

COMPUTER SYSTEMS

Concepts and Design

GLENN A. GIBSON
*Electrical Engineering Department
The University of Texas at El Paso*



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Gibson, Glenn A.
Computer systems : concepts and design / Glenn A. Gibson.
p. cm.
Includes bibliographical references.
ISBN 0-13-172958-6
1. System design. 2. Computers—Design and construction.
3. Computer software—Development. I. Title.
QA76.9.S88G43 1991
004.2'1—dc20

89-77266
CIP

Editorial/production supervision: *bookworks*
Cover design: *Bruce Kenselaar*
Manufacturing buyer: *Lori Bulwin*



© 1991 by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America
10 9 8 7 6 5 4 3 2

ISBN 0-13-172958-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

fewer instructions and are more efficient than others. Sometimes a slight change in the ordering of instructions or the inclusion of hardware that aids the execution of certain instructions will make a computer system enormously more efficient or better suited to a particular application. The overall design of a computer system is multifaceted and the optimal design should take into account not only the hardware and software, but also the interplay between the hardware and software.

1-1-1 Hardware

At the abstract level, computers are constructed from the following fundamental components and the means for connecting these components together:

Source, or **input**, devices that create or input information and then transfer the information to the other components of the computer. The origin of the information is not important; the information may be generated by the device or input from the external world.

Sink, or **output**, devices that receive information from the computer and then output it to the external world.

Storage devices that receive information and then, at a later time (when they are given the proper control signals), provide the same information to other devices in the computer.

Conversion devices that receive information, convert it, and then provide the converted information to other devices in the computer.

The information flowing among these fundamental components may be either instructions or data. In addition to information signals, control signals are communicated among the components and at least some of the conversion devices convert instructions into control signals.

The above devices can be constructed and then connected together in an infinite number of ways. It is the job of the hardware designer to decipher the description of what the computer is to do and then design the individual components and put them into a suitable configuration.

The procedure for designing any complex system, computer or otherwise, begins with a definition of its major components and a description of how these components are to be connected. Then the major components are broken down by defining their principal subcomponents, which are in turn divided into subcomponents. The subdivision process continues until the structure of the system has been completely delineated. This design approach is referred to as **top-down design** and is necessary because of the limited amount of information that the human mind can grasp at one time. It provides an orderly means of arriving at a solution to a complex problem.

Figure 1-3 shows the hierarchy associated with the hardware design of a computer system. Note that the first task is to decide on the major subsystems that are needed and how they should be connected. An example of a subsystem is a set of disk drives and its associated control circuitry. Next, the electronic

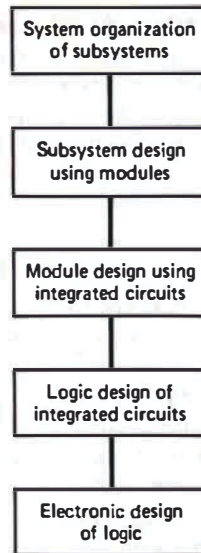


Figure 1-3 Hardware design hierarchy.

portion of each subsystem could be divided into modules. Typically, a module is the electronics on a single printed circuit (PC) board and this level of design includes determining the modules that are required and how they should be connected together. Designing a PC board is a matter of selecting the integrated circuits (ICs) and individual electronic components (transistors, resistors, capacitors, etc.) and laying out the board's conducting surfaces. Each IC is made up of logic elements and, at the lowest design level, each of these elements are constructed from elementary electronic components. The lowest design level would also be concerned with selecting the technology for fabricating the electronics.

Although this book discusses the design considerations of the system and subsystem levels, its emphasis is on the module and logic levels. Except for pertinent references that are useful to the logic level discussions, this book does not consider the electronic level, but leaves this area to books on electronics. However, some attention is paid to the restrictions imposed on a computer system by the physical properties of the electronics.

Module level design can be done by putting together the fundamental devices defined at the beginning of this section, to form the practical components of the computer. From a functional standpoint these components can be divided into the following categories:

Processing element—any hardware element that is primarily used for processing data (i.e., accepting data, performing some operation on the data, and returning results).

Memory element—any element whose primary purpose is to store infor-

mation (data or instructions). A memory element can accept information, hold it until it is requested, and then output it.

