Computer Organization and Architecture

Designing for Performance

Rishabh Anand
R. S. Salaria

KHANNA PUBLISHERS®
Investing in Learning®

Computer Organization and Architecture

Designing for Performance

Rishabh Anand
PostDoc (AI & ML), Ph.D. (Computer Science),
MBA (Finance), M.Tech. (ECE), B.E. (ECE)

R.S. Salaria
Educator, Author, Motivator, and Social Reformer
An alumnus of IIT, Delhi
Certified Software Quality Professional, Govt. of India
SUN Certified Programmer, SUN Microsystems, USA
SUN Certified Trainer, SUN Microsystems, USA.

KHANNA PUBLISHERS®
Investing in Learning®
Operational Office :
4575/15, Onkar House, Opp. Happy School,
Ground Floor, Daryaganj, New Delhi 110 002
Phones : 011-45033819 • *Mob*. 09811541460
email : contactus@khannapublishers.in

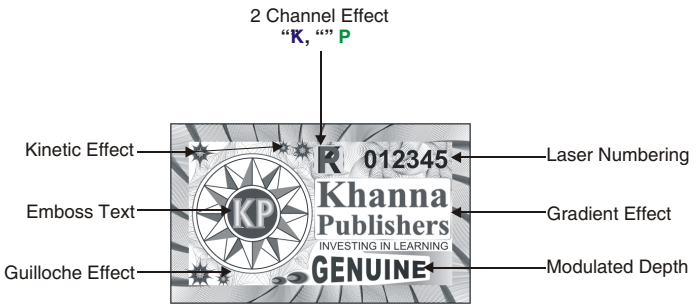
Published by :
Romesh Chander Khanna & Vineet Khanna
for KHANNA PUBLISHERS
2-B, Nath Market, Nai Sarak,
Delhi-110006 (India)

Visit us at : www.khannapublishers.in

Hologram & Description

To all readers of our books, to prevent yourself from being defrauded by piraters, please make sure that there is an Hologram on the cover of our books with the below specifications. If you find any book without Hologram and Description, please mail us at contactus@khannapublishers.in

Thanking you



ISBN No. : 978-93-92549-38-0

First Edition : 2023

Preface

The most beautiful thing we can experience is the mysterious. It is the source of all true art and science.

—Albert Einstein, What I Believe, 1930

We believe that learning in computer science and engineering should reflect the current state of the field, as well as introduce the principles that are shaping computing. We also feel that readers in every specialty of computing need to appreciate the organizational paradigms that determine the capabilities, performance, energy, and, ultimately, the success of computer systems.

Modern computer technology requires professionals of every computing specialty to understand both hardware and software. The interaction between hardware and software at a variety of levels also offers a framework for understanding the fundamentals of computing. Whether your primary interest is hardware or software, computer science or electrical engineering, the central ideas in computer organization and design are the same. Thus, our emphasis in this book is to show the relationship between hardware and software and to focus on the concepts that are the basis for current computers.

The recent switch from uniprocessor to multicore microprocessors confirmed the soundness of this perspective. While programmers could ignore the advice and rely on computer architects, compiler writers, and silicon engineers to make their programs run faster or be more energy-efficient without change, that era is over. For programs to run faster, they must become parallel. While the goal of many researchers is to make it possible for programmers to be unaware of the underlying parallel nature of the hardware, they are programming, it will take many years to realize this vision. Our view is that for at least the next- decade, most programmers are going to have to understand the hardware/software interface if they want programs to run efficiently on parallel computers.

This self-contained text devotes one full chapter to the basics of digital logic. While the initial chapters describe in detail about computer organization, including CPU design, ALU design, memory design and I/O organization, the text also deals with Assembly Language Programming for Pentium using NASM assembler. What distinguishes the text is the special attention it pays to Cache and Virtual Memory organization, as well as to RISC architecture and the intricacies of pipelining. All these discussions are climaxed by an illuminating discussion on parallel computers which shows how processors are interconnected to create a variety of parallel computers.

