MICROPROCESSORS/ MICROCOMPUTERS

An Introduction

Donald D. Givone

Robert P. Roesser

MICROPROCESSORS/ MICROCOMPUTERS

An Introduction

Donald D. Givone
Associate Professor of Electrical Engineering
State University of New York at Buffalo

Robert P. Roesser
Associate Professor of Electrical Engineering
University of Detroit

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Düsseldorf
Johannesburg London Madrid Mexico Montreal New Delhi
Panama Paris São Paulo Singapore Sydney Tokyo Toronto

This book was set in Times Roman.
The editors were Charles E. Stewart and Stephen Wagley;
the cover was designed by Antonia Goldmark;
the production supervisor was John Mancia.
The drawings were done by Santype International Limited.
Fairfield Graphics was printer and binder.

MICROPROCESSORS/MICROCOMPUTERS: AN INTRODUCTION

1234567890FGFG7832109

Library of Congress Cataloging in Publication Data

Givone, Donald D
Microprocessors/microcomputers: an introduction.

(McGraw-Hill series in electrical engineering)
Bibliography: p.
Includes index.
1. Microprocessors. 2. Microcomputers.
I. Roesser, Robert P., joint author. II. Title.
QA76.5.G516 001.6'4'04 79-13543
ISBN 0-07-023326-8

PREFACE

Since the development of the integrated circuit in 1959, there has been continuous interest in the fabrication of new electronic devices capable of performing complex functions. One of the most dramatic consequences of integrated circuit technology was the microprocessor in 1971. Although originally developed as calculator chips, microprocessors soon were recognized as having great potential as the heart of dedicated digital processors because of their low cost and programmable capability. In essence, the central processing unit of the digital computer was encompassed in the form of an integrated circuit.

It is the intent of this book to provide the reader with a foundation on digital computer principles with emphasis on the behavior, operation, and applications of microprocessors and microcomputers. This book develops in detail the basic concepts that are important in the design of most microcomputer-based systems. These concepts include microcomputer architecture, memory structure, input/output facility, interfacing, and programming. No background computer knowledge is assumed. Material on number systems, Boolean algebra, and electronic circuits is included in order to make the text self-contained. However, some knowledge in electrical circuits is assumed.

Perhaps the most difficult decision we had to make in the writing of this book concerned the selection of a microprocessor for illustrative purposes. Since our objective is to provide fundamental concepts in microcomputer design, we feel that a noncommercial microprocessor best serves our purposes. In this way, we hope that the microprocessor itself will not mask the concepts being developed. Although a chapter expounding upon the commercially available microprocessors could have been included, we feel this information is best obtained from the various manufacturers.

The material in this book is suitable for a one-semester or two-quarter course on the junior or senior level in electrical or computer engineering. Since it is self-contained, the book is also well-suited for self study. It is realized that some of the material in Chaps. 2, 3, and 4 will most probably have been included in

previous courses. However, to maintain completeness and to allow for different backgrounds, these chapters provide detailed development of their material. It should therefore not be necessary to cover certain portions of these chapters.

We would like to express our thanks to the many people who have helped and encouraged us during the writing of this book. A special acknowledgment is given to our children who did not always understand why we could not spend more time with them while engaged in this project. We hope they will be rewarded with a better tomorrow built out of the new advances of today. Finally, we would like to express our appreciation to Mrs. Marilyn Hutchings for her patient typing of the final manuscript.

Donald D. Givone
Robert P. Roesser

CONTENTS

CHAPTER

ONE

MICROPROCESSORS AND MICROCOMPUTERS: AN OVERVIEW

Microprocessors are new and fascinating logic devices that are having pronounced effects upon our lives. Microprocessors can be found in pocket calculators, checkout terminals in stores, home appliances, office equipment, scientific instruments, medical equipment, and video games, to name just a few occurrences. Furthermore, every day new applications are being found and new microprocessor-based products are being developed. Their potential effects upon our lives are almost beyond imagination. This book will introduce the reader to the internal workings of microprocessors and larger systems that contain microprocessors. This introduction requires some exposure to logic design principles, electronics, and programming. These concepts are applied where they are necessary for the effective utilization of these new logic devices.

