

**ADVANCES
IN
COMPUTER
ARCHITECTURE**

**Second
Edition**

GLENFORD J. MYERS

| Advances in
Computer Architecture
Second Edition

BIBLIOTHEQUE DU CERIST

OTHER BOOKS BY GLENFORD J. MYERS

Reliable Software Through Composite Design, 1975
Software Reliability: Principles and Practices, 1977
Composite/Structured Design, 1978
Advances in Computer Architecture, First Edition, 1978
The Art of Software Testing, 1979
Digital System Design with LSI Bit-Slice Logic, 1980

Advances in Computer Architecture

Second Edition



GLENFORD J. MYERS
Intel Corporation
Santa Clara, California



A WILEY-INTERSCIENCE PUBLICATION
JOHN WILEY & SONS

New York Chichester Brisbane Toronto Singapore

Copyright © 1978 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

Library of Congress Cataloging in Publication Data:

Myers, Glenford J., 1946—
Advances in computer architecture.

4782

"A Wiley-Interscience publication."

Includes bibliographical references and index.

1. Computer architecture. I. Title.

QA76.9.A73M93

1981

621.3819'52

81-11374

ISBN 0-471-07878-6

AACR2

Printed in the United States of America

TO MY PARENTS

BIBLIOTHEQUE DU CERIST

Preface to the Second Edition

In the three years between the publication of the first edition of this book and the writing of this edition, important advances in computer architecture have occurred. These advances were motivated by a number of factors, including the arrival of VLSI (very-large-scale integration) design and manufacturing capabilities, renewed interest in programming-language improvements (e.g., Lisp, Ada), and increasing agreement with the issues and concepts discussed in the first edition of this book (e.g., tackling the software-development problem through improved computer architectures). These advances have appeared in such systems as Intel's iAPX 432 microprocessor and IBM's System/38, and in the implementation of the SWARD machine (an early version of which was described in the first edition).

I was motivated to write this second edition by the development of these new systems, new work in such other areas as data-flow and database machines, a desire to broaden and elaborate upon the ideas put forth in the first edition, and my personal experiences in the hardware and software implementation of the SWARD system.

The organization of the second edition parallels that of the first. Part I (Chapters 1–4) has been largely rewritten, and now contains more quantitative information and, I hope, more cogent arguments. Examples from the Motorola 68000 and IBM System/38 are used in Chapter 4. The case studies in Parts II, III, and IV have been retained from the first edition, although a number of inaccuracies were corrected. Part V, the case study of the SWARD machine, was completely rewritten to reflect the current state of this architecture. Part VI is a new case study, the Intel iAPX 432, a revolutionary microprocessor architecture. Part VII is an expansion of the material in the first edition on database machines. Also

added was a chapter on data-flow machines. Finally, Part VIII contains some general information on the process of computer architecture.

I thank several individuals whose efforts contributed to this second edition, namely, George Cox, Justin Rattner, and Ken Aupperle of Intel and Dave Ditzel of Bell Laboratories.

GLENFORD J. MYERS

Santa Clara, California
October 1981

| Preface to the First Edition

Since the 1950s, we have witnessed many advances in computing systems. The software field has advanced tremendously; for instance, we now have better tools, methodologies, and programming languages, software applications are more sophisticated, new algorithms have been invented, and the construction of such programs as operating systems and compilers is fairly well understood. The construction of physical computing devices has also advanced significantly; for example, circuit speeds and densities have increased by orders of magnitude, new storage technologies have been invented, better algorithms have been devised, and the microprogramming concept has been exploited. However, we have seen almost no advances at the hardware/software interface, the level of a system usually referred to as the *computer architecture*. To be fair, there have been some significant advances, but they have not received widespread attention and have not found their way into most conventional systems. For instance, if the instruction sets of most current large-scale systems, minicomputers, and microcomputers are examined, they will be found to be strikingly similar to those of machines designed in the 1950s.

The similarity of the architecture of today's systems to earlier systems can cause us to become complacent about the subject; we look around us and see tremendous software and hardware advances, but see that the architectures of current systems are virtually the same as those of earlier systems, so we might be inclined to assume that people in the 1940s and 1950s invented all there was to be invented in the area of computer architecture. The result would be that the architecture of future systems would remain the same. This attitude is my motivation for writing this book: to destroy this complacency by showing that there are serious

problems in current computer architectures and discussing advanced architectural concepts that will solve these problems.