This task is challenging for several reasons. First, there is a tremendous variety of products that can rightly claim the name of computer, from single-chip microprocessors costing a few dollars to supercomputers costing tens of millions of dollars. Variety is exhibited not only in cost but also in size, performance, and application. Second, the rapid pace of change that has always characterized computer technology continues with no letup. These changes cover all aspects of computer technology, from the underlying integrated circuit technology used to construct computer components to the increasing use of parallel organization concepts in combining those components.

Despite the variety and pace of change in the computer field, certain fundamental concepts apply consistently throughout. The application of these concepts depends on the current

state of the technology and the price/performance objectives of the designer. The intent of this book is to provide a thorough discussion of the fundamentals of computer organization and architecture and to relate these to contemporary design issues.

The subtitle suggests the theme and the approach taken in this book. It has always been important to design computer systems to achieve high performance, but never has this requirement been stronger or more difficult to satisfy than today. All the basic performance characteristics of computer systems, including processor speed, memory speed, memory capacity, and interconnection data rates, are increasing rapidly. Moreover, they are increasing at different rates. This makes it difficult to design a balanced system that maximizes the performance and utilization of all elements. Thus, computer design increasingly becomes a game of changing the structure or function in one area to compensate for a performance mismatch in another area. We will see this game played out in numerous design decisions throughout the book.

A computer system, like any system, consists of an interrelated set of components. The system is best characterized in terms of structure— the way in which components are interconnected, and function—the operation of the individual components. Furthermore, a computer's organization is hierarchical. Each major component can be further described by decomposing it into its major subcomponents and describing their structure and function.

The objective is to present the material in a fashion that keeps new material in a clear context. This should minimize the chance that the reader will get lost and should provide better motivation than a bottom-up approach. This text is intended to acquaint the reader with the design principles and implementation issues of contemporary operating systems. Accordingly, a purely conceptual or theoretical treatment would be inadequate.

Throughout the discussion, aspects of the system are viewed from the points of view of both architecture (those attributes of a system visible to a machine language programmer) and organization (the operational units and their interconnections that realize the architecture).

The audience for this book includes those with little experience in assembly language or logic design who need to understand basic computer organization as well as readers with backgrounds in assembly language and/or logic design who want to learn how to design a computer or understand how a system works and why it performs as it does.

Hope you enjoy reading this first edition as much as I did in writing it!

—Authors

Contents

1

COMPUTER SYSTEMS—A PERSPECTIVE

LEARNING OBJECTIVES

In this chapter we will learn:

- How to view a computer system from the perspectives of end users, system and application programmers, and hardware designers.
- How to view a computer system as a layered system with facilities provided by lower layers being used by higher layers.
- Block diagram view of a computer system.
- The views of digital logic designers and those of the architects of computer systems.

1.1 INTRODUCTION

Computers are used in every walk of life to assist us in the various tasks we perform. The wide availability of the Internet has enhanced the use of computers for information sharing and communication. Today, low-cost personal computers are available in plenty at homes and workplaces. Powerful supercomputers can be accessed from remote terminals through communication networks. Computers are more widely used now than ever before, not only by specialists but also by casual users for a variety of applications. Because of the widespread use of computers, software developers nowadays consider the ease of use as the most important

design criterion. Further, the complexity of application programs has increased enormously. Thus, many application programs consist of tens of thousands or even millions of lines of code. Such large software is developed by a team of developers to operate reliably and maintained and improved over long periods to be available with minimal breakdowns. From the end users' perspective, the number of lines of code is irrelevant. Users simply interact with the application program viewing it as a 'black box' and use it through an appropriate graphical user interface. Application designers have to cater to a wide range of requirements. Hence the system software and hardware of a computer system should have facilities to support such application software developments.

Today computers have become universal machines used by almost everyone. Home computers have proliferated and are used primarily for searching the World Wide Web for information on diverse matters such as health, investment and news. E-mail is another popular application among home users. Besides this, they are used for e-shopping, booking tickets by airlines, trains, etc. Another use is to draft letters, keep diaries, look for jobs, marriage partners, etc. These users are not professionals. They are lay users and their main concern is ease of learning how to use diverse applications and use them in a simple convenient manner. Apart from individual users, organizations such as banks, hospitals, hotels, governments and industrial manufacturers use computer systems interactively to assist them in their functions. Banks use them to look up user accounts, print account statements, etc. Hospitals use them in patients' admission process, scheduling operation theatres, routine patient care, billing and so on. Hotels use them to check room reservation status, allot rooms, bill customers, etc. There are also myriad applications in the "back offices" of the organizations for payroll, personnel management, and so on.