Input/Output (I/O) element—any piece of hardware whose primary purpose is to accept information from or send information to the outside world. By the outside world we mean a person or machine that is distinct from the computer (e.g., a machine tool or another computer).

Link or bus—any connection through which one part of the computer conveys information to or accepts information from another part of the computer.

Interface—any piece of hardware whose primary purpose is to connect together any of the above four types of elements in such a way that the signal levels and timing requirements of the elements are matched by the interface.

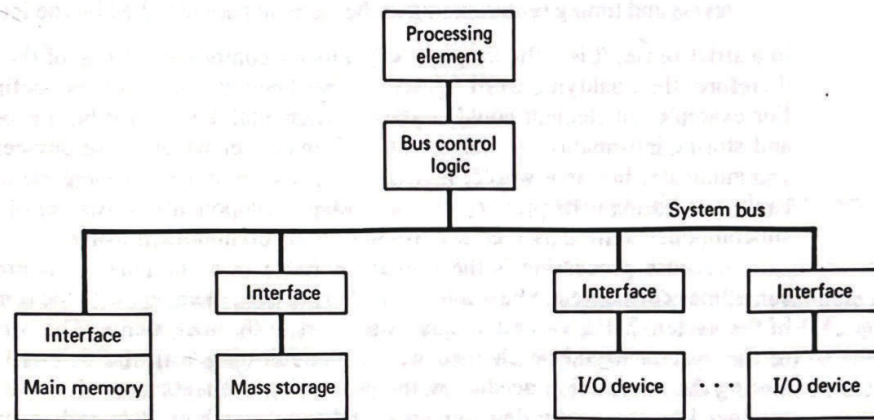
In a strict sense, it is difficult to classify a major component as one of the above; therefore, the qualifying word “primary” has been included in these definitions. For example, an element could contain fundamental devices for both processing and storing information as well as internal links over which these devices could communicate, but as a whole, it would be considered a processing element because processing is its primary function. Major components consisting of several subcomponents are classified according to their predominant usage.

Because processing is the primary purpose of a computer, a computer is sometimes considered to be a collection of processing elements with the remainder of the system being viewed simply as support to these elements. The remainder of the system would be charged with communicating with the external world, storing the information needed by the processing elements as well as the results produced by the processing elements, and supplying both data and instructions to the processing elements in a timely fashion. It is the processing elements that contain the conversion devices for converting the instructions into control signals that determine the computer’s actions.

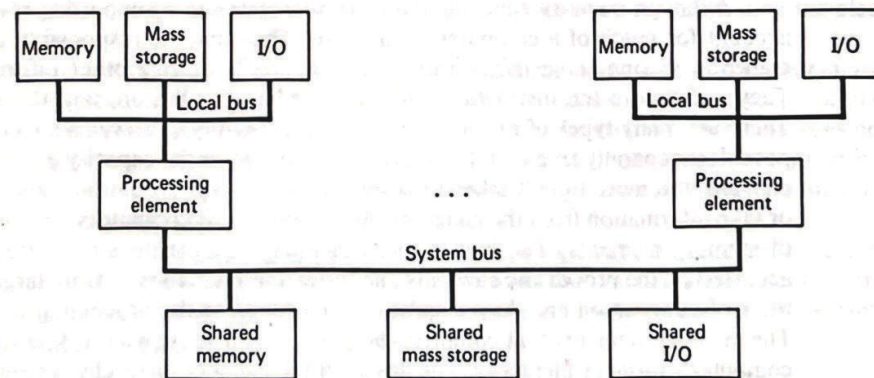
Although memory elements have been relegated to a supporting role, they account for much of a computer’s hardware. They may be responsible for permanently storing some information and temporarily storing other information. They *must* store the instructions, the data to be operated on, and the results. There are many types of memory elements, and each type serves a specific purpose. It is generally true that, for a given cost, the larger the capacity of a memory element, the more time it takes to access the element (i.e., put information into or take information from the element). As a result, most computers include levels of memory elements. The higher speed elements are the ones most frequently accessed by the processing elements and those that permanently store large quantities of information are slower and more remote from the processing elements. The memory elements that comprise these levels constitute what is known as the computer’s **memory hierarchy**. The design of the memory hierarchy is determined by the requirements for supplying instructions and data to the processing elements.

