INTRODUCTION TO COMPUTER ARCHITECTURE AND ORGANIZATION

HAROLD LORIN

INTRODUCTION TO COMPUTER ARCHITECTURE AND ORGANIZATION

HAROLD LORIN
IBM Systems Research Institute, New York

1807 1982

A Wiley-Interscience Publication
JØHN WILEY & SONS
New York • Chichester • Brisbane • Toronto • Singapore

Library of Congress Cataloging in Publication Data.

Lorin, Harold.
Introduction to computer architecture and organization.

"A Wiley-Interscience publication."
Includes bibliographies and index.
1. Computer architecture. I. Title.
QA76.9.A73L67 1982 621.3819'52 82-8640
ISBN 0-471-86679-2 AACR2

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

PREFACE

This book is an introduction to computers for people who know something about computing. It is aimed at the population of computer professionals and paraprofessionals, casual users, and all others who have some view of computing but know nothing about the logical and organizational nature of a computing device.

As the computer field has matured, it, like all advancing areas of knowledge, has fragmented into a number of specialties. There are a number of very competent computer professionals who know many aspects of software specification, data base design, large application development technique, and so on, but do not know much about the instrument on which their miracles are performed. There is also a large population of computing professionals who know about the technology of computer manufacture or packaging digital technology but have never formed a complete image of the nature of the device they manufacture or whose subassemblies they design. Finally, there is the growing population of casual users whose curiosity has been piqued by their contact with computers and who would like to know more about them.

This book is for all these people. It assumes that the reader has some knowledge of languages like BASIC, FORTRAN, and COBOL, and, starting with known concepts, guides the reader through various stages of computer hardware architecture and organization. It is my opinion, however, that the slightest familiarity with the general ideas of higher-level languages is a sufficient starting point.

This book was developed over a period of years as a result of my interaction with both graduate and professional students. A number of these students reviewed the manuscript, and the extent of coverage and many specific details are the result of their comments and class interaction.

What I try to do here is discuss ideas of architecture and organization generally to provide an overview of several immortal concepts. While specific architectures and machine organizations are mentioned in the book, for example, the IBM S/370, IBM Series/1, IBM 8100, Sperry Rand UNIVAC 1100, Cray Research CRAY-1, Digital Equipment Corporation VAX-11, Data General NOVA, INTEL 8080, and INTEL 8086, the book is not intended to describe specific machines but to use them as examples of architectural and organizational concepts. The danger in using specific examples is that the concepts they embody seem to age with the machines though the concepts are in fact ageless.

There are two great challenges in undertaking a book of this type. One, of course, is content. Naturally I hope that the content is well chosen, the omissions not disqualifying, and the simplifications appropriate. Aside from content, the greatest challenge is sequence. Topics are so interrelated that it is difficult to discuss many points without anticipating later ones. Where necessary, brief characterizing introductions are offered in one area to support discussion in another. This leads to some repetition, but the idea is to give more and more detail each time a topic is discussed. The sequence of chapters was discovered dynamically in class; it has been changed many times, and I am now confident that the conceptual flow of the book minimizes the temptation for readers to look ahead.

HAROLD LORIN

New York
September 1982

CONTENTS

PART ONE

ARCHITECTURE

Chapter 1

Architecture, Organization, and Implementation

1. ARCHITECTURE

The exact meaning of the word *architecture* in the context of computers is a little uncertain. In general, architecture refers to the visible characteristics of a system as seen by a person or a program creating code capable of running on the machine. It is common, however, to use the word to mean the view of a machine shown by its *assembly language*. An assembly language is a low-level programming language in which the basic characteristics of a computer system are more directly represented than in a language like COBOL or FORTRAN. The assembly language programmer is aware of memory locations used in the machine, the actual instructions of a machine, and possibly the general speed of instructions. He* is not necessarily aware of the underlying organization of the machine in terms of operational logical units or of the hardware technology used in the construction of the machine.

The word architecture is frequently used to mean the visible characteristics of only the element actually performing instructions, that is, the *processor* of a computer system. Since a programmer may view other elements of a computer system through the processor, architecture also more generally means the characteristics of all the component elements of a system that might concern a programmer working at machine level. These components include processor(s), *memory*, and *input/output subsystems*.