The hardware and the system software of a computer system are complex and they can be abstractly modelled to get a simplified view. The kind of details that we abstract from the model or conversely that we present in the model, depends on the purpose for which or for whom the model is created. Generally, a system designer uses the model for understanding and analyzing system before embarking on the costly processes of its design and development.

To solve problems using computers, we need two major entities: **data**—which suitably represent the relevant details of the problems to be solved in a form that can be manipulated or processed by computers; and **algorithms**—which systematically transform the input data using a step-by-step method and give the desired output. We need a language to describe such algorithms; and we need structured representation of data called 'data structures' to make them amenable for computer processing. They are structured collections consisting of primitive data elements such as integers, reals, Booleans, and/or ASCII characters. Representation and processing of the primitive data elements are supported by the computer hardware. A language used for describing algorithms meant for execution on computers is known as a **Programming Language**. It is a formal language

which is very different from a natural language such as English. Every statement of a formal language is interpreted by the computer hardware in exactly one way. Thus such a language has precisely defined syntax and unambiguous semantics.

Programming languages can be categorized into: (i) higher level languages like JAVA or C++ which are independent of the hardware, and (ii) assembly languages that are specialized and dependent on the hardware of the computer. An algorithm expressed using one of the many programming languages is called a **program**. A program consists of several statements each of which performs an *operation* on some *data*. The operation is very primitive at the hardware level: add, subtract, compare, move data from one register to another, test if a particular bit is ON or OFF, etc. On the other hand, at the application level it can be quite complex. For example, when a computer system is used for reserving an airline ticket, it could look like this: "Reserve a one-way ticket from Bangalore to Montreal by Air France for next week Sunday." It is the responsibility of the application designer to develop a clear specification for all the relevant higher level operations pertinent to the domain of application.

Often a computer system is viewed as consisting of several layers depicted in Figure 1.1. Usually the higher layers depend upon the facilities provided by the lower layers. Such a layered view provides a functional overview of a complex system such as computer. Each layer will be designed with careful considerations to the needs of the upper layers supported by it as well as objectives fulfilled by that layer and the constraints of the technology used for its implementation. In this introductory textbook, we will study about the hardware layer and the assembly language layer in detail.

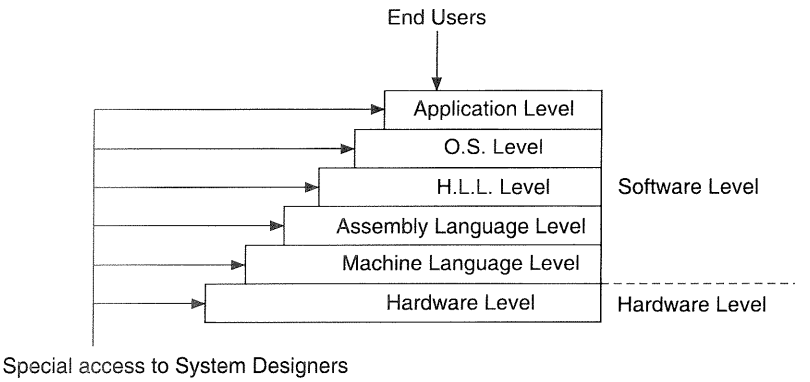


FIGURE 1.1 A layered view of a computer system.