The physical arrangement of the parts of a computer system or subsystem is called its **architecture**. The architecture of a computer can be considered as a

whole or by the architecture of one of its major components. Figure 1-4 gives two typical computer architectures. In Fig. 1-4(a), all the elements are connected to a single link, called the **system bus**, and the computer is said to have a **single bus architecture**. This configuration also contains only one processing element and is, therefore, also referred to as a **single processor architecture**. Having a single bus and single processing element considerably simplifies a design and makes it much easier to discuss the basic concepts of computers. It is this type of architecture that is assumed in most of this book.



(a) Single bus, single processor system



(b) Multiprocessing system

Figure 1-4 Typical computer architectures.

4

Fundamental Computer Operations

The function of a processing element is to retrieve the data it is to operate on, operate on the data, and store the results. This chapter introduces the fundamental operations of processing elements and describes how they are told which operations to perform. The examples in this chapter will consist of short sequences of instructions that do simple, well-defined tasks. How to put such sequences together to form programs will be the subject of Chap. 5.

In this chapter we are concerned only with the simple configuration depicted in Fig. 4-1, which consists of a processing element, a single main memory element that contains all of the instructions and data that are needed by the processing element, and a bus that joins the processing and memory elements together. We are not concerned with how the instructions and data got into the memory element or what happens to the results after they are stored in the memory element. Our interests are restricted to the interaction between the processing and memory elements and the operations carried out by the processing element.

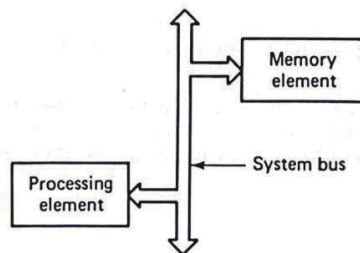


Figure 4-1 Processing configuration assumed in Chapter 4.

Figure 4-2(a) illustrates the cycle of activity that must be repeated while instructions are executed. This cycle consists of inputting, or **fetching**, the instruction from memory, executing the instruction (which may include fetching data from memory or storing results back into memory), and determining the location of the next instruction. The most important characteristic of a processing element is its speed, which is determined by the average time it takes to complete this cycle. Each of the cycle's three components takes time, but determining the location of the next instruction normally overlaps the execution of the present instruction. Thus, the cycle time is, for the most part, divided between the fetching and executing components and both have a direct bearing on a processing element's speed.

Processing elements are constructed so that the next instruction is taken from the next sequential memory location unless the present instruction indicates otherwise (i.e., the default location of the next instruction to be fetched is the one whose address sequentially follows that of the current instruction). Normally, only a few instructions can cause an instruction to be taken out of sequence. They are called **branch instructions** and may be **unconditional**, in which case the branch out of sequence is always taken, or **conditional**. For conditional branches the branch may or may not be taken depending on the state of the processing element at the time the branch decision is made. For branching purposes, this state is determined by the contents of a special set of flip-flops within the processing element that are referred to as **condition flags** (or **condition codes**). The condition flags are set or cleared according to the results of previous instructions (e.g., one of the flags may be set or cleared depending on whether the result of an arithmetic operation is zero or nonzero). It is conditional branch instructions that give a processing element its decision-making capability.

Shown in Fig. 4-2(b) are the principal components of the elements in Fig. 4-1. The memory is made up of groups bits, called **locations**, with each location being used to store a basic unit of information (e.g., a byte). Each location has associated with it a unique identifying number, called its **address**, which is used to put information into and take information from the location.

Included in the processing element are the bus control logic for interfacing to a system bus, the control unit that is responsible for decoding and executing the instructions, the arithmetic/logic unit (ALU) that performs the arithmetic and logical operations, and an internal memory consisting of a set of registers, called the **working registers**, which hold information that can be accessed quickly. The control unit also includes a set of registers. They hold instruction-related information that is needed during the execution of the instructions. The control unit's set of registers typically includes the:

Instruction register (IR): For holding the current instruction while it is being decoded and executed.

Program counter (PC): For holding the address of the next instruction.

Processor status word (PSW): A collection of flip-flops that includes the condition flags and other status information needed during the operation of the processing element.

