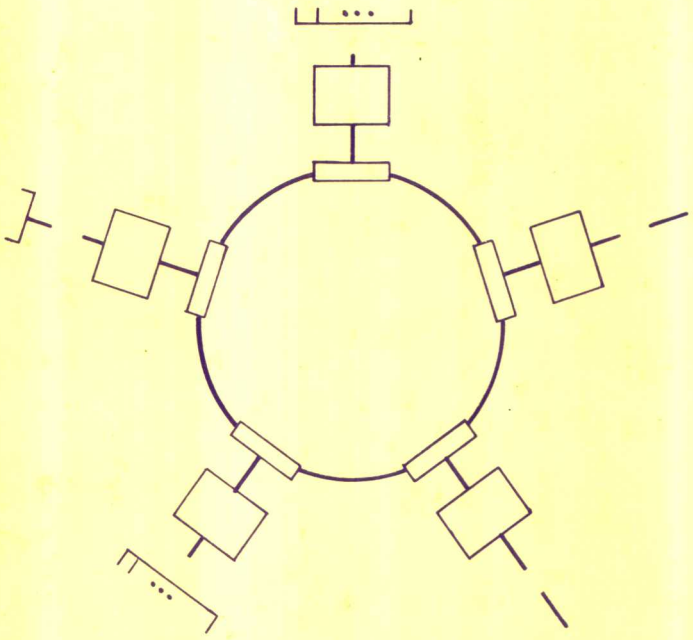
COMPUTER SYSTEMS ARCHITECTURE

Jean-Loup Baer



COMPUTER SCIENCE PRESS

COMPUTER SYSTEMS ARCHITECTURE

Jean-Loup Baer

COMPUTER SCIENCE PRESS

Copyright © 1980 Computer Science Press, Inc.

Printed in the United States of America.

All rights reserved. No part of this work may be reproduced, transmitted, or stored in any form or by any means, without the prior written consent of the Publisher.

Computer Science Press, Inc.
11 Taft Court
Rockville, Maryland 20850

5 6 85

Library of Congress Cataloging in Publication Data

Baer, Jean-Loup.
Computer systems architecture.

Includes bibliographical references and index.
1. Computer architecture. I. Title.
QA76.9.A73B33 621.3819'52 79-27039

US ISBN 0-914894-15-3
UK ISBN 0 273 01474 9

Preface

The term *Architecture* when applied to *Computer Systems* is relatively recent. Ten years ago, when Foster used it in the title of his book, some University librarians were thoroughly confused. In at least one case, the book was miscatalogued and sent to the School of Architecture specialized library. Even today, there is no entry specific to Computer Architecture in *Computing Reviews*, the Review Journal of the Association for Computing Machinery; papers and books on this topic are indexed under the Computer Systems category. I hope that this omission will soon be remedied since there is now common agreement on the existence of Computer Architecture although there is no consensus on what this field encompasses.

I was guided in my choice of including *Architecture* and *Systems* in the title of this book by the usual definitions of these terms. According to the American College Dictionary, architecture is the "art or science of building, including plan, design, construction and decorative treatment," and system is an "assemblage or combination of parts forming a complex or unitary whole." That modern computers form a system should not require any justification: even the von Neumann machine, the simplified model of a stored program computer, consists of an aggregate of five blocks. The architectural definition requires more elaboration. Because computers are tools to be used by people, I will add "engineering" to the above "art or science" but I will not try to draw the fine line between the Computer Science and Computer Engineering disciplines when it comes to the study of Computer Architecture. I prefer to indicate the features corresponding to "plan, design, construction and decorative treatment" when viewed in the context of sound computer architectures, namely:

- Structure: the static arrangement of the parts (plan);
- Organization: the dynamic interaction of these parts and their management (design);
- Implementation: the design of specific building blocks (construction);
- Performance Evaluation: the behavioral study of the system or of some of its components (decorative treatment!).

Naturally, computers are too young to have the historical richness of architectural masterpieces. But schools of thought exist in which stylistic issues

(stack architectures, orthogonal instruction sets, hierarchic and distributed systems etc.), when resolved, give an artistic seal to the final product.

