. The use of machine language, assemblers, macroassemblers, compilers, and debuggers bears a strong resemblance to their development for larger computers two decades earlier. The reasons for this are technological and organizational:

- o The first developers of microprocessors were technologists whose principal expertise was in microelectronics, not software.