The intent of this book can also be expressed by examining two possible alternative titles that were considered. One title was *Fifth-Generation Computer Architectures*. The term "fifth" was selected because of the feeling that the fourth generation is already on the drawing board and that these systems would undoubtedly retain the architecture of earlier systems. This title was discarded, however, because it looked too "flashy." Also it would be misleading because some of the concepts discussed in the book arose in certain second-generation systems. Another possible title was *Second-Era Computer Architectures*, but this title was discarded because of the feeling that prospective readers would confuse "era" for "generation" and form the impression that the book is a historical survey of the IBM 1401, Burroughs 200, and other "second-generation" (discrete transistor) machines.

The chapters within this book are organized into six parts. Part I defines computer architecture, takes a critical look at current architectures, and discusses a set of properties needed in future computer architectures. Parts II, III, IV, and V are case studies; they discuss four advanced architectures having many or all of the desirable properties discussed in Part I. Part VI discusses other aspects of computer architecture, such as input/output considerations and the optimization or "tuning" of an architecture.

The book is intended for two audiences: for use as a text in a "second course" on computer architecture (where the "first course" would presumably cover conventional architectural concepts), and to spread some of these ideas to computer professionals in general. The reader is expected to have a good grasp of computer system fundamentals. In particular the reader should be knowledgeable of programming language concepts (e.g., the phrase "scope of names in a block-structured language" should be meaningful to the reader), have an understanding of the machine or assembly language of a conventional machine (e.g., S/370, PDP-10, CDC 6600), be familiar with operating system and compiler concepts (e.g., the term "reverse Polish notation" should be a familiar one), and have a grasp of the concept of microprogramming. A basic premise of this book is that this knowledge is prerequisite to the development of computer architectures.

I have found that the most effective way to understand an architecture is to do a mental compilation of a high-level language program to the architecture; many of the examples in the book were developed along this line. If the book is being used as a text, the student should be assigned a number of small PL/I, Cobol, or Fortran programs to be mentally compiled to each architecture.

PREFACE TO THE FIRST EDITION

xi

Lastly, the opinions in the book are those of the author and do not necessarily represent the opinions, or future product directions, of the IBM Corporation.

GLENFORD J. MYERS

January 1978

BIBLIOTHEQUE DU CERIST

Contents

PART I THE NEED FOR ARCHITECTURAL ADVANCES

1. A Definition of Computer Architecture	3
The Role of the Computer Architect, 7	
Analyzing Architecture Performance, 9	
References, 13	
Exercises, 14	
2. A Critique of the Conventional von Neumann Architecture	15
The Semantic Gap, 17	
Consequences of the Semantic Gap, 22	
The von Neumann Architecture, 29	
Other Undesirables, 32	
References, 37	
Exercises, 39	
3. The Binding of Programs to Machines	40
Language-Directed Architectures, 43	
Type A High-Level-Language Architectures, 45	
Type B High-Level-Language Architectures, 46	
Hardware/Software Cost Trade-Offs, 47	
Type C High-Level-Language Architectures, 49	
Architectures and Compilers, 51	
References, 52	
Exercises, 57	
4. Requisites for Improved Architectures	58
Self-Defining Data, 58	
Self-Defining Data Collections, 68	

Small Protection Domains, 74
 Subroutine Management, 76
 Capability-Based Addressing, 81
 Single-Level Storage, 91
 Process Management, 98
 Instruction Forms, 106
 Higher Level Operations, 117
 Lexical-Level Addressing, 118
 References, 120
 Exercises, 123

PART II A LANGUAGE-DIRECTED ARCHITECTURE

- 5. The Student-PL Machine** **127**
- The Student-PL Language, 127
 SPLM Storage Structure, 129
 Reference, 133
 Exercises, 133
- 6. Program Compilation and Execution on SPLM** **134**
- Program Segments for IF Statements and DO Loops, 138
 Subroutine-Call Example, 141
 Significance of SPLM, 143
 Exercises, 145
- 7. SPLM Instruction Set** **146**
- Data-Access and Addressing Instructions, 147
 Data-Operation Instructions, 149
 Control Instructions, 151
 Procedure Instructions, 153
 Array-Storage Instructions, 155

PART III A HIGH-LEVEL-LANGUAGE ARCHITECTURE

- 8. System Architecture of the SYMBOL System** **159**
- System Configuration, 160
 Job Flow Through the System, 163