The parallel between the two building processes could be continued. However, our last analogy is to note that while architects work with contractors, electricians, plumbers and the like, computer architects take their decisions after consulting software specialists and chip designers among many others. Nonetheless, even with this help, the range of expertise required from computer architects is very wide and the number of topics covered here is necessarily limited. However, it corresponds broadly to the guidelines set forth by the IEEE Task Force on Computer Architecture with a more systems oriented approach to the subject.

This book is divided into three parts. In the first two, I do not present specific architectures in detail but I draw many examples from current systems such as IBM System/370, CDC 6600, DEC PDP-10 and PDP-11 and also from more powerful or less common machines since it is my belief that advances in the more sophisticated models are echoed in subsequent standard products. In the third part, by contrast, I describe some complete systems.

Part I is an overview of what Bell and Newell have called the Computer Space. Chapter 1 surveys the historical dimension from ENIAC to current microcomputers. Chapter 2 introduces the PMS, ISP, CDL and Petri Net notations which will be used consistently in the remainder of the text. This provides the opportunity of evaluating instruction sets and addressing modes.

Part II is the study of the main (hardware) blocks, their control and their (software/hardware) interactions. Chapter 3 presents algorithms for the four basic arithmetic operations for the fixed and floating-point representations of numbers. The emphasis is on methods used to achieve "engineering" speed-up. Chapter 4 is devoted to central processors and to methods to render them more powerful such as look-ahead, functional unit distribution and automatic detection of parallelism. The concept of a stack computer is also introduced in this chapter. Chapters 5 and 6 deal with the memory hierarchy found in modern computer systems. Chapter 5 is technologically oriented: RAM's and their interleaving, rotational secondary memories and electronic disks are presented. Associative memories are briefly described. Chapter 6 is devoted to the virtual memory concept and to its implementations at the primary-secondary memory and cache-primary memory levels. The tradeoffs between various schemes are examined and evaluated by referring to simulations and actual measurements. Chapter 7 examines microprogramming and its applications to emulation, high-level language interpretation and tuning of architectures. The chapter ends with the presentation of an ideal microprogramming system. Chapter 8 is con-

试读结束：需要全本请在线购买： www.ertongbook.com

cerned with input/output: interrupts, channels, I/O devices and an evaluation of asynchronous input/output processing are discussed.

Part III presents specific systems. Because the numerous examples provided in the first two parts give a good representation of a medium-size architecture, I direct my attention mostly to small machines and to supercomputers. Chapter 9 examines the small end of the spectrum with the DEC PDP-11 family, including the VAX 11/780, being the vehicle for minicomputers and the Intel 8080 and Zilog Z8000 being examples of 8-bit and 16-bit microcomputers. The last section of the chapter presents the stack architecture of the Hewlett-Packard HP-3000. Chapter 10 deals with supercomputers. Pipeline processors (Amdahl 470 V/6, CRAY-1, CDC STAR-100, TI ASC), array processors (ILLIAC IV), and multiprocessors (C.mmp, C.m*, PLURIBUS) are discussed. Hardware and software considerations (synchronization, scheduling etc.) are included in the presentation. Finally, Chapter 11 mentions some specialized architectures, gives an overview of several options used to achieve distributed processing and attempts to assess the impact of VLSI on future systems.

This book grew out from notes given to first year Computer Science Graduate Students at the University of Washington. The material has been expanded and is appropriate for a two-quarter, or even a two-semester, sequence at the senior-first year graduate student level. It is suitable for Computer Science, Computer Engineering and Electrical Engineering majors. Although a course on Logical Design is not a formal prerequisite, it is my experience that students with a background in Digital Design can understand some parts of the text more quickly. I assume tacitly that the readers have had some exposure to high-level language programming and I use a Pascal-like notation to express algorithms.