This chapter will provide an overview of the behavior and structure of microprocessors and microcomputers. A great deal of terminology will be introduced during this discussion. The intent, at this time, is to establish in the reader a general feeling for the numerous concepts involved in microprocessor systems and the interrelationships of these concepts. In the ensuing chapters, the details will be developed and exemplified. However, if during the course of study the broad picture established in this chapter is kept in mind, the reader will be able to appreciate more fully the operation and design of a microprocessor system.

1.1 A NEW ERA OF COMPUTATION

Digital computers have had a great deal of influence upon our society and manner of living since 1951, when the first commercial digital computer (Univac I) became available. A new technology had emerged. Such terms as "digital computation,"

"logic design," and "programming" became concepts integral to science and engineering. However, the diversity of these concepts frequently caused a split in interest. For example, there were those whose interest was in using or programming computers (software), while there were others whose interest was in designing computers (hardware). Although this dichotomy in interest may have real foundations in the case of large-scale computers, the problems facing application programmers and computer designers became more interrelated with the introduction of minicomputers in 1965. These computers were no longer intended solely for data processing and problem solving, but were to become a part of systems requiring immediate computer decisions, called *real-time systems.*

The introduction of the microprocessor in 1971 closed this interest gap even further. An era of software logic design, or programmed logic, has resulted. In this era, programming concepts and logic design principles have merged to the point that their interactions require scientists and engineers to have complete familiarity with both the software and hardware principles of computers to utilize fully the potential of the microprocessor. It is toward an understanding of these principles and their interrelation that this book is directed, with the hope that the full beneficial potential of microprocessors can be realized.

1.2 MICROPROCESSORS AND MICROCOMPUTERS

Since this book is about microprocessors and microcomputers, let us begin by defining these two words. Precise and detailed definitions are really impossible since microprocessors and microcomputers deal with a dynamic technology, and as this technology changes, so must their definitions be modified.

One of the results of the advancement in solid-state technology is the capability of fabricating very large numbers of transistors (say, 1000 and over) within a single silicon chip. This is known as *large-scale integration.* A direct consequence of large-scale integration is the microprocessor. In general, a *microprocessor* is a programmable logic device fabricated according to the concept of large-scale integration. As will be seen, a microprocessor has a large degree of flexibility built into it. By itself it cannot perform a given task, but must be programmed and connected to a set of additional system devices. These additional system devices usually include memory elements and input/output devices. In general, a set of system devices, including the microprocessor, memory, and input/output elements, interconnected for the purpose of performing some well-defined function, is known as a *microcomputer* or *microprocessor system.*

Although microcomputers have the general characteristics of digital computers, a notable property of microcomputers is their relatively low cost and small size. This has greatly contributed to their popularity and success. While large mainframe computers and minicomputers have more computational power than microcomputers, this power is not always needed in every application. Further, the cost of large computers and minicomputers has often precluded their adoption into certain electronic systems which would have otherwise benefited from their

inclusion. The microprocessor has now made it possible to use a programmable device in a logic system where cost constraints, rather than speed and computational power, are important.

Microprocessors are finding new applications every day. Currently, they are used for instrumentation, point-of-sale terminals, intelligent terminals, and calculators. They are also vital elements in the control of machines, chemical processes, computer peripherals, traffic systems, major appliances, and automobiles. Finally, in the recreational area, these devices are responsible for a new group of hobbyists who are building their own computers.

1.3 A BASIC MICROCOMPUTER ORGANIZATION

A basic computer system consists of five functional units: the *input unit*, the *memory unit*, the *arithmetic unit*, the *control unit*, and the *output unit*. One such basic system is illustrated in Fig. 1.1.