1.2 A Programmer's View of a Computer System

Consider the following program segment written in a higher level language to sum 64 numbers stored in an array and find their average.

```
Total  := 0
For i=1 to 64
        Total := Total + Marks (i)
End
Average = Total / 64
```

Later in Chapter 8 we will study the details of NASM, the assembly language of the popular Pentium processor. The above program segment can be written in assembly language as below:

```
        mov     ecx,64          ; initialize ecx register to 64; loop count
        mov     eax,0           ; initialize the eax register to zero
        mov     esi,0           ; initialize index 'i' to 0
        mov     ebx, marks      ; store the base address of the array in ebx
;
addnext add     eax, [ebx+esi]  ; add the i-th element to eax
        inc     ebx,4           ; increment by 4 to get the next element
        loop    addnext         ; this instruction uses the ecx register
        shr     eax,6           ; a faster way to divide by 64 (2^6)
```

Let us suppose that a professor has 64 students in his class. He, as the end user of a computer software, issues a command: "Find the average mid-term marks of my class." If the software has the knowledge that this professor has 64 students in his class and their mid-term marks are stored in the vector *Marks*, the above program segment can perform this task. A programmer using the higher level language would focus more on the algorithm and the data structures needed to solve the problem. The assembly language level programmer, on the other hand, would have to focus not only on the algorithm and the data structures, but also on finer details such as which hardware registers of the processor to use, where to store the program and data in memory, and what instructions would efficiently implement the operations needed on the data, and so on. A **compiler**, which is a software system, can perform the task of translating a given higher level language program into its assembly language equivalent automatically. People who develop such compilers are also called programmers. Their needs are relatively more specialized than the needs of the application programmers.

For a given computer system, there are several compilers to facilitate the users to develop application programs in the languages that are suitable for them. Compilers are thus one important software resource of a computer system. If we add up the storage needed by all compilers, it is quite large. As the demand for storage is very high in a modern computer system, there are at least two kinds of memories: Primary memory or **Random Access Memory** (RAM) and Secondary memory, for example disks. Disks are much cheaper than RAMs and hence it is cost effective to have billions of bytes (giga bytes or GB) of disk storage in a computer. When large disk storage is available, several users of a computer system can keep their large data files in it. Thus, disk space is yet another resource that is shared by users. It should, therefore, be properly managed, that is, allocated and

de-allocated to users. Apart from memory, another resource to be managed in a modern computer system is the "processor's time". A typical processor of today can execute billions of instructions per second. Such a high capacity can be shared among multiple tasks. Coordination of multiple tasks together solves a more complex problem. Manager of all such resources in a computer system is also a software that is known as the **operating system.** There are programmers of different kinds—operating system developers, compiler writers, application developers, and so on. Their needs and focus vary.

1.3 Hardware Designer's View of a Computer System

A block diagram model of the computer system as viewed by a hardware designer is shown in Figure 1.2. This classical model of a computer hardware system consists of five basic subsystems:

1. Input
2. Output
3. Memory
4. Processor
5. Control

These subsystems communicate with each other by transferring *data* and *control* signals among themselves which are denoted by the lines connecting the blocks in Figure 1.2. In this figure the solid lines denote the data flow and the dotted lines denote the control flow.

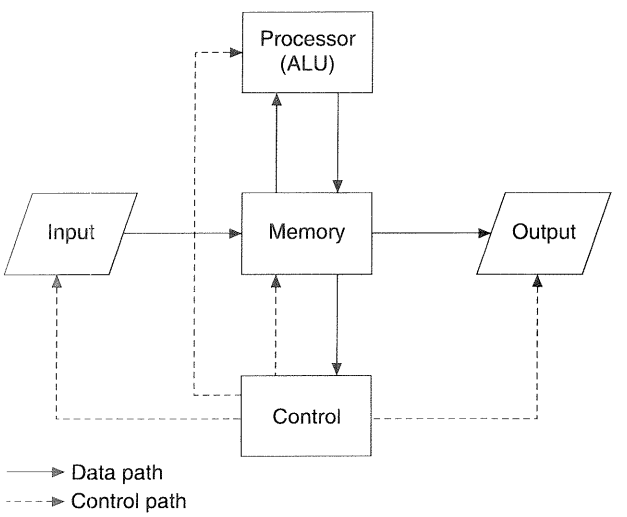


FIGURE 1.2 Block diagram of a computer.