CONTENTS**xv**

The SYMBOL Programming Language, 164
References, 168
Exercises, 170

9. Computer Architecture of the SYMBOL System 171

Representation of Data, 171
The Name Table, 173
The Object-Code String, 178
Exercises, 184

10. SYMBOL Processor and Configuration Architecture 186

The Main Bus, 186
Memory Management, 188
The System Supervisor, 196
The Central Processor, 200
The Translator, 203
The Remaining Processors, 203
System Software, 204
Significance of the SYMBOL System, 205
Exercises, 207

PART IV A MULTIPLE-LANGUAGE-DIRECTED ARCHITECTURE**11. The Burroughs B1700 System 211**

B1700 System Architecture, 212
Implementation Considerations, 213
Storage and Performance, 214
References, 215

12. Burroughs B1700 Cobol/RPG Architecture 216

Data Types, 216
Program Parameters, 217
Storage Structure, 218
Instruction Formats, 220
Machine Instructions, 222
Reference, 233
Exercises, 233

PART V A SOFTWARE-ORIENTED ARCHITECTURE

- 13. The Rationale for SWARD** **237**
- The Design Goals, 238
 - Error-Detection Comparison, 241
 - Overview of SWARD, 243
 - References, 249
- 14. The SWARD Machine** **251**
- Data Types, 251
 - Objects, 263
 - Instruction Formats and Addressing, 267
 - Fault Handling, 271
 - Instruction Summary, 275
 - A One-Module Example, 279
 - Significance of SWARD, 284
 - The System Environment, 289
 - Exercises, 291
- 15. SWARD Instruction Specifications** **292**
- General Instructions, 293
 - Arithmetic Instructions, 296
 - Comparison-and-Branch Instructions, 298
 - Boolean Instructions, 301
 - String-and-Search Instructions, 301
 - Control Instructions, 303
 - Addressing Instructions, 306
 - Process-Machine Instructions, 311
 - Debugging Instructions, 318
 - Implementation Notes, 320
 - Process Machines and Storage, 322
 - Performance in the Small, 323

PART VI AN OBJECT-ORIENTED MICROPROCESSOR

- 16. Overview of the Intel iAPX 432 Microprocessor** **335**
- The Design Goals, 335
 - Overview of the Architecture, 336

Components and Configurations, 343
References, 344

17. iAPX 432 GDP Architecture 345

Segments and Data Types, 345
Segments and Objects, 348
Addressing, 350
Protection, 361
Program Structure, 363
Processors and Processes, 366
Interprocess Communication and Synchronization, 374
Software-Defined Objects, 377
Storage Management, 380
Fault Handling, 382
Significance of the 432, 386
References, 388
Exercises, 388

18. iAPX 432 GDP Instruction Specifications 390

Instruction Formats, 390
General Instructions, 396
Arithmetic Instructions, 398
Comparison Instructions, 401
Boolean Instructions, 403
Control Instructions, 405
Addressing Instructions, 407
Process/Processor-Related Instructions, 411
Synchronizing Instructions, 414
Use of the Operand Stack, 414
On-Chip Associative Address Translation, 415
Exercises, 417

PART VII ARCHITECTURES FOR DATABASES

19. Associative Memories 421

Addressing by Content, 422
Partial Associativity, 424
Associative Memories as Database Processors, 426
References, 427

20. The Relational Associative Processor (RAP)	429
The Cell, 430	
Data Representation, 431	
Instruction Format, 433	
Instruction Set, 434	
Program Examples, 439	
Performance, 441	
RAP.3, 442	
RAP in a Storage Hierarchy, 443	
Significance of RAP, 444	
References, 445	
Exercises, 446	
21. Other Database Machines	447
CASSM, 447	
The Database Computer (DBC), 453	
References, 461	
22. Data-Flow Architectures	463
The Data-Flow Concept, 465	
A Data-Flow Language, 467	
A Data-Flow Machine, 475	
Packet Communication Networks, 482	
Congestion and Deadlock, 486	
Structure Processing, 489	
References, 492	
Exercises, 494	
 PART VIII RELATED TOPICS	
23. Architecture Optimization and Tuning	497
Instruction-Set Optimizations, 498	
Operation-Code Optimization, 505	
Address Optimization, 512	
References, 516	
Exercises, 517	

CONTENTS	xix
24. The Practice of Computer Architecture	519
References, 525	
Answers to Exercises	527
Index	541