The physical components and circuits that comprise a computer system are known as its *hardware*. These circuits are capable of performing only a small number of different operations. Any additional operational capabilities of the computer must be accomplished by *programming*. A *program* is an organized collection of elementary computer operations, called *instructions*, that manipulate

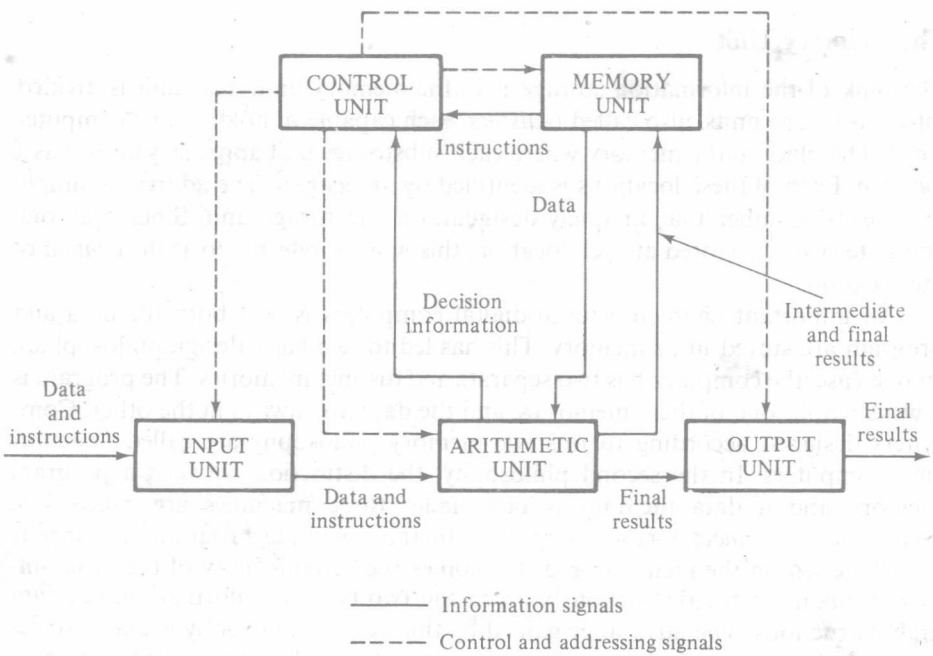

Figure 1.1 An organization of a basic computer system.

information, called *data*. The programs that are written for a computer are called its *software*.

The program and data are first stored in the memory unit via the input unit. The individual instructions of the program are then automatically entered, one at a time, into the control unit, where they are interpreted and executed. The execution usually requires data to be entered into the arithmetic unit, where the circuitry necessary for manipulating the data is contained. During the course of computation or at its completion, the derived results are sent to the output unit. The arithmetic unit and control unit together are normally called the *central processing unit* (CPU). The central processing unit of a microcomputer system is the microprocessor.

In addition to the memory unit, the other computer units are also capable of storing information. Information is stored as groups of binary digits (called *bits*) in storage devices called *registers*. Essentially the operation of a computer can be regarded as a series of information transfers from register to register with possible information modification (e.g., addition) being performed between transfers. The group of binary digits handled all at the same time by the computer is known as a *word*, and the number of binary digits that make up the word is the *word length*. A word is the basic logical unit of information in a computer. An instruction or data consist of one or more words. Microprocessors are typically available with 4, 8, 12, and 16 binary-digit word lengths. Since an 8 binary-digit word length is so common, it is given the special name *byte*.

The Memory Unit

The bulk of the information storage is in the memory unit. This unit is divided into substorage units, also called *registers*, each capable of holding one computer word. The place in the memory where each substorage unit appears is known as a *location*. Each of these locations is identified by an *address*. The address is simply an integral number that uniquely designates a substorage unit. Since only one computer word is stored at each location, this word is referred to as the *content* of the location.

An important characteristic of digital computers is that both the data and program are stored in its memory. This has led to two basic design philosophies. In one case, the computer has two separate and distinct memories. The program is always within one of these memories, and the data are always in the other. Computers designed according to this two-memory philosophy are called *Harvard-type* computers. In the second philosophy, the distinction between a program memory and a data memory is not made; these machines are called *Von Neumann-* or *Princeton-type* computers. In this case, the program can appear anywhere within the memory, and it becomes the responsibility of the programmer to maintain the distinction between the two types of information, i.e., data and instructions. The advantage gained by this design philosophy is that instructions can be treated as data, so that the computer is allowed to modify its own

instructions. Microcomputers have been designed according to both philosophies. This book will always assume Von Neumann-type computers.