A processor is a unit that interprets instructions and changes data in conformity with the instructions of a program. A *computer* may have one or more processors either dedicated to the execution of specialized instructions or capable of performing all instructions. Memory is a device that holds instructions to be executed and data to be operated on. Input/output subsystems are collections of units that connect processors and memories with the devices (1) that interface with the outside world (printers, terminals, sensors, etc.) or (2) on

* Throughout this book "he" is used to mean "he" or "she." This avoids the cumbersome "he/she."

which large amounts of additional data or instructions are stored and from or to which data is moved to or from memory.

The word architecture is also used to suggest the relationships among various building block elements of hardware and software systems. A system may be thought of as having several *architectural levels*. Figure 1 shows a representation of architectural levels. Each rectangle represents a set of functions and the horizontal lines are the interfaces between the functions. The first part of this book discusses the architectural level represented by the line labeled *machine-architecture interface*. The second part of the book addresses the levels that support this interface.

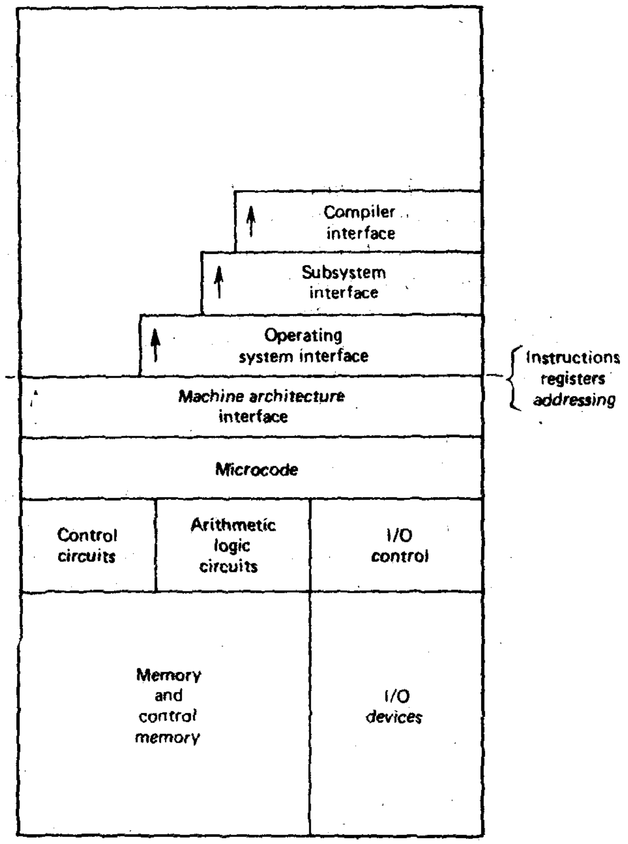

Figure 1. Architectural levels.

2. ELEMENTS OF AN ARCHITECTURE

From the point of view of a programmer looking at a computer system from an architectural level, a system has a set of characteristics that define how data will be represented and referenced, the operations that can be performed on data, and other features that determine how sequences of operations must be executed to achieve a computational result

2.1. Mode of Data Representation

This refers to the representation of values in the processor and memory and the manner in which strings of binary digits should be interpreted at different points in the architecture. The representation of information may be in either pure base 2 (binary) or encoded form generically called the *binary-coded decimal*. There is also a form of scientific notation, called *floating point*, where values and exponents may be represented separately in either pure binary or coded form.

When a value is to be manipulated by a processor, the circuits of the processor will be designed to treat the bits of that value as either pure binary, binary-coded decimal, or floating point. For example, a unit designed to add binary numbers will produce an erroneous result for the logic of a program if the data presented to it is in binary-coded decimal form. As another example, a program must know the mode of representation for various values printed on a system printer. Devices like printers always require data to be in some coded decimal form, which may not be the form in which the data is arithmetically manipulated in the processor. The program must issue instructions to put the data in proper form for printing.

2.2. *Size of the Basic Data Structure*

This refers to the organization of values in the memory of the system and in the processor. The basic architectural data structures are *bits, bytes, characters, and words*.

A computer system's memory is organized into *locations*, which may be visualized as elements in a one-dimensional array. Each element has a name, or an *address*. Thus we speak of a memory having 4096 locations; each location can be directly referenced according to its position on the list. For example, there are memory locations 0000, 0001, 0002, . . . , 4094, 4095. When one of these numbers appears in control circuits connected to memory either the contents (the value that exists in the location) will be brought from memory to the circuits of the processing unit or information in the processing unit will be stored in the memory, depending on the work associated with the address.

Each location in a memory has a fixed size that represents the number of *bits* (binary digits) held in that location. Thus we speak of machines with 8, 16, 24, and 32-bit memories. This describes the number of bits that will be transferred