Many colleagues and students have helped me during the preparation of the book. In particular, I would like to thank P. Borgwardt, D. P. Bovet, C. Crowley, C. Ellis, T. Hou, J. Jensen, K. Kim, E. Lazowska, B. Nussbaum, and M. Sievers for their technical and "grammatical" criticisms. I owe a great debt of gratitude to Professor G. Estrin who introduced me to the field of Computer Architecture. Some sections of this text are direct outgrowths of research supported by the National Science Foundation. I might not have written this book if it had not been for the text editing and formatting systems implemented by R. Tomlinson and G. Houston. I am indebted to them and to M. Reidel and H. Sardarov for their typing assistance. The patience and continuing support of my wife Diane Roseman deserves more than the usual amount of credit. Finally, I take full responsibility for any remaining gallicisms.

Jean-Loup Baer

CONTENTS

PART III Complete Systems: From Micros to Supercomputers

Chapter 1

HISTORICAL SURVEY OF COMPUTER SYSTEMS ARCHITECTURE

1.1 INTRODUCTION

Defining what is meant by Computer Systems Architecture is not a simple task. First, we must not restrict ourselves to the sole aspect of the hardware. Building black boxes of increasing size and complexity from some primitive entities such as adders, shift registers, or memory chips is certainly part of the process. Interconnecting these boxes through buses, switches, and controllers is also an important contribution to the overall design. But, the harmonious cooperation between the components in the system, the way messages are passed and received, and the blend of hardware and software features which make the system operate must be included. Thus, we shall consider Computer Systems Architecture as the design of the integrated system which provides a useful tool to the programmer. Since the term programmer itself is subject to various interpretations, as in systems programmer, applications programmer, coder, or programmer analyst, we shall study the architecture of computer systems at different levels. In particular, we shall treat:

- the internal workings of the black boxes which are the main components of the system;
- the means of interconnecting these boxes, their parallel activities, and cooperation.

This will imply not only the description of parts of the system and of their interconnections, but also, when necessary, of the supervisory control which might well be a software product.

Before proceeding to a short outline, two words of caution are in order. First, since this book is intended as continuing an introductory text on Digital Systems design, it is assumed that the reader has a basic knowledge of digital logic. For example, it is expected that the principles behind the construction of a full adder or a half-adder, the decoding of an instruction, etc., are well understood. We shall seldom be at that level of detail.

Second, this book will often be biased towards the study of large and powerful systems. Although this might not be the trend in the development of computers, we believe that it is in the study of large and powerful machines, even special-purpose ones, that advances in the state of the art are realized. In addition, we shall emphasize the performance aspects in evaluating components and total systems by quoting monitoring studies or by relying on the results of analytical models and simulations.

This philosophy of looking at large computers in order to improve more standard units is neither new nor limited to Computer Architecture. For example, the development of racing cars has had a significant impact on the safety of current automobiles. As early as 1952, shortly after the completion of the IAS machine which since then has become the model of the stored program computer, its inventor, von Neumann, sensed the need for a continued experimentation in building unique systems with "specifications simply calling for the most advanced machine which is possible in the present state of the art." Von Neumann and his associates were able to convince the Atomic Energy Commission, Sperry-Rand, UNIVAC and IBM of the validity of this approach. This resulted in the construction of giant computers such as LARC (UNIVAC) and STRETCH (IBM) which brought forth "major advances in Computer Technology that were rapidly reflected in the product line of these companies." The supercomputers of the seventies might play the same roles and it is perhaps no coincidence that a successful computer using advanced pipeline concepts was built by Texas Instruments, a manufacturer of integrated circuits and small calculators.

This book is divided into three parts. Part I is an overview of what Bell and Newell have called "the computer space." We shall look at the historical development of Computer Systems Architecture, and follow this by a study of the representation of computer systems at various levels. This study of methodologies and notations will allow us to give an overview of the development of systems from structural and behavioral viewpoints. Part II will be centered on the four main components of a von Newmann machine, i.e., the arithmetic-logical unit, memory and the management of a hierarchy of storage components, the control and its orderly implementation via microprogramming, and the connection of input-output devices with the remainder of the system using channels operating in parallel with the main

试读结束：需要全本请在线购买： www.ertongbook.com

processor. Part III will look at specific systems, such as mini and microcomputers, supercomputers of the array and pipeline families, multiprocessors, and at the impact of new technologies on future systems.