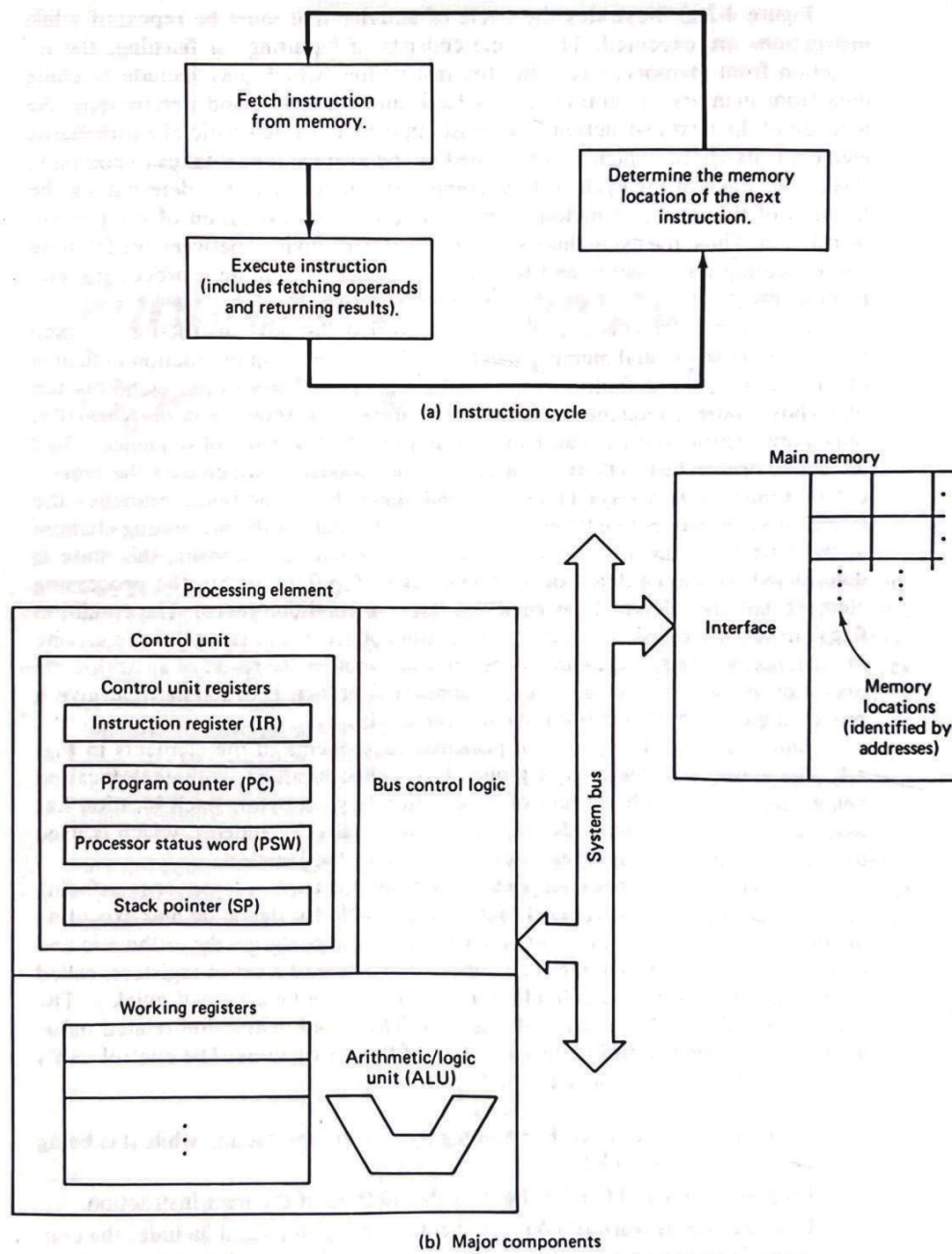


Figure 4-2 Processing operation and major components.

Stack pointer (SP): Used for accessing a special part of memory called a **stack**. (Stacks are discussed in Sec. 4-3.)

