BOOK REVIEW

Advances in Computer Architecture

by

Glenford J. Myers
IBM Systems Research Institute

Published by Wiley-Interscience Division
of John Wiley and Sons, New York, 1978, $21.00.

Myers defines computer architecture to be the process of defining the interface between software and firmware/hardware and distributing computing system functions on both sides of this interface. He goes on to propose and discuss ten "levels of architecture" within a computing system. "The computer architect, then, makes three broad classes of decisions: the form in which programs are presented to the underlying machine, the methods with which these programs name or address their data, and the representation of data."

A major premise of this book, as stated in Chapter 1, is that most computer architects have not viewed their roles this way. If the architect views the job of architecture properly, he or she should be concerned with efficiency of problem solution, rather than the average raw speed of the machine instructions; that is, the architect must consider efficiency from the point of view of the programming-language/compiler/machine triplet. The author suggests that it would be a step in the right direction if architects for new machines were required to personally design, code, and debug a compiler and operating system for a previous machine. (An input to curriculum makers.)

In Chapter 2, in conjunction with a duly respectful critique of the von Neumann architecture, the author states that "except for a few machines (e.g. some made by Burroughs Corporation), there have been no advances in the computer architecture (as defined above) of current systems since the 1950's. This premise coupled with positive developments in high-level languages has given rise to a semantic gap between the concepts in high-level languages and the concepts in computer architecture. The authors arguments are convincingly made and well illustrated by examples.

In Chapter 3, before the chapters which present a detailed analysis of how the semantic gap might be narrowed, the author surveys previous attempts, and places them into five basic categories of strategies used. These categories are language-directed architectures, three variants of high-level-language machines, and application-directed architectures. The classifications are not mutually exclusive but are nevertheless a useful conceptualization.

After establishing the need for advances in computer architecture in Part I (Chapters 1-4), the book moves on to case studies and examples of implementations of the ideas which have been proposed. In Part II (Chapters 5-7) the architecture of the Student-PL Machine (SPLM), originally proposed in Wortman's Ph.D. thesis, is examined. SPLM is a language directed toward a subset dialect of PL/1. It contains tagged storage, descriptors of data objects, pushdown stacks for expression evaluation, subroutine management, PL/I-controlled storage, and lexical-level addressing.

Chapter 5 consists of a description of the Student-PL Language and a block-diagram machine level description of the storage structure. In Chapter 6 the reader is walked through the compilation and execution of two Student-PL programs and then offered a brief comparison to an equivalent PL/I program compiled to the S/370. Chapter 7 is a specification of the SPLM instruction set.

Part III (Chapters 8-10) is the second case study and unlike the paper machine, SPLM, it is an operational system, SYMBOL. This system, developed by Fairchild is in use at Iowa State University. The design goals for SYMBOL included achieving substantial performance increase by directly implementing a high-level language and a virtual-storage, time-sharing operating system in hardware. It provides facilities to assist the programmer in working with non numeric data and elaborate data structures. Myers classifies SYMBOL as a "type-B" variant of a high-level language architecture. SYMBOL is also interesting in that it is an ensemble of functionally-dedicated processors: one for compiling, one for dispatching and paging, etc. Myers has collected together in a concise and readable form the contents of numerous internal reports concerning SYMBOL. Due to incomplete documentation he has had to infer the structure in some cases. This section covers the system architecture, instruction set and data structure facilities of SYMBOL. It concludes with a well balanced discussion of the architectural and implementation significance of the machine.

The Burroughs B1700 System, is presented as an example of multiple-language-directed architecture in Part IV (Chapters 11-13). Although much of the material is from papers in the open literature, the author did get Burroughs' permission to include material from their B1700 COBOL/RPG S-Language Manual. There is less detail for this machine than the others. This reviewer would have valued a brief description of the machine language level (highly vertical "microcode") and processor structure of the B-1700.

In Part V (Chapters 13-15) the author offers his own contribution to an advanced

computer architecture: the SWARD (software-reliability-directed) machine, developed in his 1977 Ph.D. dissertation at Polytechnic Institute of New York. We enjoyed reading his proposal and commend it as an example of the type of thinking which should be going on in collaborative efforts between experts in computer architecture, software engineering, and reliable/fault-tolerant computing. The novelty and level of detail slow-down one's reading at this point.

Part VI (Chapters 16-18) is a collection of related topics in computer architecture, namely: input/output architecture, architecture optimization and tuning, and the "art of computer architecture." Chapter 16 briefly surveys front-and back-end processors and associative-storage processors both a la STARAN and the smart-disk approach such as proposed in the Toronto Relational Associative Processor (RAP). The optimizations mentioned in Chapter 17 relate to frequency dependent coding and the creation of new instructions which subsume a frequently occuring sequence of instructions. The book concludes with a short discussion of goals and tools that should be considered by the computer architecture.

This book includes references and a set of realistic and generally interesting exercises for each chapter. Answers to the exercises (many of them "thought questions") are provided.

We commend this book to professionals such as the readers of CAN. It should be on the required reading list for senior/graduate level courses in computer architecture, i.e. a course which presumes that the student is familiar with principles of operating systems, compilers, microcode and traditional computer organization.

D. E. Atkins
Department of Electrical and
Computer Engineering
The University of Michigan
Ann Arbor 48109