Several important issues are either not treated or barely mentioned. For example, associative processing receives little attention because of its very special-purpose nature and of its high manufacturing cost which makes it noncompetitive at the present time. Geographically distributed computer networks are simply mentioned. Another book would be necessary to treat them at a substantial level. Justice might not have been done to newly emerging architectures such as high-level language direct execution machines. Stack machines do not receive the treatment that their advocates would expect. This is a question of choice for which this author takes full responsibility.

1.2 HISTORICAL SURVEY

Until 1970, it was customary to review the history of electronic computers by providing a classification by generations. We shall adhere to this convention and follow the chronological outline of Rosen's authoritative survey (cf. bibliography at the end of the chapter). For the post 1970 era, we shall point out the trends in computer architecture which indicate a departure from the designs realized in the previous generations. We do not present an encyclopedic overview of the computer genealogical chart. Instead, we shall limit ourselves to what can be considered as the landmarks in the development of computer systems. The prehistoric age of computers, that is the period of mechanical and electromechanical machines, is not covered. The interested reader can consult a fascinating anthology edited by Randell as well as Goldstine's historical account from which the quotes from the previous section were taken. This latter book also provides an interesting view on the birth of the early computers as seen by an active participant.

1.2.1 Generation O (Prior to 1950)

The first electronic computers were not the product of commercial enterprises. They were unique realizations, built or contracted by Universities or Government supported research institutions.

ENIAC, completed in 1946 at the Moore School of Engineering (University of Pennsylvania) under the direction of Eckert and Mauchly, can be considered as the first electronic computer. It was made of more than 18000 vacuum tubes and 1500 relays, most of which had to operate simulta-

neously. ENIAC's primary function was to compute ballistic trajectories. It was able to perform nearly 5000 additions or subtractions per second. Arithmetic operations were done in decimal, serially, with each digit being implemented as a ring of 10 flip-flops. This ring lay-out was certainly inherited from previous electromechanical devices. There was no programming in the sense we give to this word today. The computer was wired in for specific computations and modifications or replacements of programs made new rewirings necessary. Hence, the set-up time was tremendous. Despite these shortcomings, ENIAC was to be used for ten years before being retired.

The idea of having the program stored in memory is generally attributed to von Neumann. While a consultant at the Moore School, he and the originators of the ENIAC designed the first stored program computer named EDVAC. At the same time, Wilkes, who had spent a summer in Philadelphia studying with the investigators of ENIAC, built the EDSAC at Cambridge University (U.K.). EDSAC was the first stored program computer to be operational (1949). Its storage was much more extensive than ENIAC's: it had a primary memory of 1024 words, a mercury delay line acting as a shift register, backed up by a drum of 4600 words. This has to be compared with the 20 registers and 312 words of externally alterable read-only memory of ENIAC. Besides being the first computer to execute stored programs, EDSAC can be considered as the first one to introduce the notion of a memory hierarchy.

The EDVAC project was severely crippled when Eckert and Mauchly left the Moore School to form their own company. Von Neumann, Goldstine and their collaborators at the Institute for Advanced Studies (IAS) in Princeton had already started on a new venture, referred now as the IAS or von Neumann machine. Because the von Neumann machine is still the usual frame of reference for many modern computers, we present its overall structure in some detail.

A von Neumann machine is a system composed of five basic units (cf. Figure 1.1) whose functions can be summarized as follows:

- The *input* transmits data and instructions from the outside world to the memory;
- The *memory* stores instructions and data, intermediate and final results;
- The *arithmetical-logical unit* (*ALU*) performs arithmetic and logical operations;
- The *control* interprets the instructions and causes them to be executed;
- The *output* transmits final results and messages to the outside world.