Figure 4-3 is a flowchart of the activity that takes place during the fetch and execution of an instruction. After the execution of an instruction is complete and the PC has been updated, the address of the next instruction is sent to the memory. The memory returns the instruction, which is then stored in the IR where it is decoded. If it is an unconditional branch instruction or a conditional branch whose

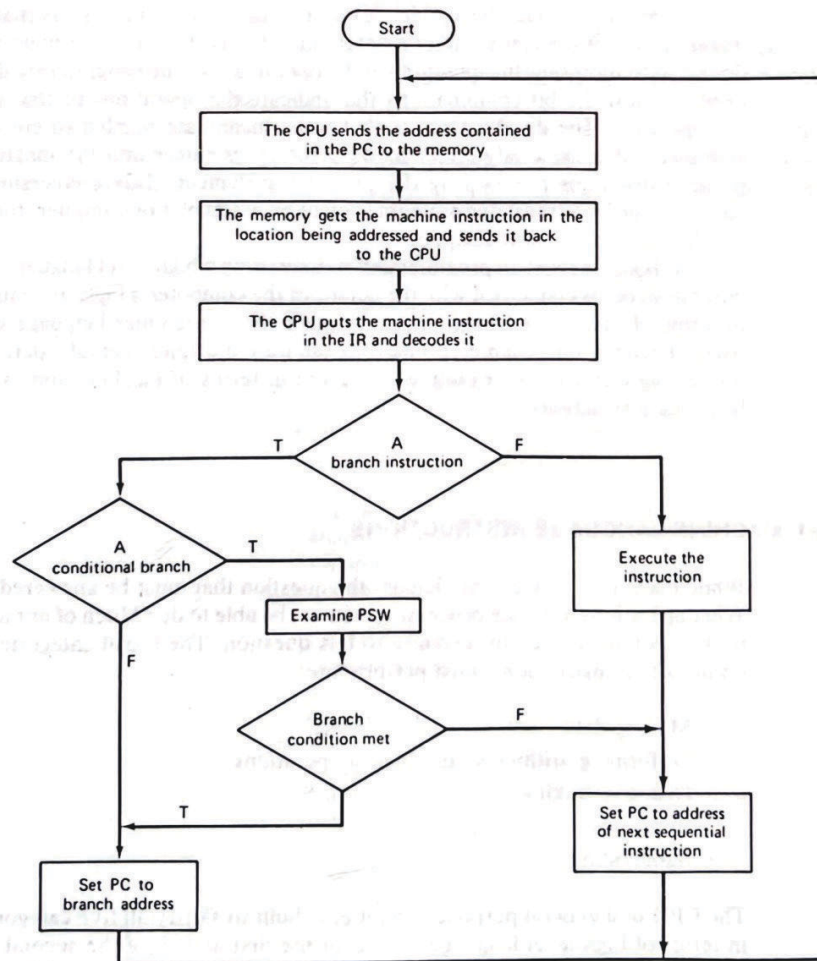


Figure 4-3 Flowchart of instruction execution activity.

condition is satisfied, then the PC is filled with the address of the location that is to be branched to, an address indicated by the branch instruction. Otherwise, the next sequential address is put into the PC. In any event, after instruction execution is complete the new contents of the PC are sent to the memory and the cycle is repeated.

For a processing element to decode instructions, all instructions must be presented to it as combinations of 0s and 1s. As indicated in Chap. 1, such instructions are referred to as machine language instructions because they can be directly understood by the processing element's decode logic. However, because it is extremely tedious to write programs in machine language, almost all programs are written in an assembler or high-level language. The advantage is that meaningful strings of alphanumeric characters may be used to indicate what is to be done and to represent the quantities to be operated on. The programmer does not need to know the bit combinations that indicate the operations or that point to the operands. The disadvantage is that some means are needed to convert the statements that are easily understood by the programmer into the machine language instructions required by the processing element. This conversion is, of course, done by a translator program such as an assembler or compiler, the design of which is discussed in Chap. 5.

Although almost all programming is done using a high-level language, a computer designer is concerned with the details of the computer's logic and must think in terms of machine language instructions or their assembler language counterparts. Because this chapter primarily examines the fundamental operations of processing elements, its examples are given in terms of machine and assembler language instructions.

4-1 MACHINE LANGUAGE INSTRUCTIONS

While planning a processing element the question that must be answered first is: What operations must the processing element be able to do? Much of our attention in this section is given in response to this question. The broad categories of operations that instructions must perform are:

- Moving data
- Performing arithmetic and logical operations
- Decision-making
- Repeating
- Subdividing

The CPU of a general purpose computer is built to satisfy all five categories and, in terms of high-level languages, some of the first and all of the second of these categories are satisfied by assignment statements, the third by IF-THEN-ELSE statements, the fourth by looping, and the fifth by procedures, or subprograms.