As a computer system is used to solve a problem it is necessary to feed data relevant to the problem. This is done via the input subsystem. The results obtained by solving the problem must be sent to the person (or even a machine) who gave the problem. This is done by an output subsystem. The memory subsystem is needed to store data as well as the program. The program is a step-by-step description of the algorithm which describes the process of transforming the input to the desired output. It is normally expressed in a language that the hardware can understand. The processor is required to transform the input data based on the *primitive operations,* which the hardware is designed to perform. Let us call these primitive operations as *machine instructions.* There is a one-to-one correspondence between the machine instructions and the assembly language instructions. For ultimate execution, a program will be composed of such machine instructions. The control subsystem is needed to correctly sequence the execution of various machine instructions. It should be noted that the steps of an algorithm, written in the form of a 'program' are also stored in the memory and fetched from memory one machine instruction at a time for execution. The control unit knows how to control and coordinate the various subsystems in executing a machine instruction. To denote the fact that the machine instructions coming from the program stored in memory is used by the control unit, we have drawn a solid line between memory and control blocks in Figure 1.2. The processing subsystem performs both arithmetic (ADD, SUBTRACT, MULTIPLY, DIVIDE) and logic (AND, OR, NOT) operations and thus it is also known as Arithmetic and Logic Unit or ALU. Sometimes we refer to the combination of processing and control subsystems as CPU.

We note in Figure 1.2 that memory is the 'heart' of this hardware organization. A program is stored in memory and the data on which it operates is also stored in memory. Instructions from this program are fetched by the control subsystem one after another in an orderly fashion and the operations corresponding to that instruction are performed by the ALU. We can redraw Figure 1.2 as shown in Figure 1.3. In this new organization, we have explicitly shown the communication path, in the form a hardware bus, between the various blocks. A **bus** in the hardware is a 'bunch' of parallel wires and the associated hardware circuits which transport the binary signal from source to destination. Two *buses* are shown in this figure and both of them are bi-directional. The I/O bus connects the input/output subsystems to memory; and the processor bus connects the CPU to memory. CPU is made of hundreds of millions of electronic circuits that are suitably packaged into components. These components are used to build subsystems and they are controlled by a clock that runs at mega cycles per second (abbreviated MHz) or giga cycles per second (GHz). On the other hand, the I/O is made of electro-mechanical and electro-optical components and thus they operate only at several kilo cycles per second (KHz). Figure 1.3 brings out an important problem in computer design. We see that there are two entities which communicate with memory. Their speeds are very different (ratio of 1 to 1000). These multiple entities may require the services of memory (READ or WRITE services) simultaneously. How should we deal with this speed mismatch? How do we resolve the conflict which may arise if both the

subsystems attempt to simultaneously access the same memory location? The computer hardware designers and the system software designers together should address these two questions. We will see later in this book how these problems are satisfactorily solved.

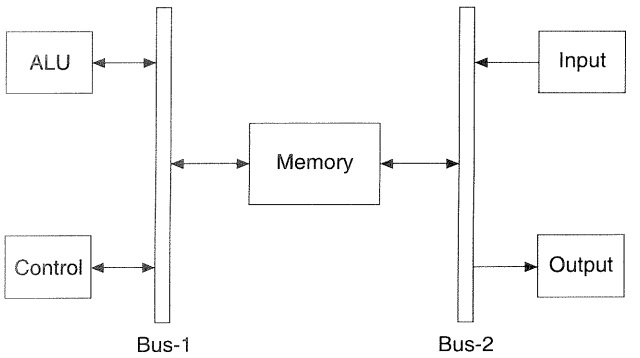


FIGURE 1.3 Bus-based organization of a computer.

A question arises, "How is this hardware used by the software?" The organized collection of machine instructions is what we call a 'program'. The program is written in an appropriate programming language that the machine can interpret and execute. The program embodies the method of solving a problem or the process that transforms the input data into the desired output result. Normally, the program, developed by a software engineer, may have thousands or millions of instructions put together and it is stored in the memory. The program will then be executed under the control of the control subsystem instruction by instruction. Thus, we can distinguish between the two clear stages. The first stage is the one in which the program is developed by a programmer or a team of software engineers using some software tools and it is stored in the memory with the help of the input subsystem. The second stage is when the program is executed and the end user interacts with the computer and performs his or her intended tasks. This two-stage approach forms the basis for the "stored program concept" which we will explain in greater detail in the subsequent chapters.