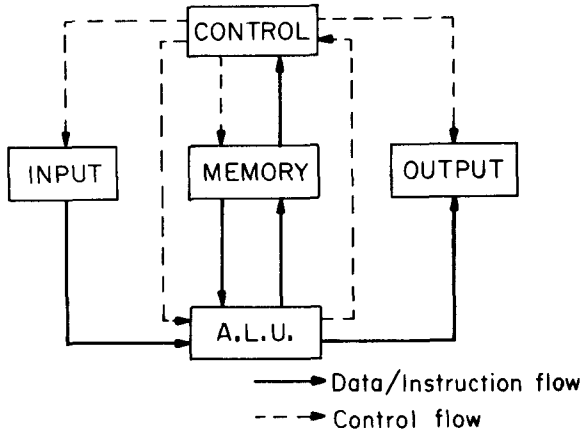


Figure 1.1 A von Neumann machine

For the execution of a non-branching instruction, the following sequence of actions, or *instruction execution cycle*, is undertaken:

1. The control requests and fetches from memory the next instruction to be executed;
2. The control decodes the instruction;
3. Depending on the results of Step 2:
   (a) An operand is fetched from memory, stored in a register of the arithmetic-logical unit and control is given to the latter in order to perform an operation; or
   (b) An operand is stored, from a register of the arithmetic-logical unit, in the memory; or
   (c) A request is made to the input (output) to accept (transmit) a word from (to) the outside world;
4. Upon termination of Step 3, return to Step 1.

In the original IAS machine, whose inner workings are described in the classic proposal by Burks et al., the memory consisted of 4096 (2^{12}) words of 40 bits. The medium used for storage was a cathode ray tube. Mention was made of an automatically managed memory hierarchy. The arithmetic-logical unit was of the single accumulator type and operations were performed in parallel binary mode (this is to be contrasted with ENIAC's serial arithmetic). The internal representation was 2's complement. Floating-point operations were not implemented; maybe the fact that von Neumann and

his collaborators were brilliant mathematicians prevented them from realizing the difficulties involved in having the programmers plan their own scaling factors. The control unit was hardwired, with a decoding mechanism allowing for an instruction set of 64 opcodes. The instructions were of the one address type, with the format:

opcode operand address

with the second operand being implicitly the ALU's accumulator. Since 6 bits had to be reserved for the opcode and 12 bits for the address, two instructions were packed in a single word. Words were also divided into tetrads (4 bits) and hexadecimal notation was advocated for debugging purposes.

Several other machines quite similar to the IAS computer were built subsequently, including ILLIAC at the University of Illinois, JOHNIAC at the RAND Corporation, MANIAC at Los Alamos, WEIZAC at the Weizman Institute in Israel, AVIDAC and ORDVAC at the Argonne National Laboratories.

ENIAC, EDSAC and the IAS-type computers were all geared to perform scientific tasks. The first machine whose goal was real-time computations (aircraft simulator, air-traffic and industrial process control) was Whirlwind I (WWI), started at MIT in 1947 and completed in 1951.

Like all its successors in this line of application, WWI had a short word length (16 bits). With room for 32 opcodes, only 27 instructions were actually implemented leaving a potential addressing space of 2048 words. Using 0.5 μs circuitry, WWI could perform 20,000 operations per second since the limiting factor was the memory cycle. In 1951, memory consisted of over 1200 registers of electrostatic storage tubes. This was soon to be replaced by 2K words of 8 μs cycle time of coincident-current magnetic core memory, a design developed by Forrester at MIT. The advent of core memory has certainly been one of the most important innovations in the history of Computer Systems.

Besides being the first computer with a core memory, the Whirlwind project made a significant contribution to computer science in educating a number of pioneers by making available detailed documentation of the logical design of the main parts of the machine.

Before core, the principal medium used for memory was an electrostatic storage tube, as in the early design of Whirlwind. The first practical version of the electrostatic storage device was due to Williams from Manchester University. But the main architectural advance in the Manchester computer was to be the introduction of index registers. Of importance too was the use of a magnetic drum for secondary memory. The drum was divided into

blocks of equal size, or pages. As we shall see later, the automatic management of a paged memory hierarchy was also "initiated" in Manchester a little more than a decade later.