1.4 Objectives of the Computer Architect

In this section, we briefly explain the three terms: digital logic, computer organization and architecture. A computer system, either at the hardware level or at the software level, can be viewed as an abstraction consisting of several layers. **Digital logic** is primarily concerned with the hardware layer. **Computer organization's** primary concern encompasses both hardware level and computer language levels. Referring to Section 1.2, we note the design of processing unit is closely linked with the machine language instructions one would like to support. The machine language

instructions to be supported depend primarily on the type of application programs one would like to run on the machine. The complete set of instructions supported by a computer in its hardware is called the instruction set of that machine. The choice of instruction set is one of the main concerns of computer organization. This choice is governed by many factors such as the hardware-software trade-off and the compatibility of the instruction sets of the computer as it evolves from one generation to the other. The trade-off implies that a large number of instructions supported by complex hardware would simplify writing machine language programs but it would also increase the complexity of the hardware and hence the cost. Computer organization is also concerned with the organization of memory and its interconnection with CPU and I/O subsystems. One of the major problems in designing memory is the speed mismatch between memory and CPU. Over the past decades devices used to design CPU have always been 10 times faster than those used in memory. This is primarily due to economics; in the sense that large memories are needed to support most applications and thus slower cheaper devices are used to design them compared to CPU. Thus in computer organization one uses the idea of designing a hierarchy of memories to reduce the speed mismatch. This is one of the major topics we will discuss. Lastly the organization of I/O subsystems and their interconnection with memory and CPU is a major concern. Here the problems are the diversity of I/O devices which range in speeds of 1 to 100 and their mode of operation also vary widely. While some devices transfer data one byte at a time, others send streams of byte in chunks of the order of 200. We will describe how to tackle this problem in this book.

A **computer architect** is in overall control of the design of a computer system. We may compare the job of a computer architect with that of a building architect. A building architect's primary concern is planning a building, taking into account the needs of prospective occupants of the building. This is coupled with aesthetics of look and feel of the building. While these are the major objectives they have to be met most economically using the latest materials available for construction. The architect must be knowledgeable about the design of structures and be able to interact with structural engineers who will design the structure.

A computer architect's job is somewhat similar. The primary objective of a computer architect is to meet the requirements of the end users of the system making optimal use of hardware components available at reasonable cost. We have described in some detail the end users' perspective. Specifically we have seen that computers are now widely used by diverse users. A computer architect has to be aware of both the hardware and software layers and their interactions. Going back to our analogy, a person dealing with computer organization is similar to a structural engineer. The computer architect must thus be conversant with computer organization. He has to pay special attention to deal with the speed mismatch between main memory and CPU as well as that between I/O systems and memory/ CPU. The architect has to understand the prevailing technology and changing user requirements and accordingly tune various systems and their parameters to interoperate and create a balanced system. Specifically an architect uses facilities

provided by the operating system to solve the following problems related to memory systems.

- Increase the effective speed of memory by combining a large low cost memory and several fast smaller memories to provide an effective faster access to those data immediately required by the CPU.
- Increase the effective capacity of memory by combining a semi conductor Random Access Memory (RAM) (whose capacity is small but whose speed is high) with a disk memory (whose capacity is 1000 times more than that of RAM but is 1000 times slower) to provide a "Virtual memory" with capacity of a disk and a speed approaching that of the RAM.

By combining the above two techniques an architect succeeds in providing a large capacity fast memory required by today's large applications. In doing this, the services of the operating system as well as the underlying hardware are made to cooperate by the computer system architect.

Another area of importance to computer architects is I/O systems. As we pointed out I/O devices are diverse and slow. They could become a bottleneck to computation. Methods of alleviating this speed mismatch by both hardware and software are devised by architects.

In modern times, two other important developments have taken place. One of them is the Internet where computers across the world are interconnected by a communication network. This interconnection has led to several important applications such as the World Wide Web, searching for information on the web, e-mail and e-commerce. This has been enabled by computer network architects who evolved several standards, both at the hardware and software levels, which make the interoperability of diverse computers possible. With a wide diversity and access, newer problems arise. Privacy and security become very important issues.

The second important development is the design of parallel computers. Both for increasing the computational power and the availability under certain failures, parallel use of multiple computers have become common. Parallel computers operating cooperatively achieve high speeds needed to solve highly complex problems. With developments in the fabrication of electronic components like the CPU, today people are able to fabricate multiple processors in a single component. A challenge to an architect is to effectively use this power provided by the hardware to speed up problem solution. A further challenge arises in making several independent processors share a large memory system and access it in an orderly manner. There are several such problems which arise and pose challenging issues to today's computer system architects.

1.5 Some Invariant Principles in Computer Design

Computer systems are designed cooperatively by hardware designers, software designers and computer systems architects. A computer architect needs to

understand the current requirements of the population of users, the need for effective use of the final system and the prevailing technology. These change quite rapidly. In the last five decades, there have been huge shifts in user requirements and prevailing technology of both hardware and software. In spite of these rapid changes there are some major principles which have remained invariant over a longer period. They are:

Upward Compatibility This is essential as software development is human intensive and thus expensive to develop and maintain. Thus as new computer hardware is introduced, it is essential to make sure that software developed for earlier machines is able to run with little change in the new generation machines.

Software-Hardware trade-off It is possible in computers to obtain certain functionalities either by hardware or by software. Hardware is faster but more expensive whereas software is cheaper but slow. The trade-off between speed and cost changes as technology evolves. There is, however, always a possibility of hardware-software trade-off. This is what distinguishes computer system design from other areas of engineering systems design.

Locality of reference of instruction and data A computer stores a sequence of instructions in the memory in the order in which they are executed. Sequential thinking seems to be natural for humans and sequential programming has always been a dominant programming style. A number of data structures such as vectors, sequential files and strings are also linear and sequential. In other words, if one data item in the structure is referred, it is most likely that its neighbors will be accessed next. In over five decades of evolution of computer systems, the locality of reference has remained an important invariant in design. The design of pipelined processors, cache memory and virtual memory are all dependent on this principle. Pipelined processors use temporal parallelism, which is possible if different phase in the execution of an instruction can be overlapped when a sequence of instructions are carried out. We will describe this in great detail in this book.

Parallelism in applications There is inherent parallelism in many programs. For example, the operation of two multiplications and a division in the expression (a*b-c/d+f*z) can all be carried out simultaneously if there are two multipliers and a divider which can work independently. Such parallelism is used in pipelined processors. Data parallelism occurs when the same instruction can be carried out on several data items such as simultaneous processing of pixels in a picture or a video frame. Besides this, it is possible to overlap operations of CPU.

Amdahl's Law This law states the simple fact that when a system consists of many subsystems, in order to increase the speed of the system as a whole all subsystems must also be speeded up by the same amount. Thus in order to decrease the overall time for solving an application, it is essential to increase uniformly the speed of all parts of computing systems. Thus computer designers have to pay particular attention to increase the speed of the slowest subsystem such as I/O by appropriate architecture.

The Pareto principle or 20-80 law This law states that in a large number of systems, 20% of the system consumes 80% of the resources. It is thus important for a computer architect to identify this critical 20% of the system and spend maximum efforts to optimize its operation.

During early days of computers, experimental data on working computers were not easily available. Thus the design used mostly common sense and several hunches which experienced engineers had. This situation has now changed. A lot of experimental data on several computers using a variety of applications programs have been gathered. Thus the actual performance of computers is available allowing an architect to understand the critical parts of a system. The modern methodology of design consists of collecting a large number of typical application programs known as benchmark programs using which a proposed design will be assessed. Before committing hardware resources and building a system, it is simulated in an existing computer and various alternatives are tried to arrive at an optimal design.