Electrostatic storage was also used in the SWAC (Standard Western Automatic Computer) one of the fastest, and smallest, of the early computers. Its eastern counterpart, the SEAC, used memory delay lines, at least initially. SEAC might have been the first operational stored program computer in the U.S.A. (1950). Its successor, MIDAC, used a three address instruction format (for two operands and a result) with possible indexing on each address. The two index registers were the program counter (for looping) and a base register.

These early computers were operated on an open-shop basis. There was no operating system and only primitive input-output devices. Debugging was generally performed on-line, stepping up instruction by instruction, reading memory contents while the machine was running (this was made relatively easy by the CRT memories), and changing the contents of the registers through console switches. This was facilitated in part by the fact that programming was done in machine language (there existed no high-level language yet) and by the high competence of the mathematician-programmers.

1.2.2 Generation 1 (1950–1958)

The 1950's saw the birth of the computer industry. Rather than following a breadth first review of the genealogical tree, we consider the major companies of the time. Interestingly enough, the current giant, IBM, was not the first one to participate in the commercialization of electronic computers. Its superiority in sales management and public relation activities enabled it to overcome its relatively late start.

When Eckert and Mauchly left the University of Pennsylvania, they formed their own company and started to build BINAC, a small binary computer, intended for the Northrop Corporation. BINAC was not a stored program computer and never worked satisfactorily. Almost at the same time as BINAC was realized, Eckert and Mauchly started on the design of the UNIVAC (UNIVersal Automatic Computer) which was commissioned by the Bureau of the Census for its 1950 calculations. UNIVAC I was delivered in 1951.

One of the major architectural advances of UNIVAC I was its tape system. Magnetic tapes could be read forward and backward with buffering capabilities and error-checking procedures. UNIVAC I was a decimal machine with 12 digits/word. It had a single address instruction format and

two instructions were packed in each word. Its memory was still of the mercury delay line type, a factor which made it rapidly obsolete. Its successor, UNIVAC II, had core memory and was upward compatible with UNIVAC I. Late deliveries of this new model (1957) hampered the growth in the computer field of the Remington Rand Corporation which had acquired the Eckert and Mauchly original company.

UNIVAC I was intended, as its name implies, for both scientific and commercial applications. The first paper describing the system listed matrix algebraic computations, statistical problems (census related), premium billings for a life insurance company and logistical problems (this was before the invention of PERT networks) as a sample of the tasks that could be performed by the system. UNIVAC I was certainly the first successful commercial computer.

UNIVAC II was mostly intended for commercial applications. After acquiring Engineering Research Associates in 1950, Remington Rand built the scientifically oriented UNIVAC 1103 and 1103A, the latter delivered in 1956. They were 36 bit machines, using binary arithmetic and 1's complement representation, and a two-address instruction format of the form:

$$\text{Accumulator} = \text{Contents(address 1) op. Contents(address 2).}$$

The UNIVAC 1103 had floating-point hardware and was the first computer provided with a program interrupt capability.

At the time of their delivery the UNIVAC 1103 and the IBM 704 (see below) were the most advanced scientific computers of their generation.

The real entry of IBM in the electronic computer market began with the introduction of the Defense Calculator soon to be called the IBM 701. Prior to that date (1950), IBM's interest in the calculator area had been mostly restricted to office equipment and punch card machines. In the 1930's IBM had manufactured electromechanical devices to complement card punches; in the late 1940's some electronic equipment was added (IBM had an electronic multiplier at that time) to be known as the 600 series or Electronic Calculating Punch. In connection with the Northrop Corporation, a Card Program Calculator (CPC) was designed by coupling a calculating machine with a card punch. The CPC's were not stored program computers and were only semi-automatic in the sense that they needed human intervention to feed their (card) programs. However, they were extremely popular and in high-demand in computing centers, thus setting a base for further IBM deliveries.

IBM, as a company, was not overly interested in the development of early electronic computers. Still, in 1939, Aiken, who was designing an electro-

试读结束：需要全本请在线购买： www.ertongbook.com