SUMMARY

1. A computer system may be viewed from the perspective of end users, system and application programmers and hardware designers.
2. End users' primary concern is ease of use and good user interfaces which may be graphical, audio and even video.
3. The main concern of system and application programmers is the facilities provided by the hardware which eases their task and enables them to develop error-free maintainable programs.
4. Hardware designer's concern is to provide the best design with currently available technology at reasonable cost.
5. A complex system such as a computer is usually viewed as consisting of several layers. Each layer may be independently designed. Higher layers depend on the services provided by the lower layers.
6. We can view a computer system as consisting of six layers. The lowest to highest layers are respectively hardware, machine language, assembly language, high level language, operating system and application programs.
7. A computer hardware system consists of five subsystems. They are Input, Output, Memory, Central Processing Unit and Control Unit. In the rest of this book, we will describe each one of them in detail.
8. At the lowest level in the hierarchy of languages is machine language which consists of instructions provided by the hardware of the computer.
9. Assembly language uses mnemonics for hardware instructions and symbolic representation of memory address. It is a one-to-one representation of machine language.

Computer Organization and Architecture

Designing for Performance

About the Book

With clear, concise, and easy-to-read material, the first edition of Computer Organization and Architecture is a user-friendly source for students studying Computer Science, Computer Applications, Electronics Engineering, and Information Technology. Using brand new material and strengthened pedagogy, this text ensures that students are effectively engaged in the world of computer organization and architecture.

Computer Organization and Architecture has a good mix of hardware- and software-oriented topics. A comprehensive and up-to-date view of the architecture and internal organization of computers has been presented in this book. The processor design has been dealt with in an in-depth manner. This book presents state-of-the-art with a unique balance among the theoretical principles, design approaches and practical implementation of the computer organization and architecture.

Computer Organization and Architecture stresses the structure of the complete system (CPU, memory, buses, and peripherals) and reinforces that core content with an emphasis on divergent examples. It interrelates three different viewpoints to provide unique understanding of the subject: the Perspectives of the Logic Designer, the Assembly Language Programmer, and the Computer Architect.

About the Authors

Rishabh Anand is an eminent academician; plays versatile roles and responsibilities juggling between industry, research, publications, and consultancy. He has completed his PostDoc (AI & ML) from Sao Paulo State University, Brazil, PhD (Computer Science) from University of Bristol, United Kingdom, Degree of Master of Business Administration as MBA Management from International MBA Institute, Switzerland and Program Diploma in Innovation Management from International Business Management Institute, Germany. Also, he is the Reviewer/Editor for IJTESSS, IJISP, IJCAC, IJECME, IJICTE, JITR, IJMPA, IJTHI, IRJET, IJCRT and IJSDR. He is a prolific author with 38 Text and Reference books to his credit. He is also associated with various professional bodies like IEEE, LMCEGR, IAENG, Internet Society, IAOP, and IAOIP. He is currently working in IT and ITES industry with overall 16 years of experience. He is CDPT™, PMP®, PRINCE2®, DevOps-PM™, ITIL4®, CSM® & Kanban-ASC™ Certified Professional

.

Prof. R. S. Salaria is an outstanding teacher, a prolific author, a great motivator, and a social reformer. He is alumni of IIT, Delhi. He is a Certified Software Quality Professional (CSQP) by Govt. of India; SUN Certified Programmer and Trainer by SUN Microsystems, USA. He is life member of Computer Society of India (CSI), Institution of Electronics and Telecommunication Engineers (IETE), Indian Society for Technical Education (ISTE), and Punjab Academy of Sciences (PAS).

He has delivered enlightening sessions in numerous Faculty Development Programmes (FDPs)/Faculty Empowerment Workshops (FEWs) to empower the faculty to become passionate teachers. He has also delivered enlightening sessions for students pan India to empower them to become passionate learners, great professionals, and good human beings. He is also taking initiatives to empower people of various sections and age groups on various issues such as character building, women empowerment, female foeticide, honour killings, and various dimensions of corruption in public and private life.

KHANNA PUBLISHERS®
ISO 9001:2015
4575/15, Onkar House, Opp. Happy School,
Ground Floor, Daryaganj, New Delhi-110002
Phones: 011-45033819, 9811541460
E-mail: contactus@khannapublishers.in

Website:
www.khannapublishers.in

ISBN 939254938-5
9 789392 549380