Because of their low cost, microprocessors are playing a significant role in dedicated applications. Large general-purpose digital computers are continually reprogrammed so that they can be used for solving a large variety of problems. Microcomputers in a dedicated application do not need this flexibility. Once the program is written and tested, it usually is not subject to further change. Thus, microcomputers frequently have two different forms of memory within the memory unit. There is the *read-only memory* (ROM) and the *read/write memory* (RWM).† Once information is placed into a read-only memory it cannot be modified easily, if at all. The read-only memory, with its lower cost, is used to hold the program and any constant data, while the read/write memory is used for holding information that is subject to change.‡

The Arithmetic Unit

The arithmetic unit is where most of the data manipulations occur. These manipulations involve both arithmetic and logic computations. As will be seen in Chap. 7, where a typical microprocessor is described, the allowable computations are very elementary. The more complex mathematical operations have to be performed by means of a program which utilizes the allowable operations.

Typically, the most important register in the arithmetic unit is the *accumulator*. This register normally contains one of the operands prior to a computation and the result after the computation.§ In addition, several auxiliary registers, called *scratchpad registers*, frequently appear in the arithmetic unit to facilitate the writing of programs.

Also included in the arithmetic unit are *flag bits*. These bits provide status-type information that can be important for determining the course of computation. For example, a flag indicating that the result of a computation is 0 might be provided. The programmer can then use the detection of this condition for decision making. Typically, if a particular computation produces a result of 0, one set of future computations is performed; if the result is not 0, an alternate set of future computations is performed. The set of flag bits indicating the results of computa-

† It has become conventional to refer to read/write memory as *random-access memory* (RAM), even though both read-only and read/write memories generally have the property of random access. Random access refers to the fact that all locations within the memory are accessible in the same amount of time.

‡ The reader should be careful to note that the existence of a read-only memory does not imply a Harvard-type computer. In a Von Neumann-type computer the ROM locations may appear anywhere within the memory unit. The key point is that the microprocessor does not know whether it is getting its information from ROM or RWM.

§ In some microprocessor designs, there are several accumulators, called *general-purpose registers*, that are used for the computational processes.

tions and tests is often kept (along with other machine status information) in a special register called the *program status word* (PSW).

The Control Unit

The function of the control unit is to oversee the operation of the computer. It automatically receives the instructions, one at a time, from the memory unit. Then it decodes each instruction and generates the necessary signals to provide for its execution. For the control unit to obtain an instruction, it must first know the instruction's location in the memory. Normally the instructions are in sequential order, and their location is indicated by a *program counter* within the control unit. Furthermore, to decode and provide for instruction execution, the control unit must hold the current instruction. It is the *instruction register* that stores the instruction within the control unit for these purposes.

In order for the control unit to interpret an instruction correctly, the instruction must have a definite organization, called the *instruction format*. The exact instruction format is dependent on the particular microprocessor. However, certain information must be included within an instruction. Most significant are the *operation code* and, in some instructions, an *address*. The operation code is a set of binary digits that uniquely define what operation is to be performed during the execution of the instruction. The address portion of an instruction, if it exists, indicates the location (e.g., in memory) that must be accessed to execute the instruction. For example, if an addition operation is to be performed, the address portion of the instruction may indicate the location of the addend.

It is important that the reader fully understand the use of the word "address" and the difference between the address of a location and the content of the location. There is an address portion in the format of some instructions. This is the numerical designator of a location associated with an operand. However, prior to the execution of an instruction, the instruction itself is stored in the memory. Thus, the instruction itself has an address associated with it. In general, this address is not the same as the address portion in the instruction format.

Maintaining synchronization between the various computer units is another function of the control unit. This is achieved by means of a *clock*. Several clock periods are needed to handle an instruction. In general, an instruction must be fetched from memory, decoded, and then executed. The fetching, decoding, and executing of an instruction is broken down into several time intervals. Each of these intervals, involving one or more clock periods, is called a *machine cycle*. The entire period of time associated with the fetching, decoding, and executing of an instruction is called an *instruction cycle*.

The Input/Output Units

The final two units of a computer are the input and output units. These units are the contact between the computer and the outside world. They act as buffers, translating information between the different speeds and languages with which

computers and humans, or other systems, operate. The input unit receives data and instructions from the outside world, which eventually are entered into the memory unit. The output unit receives the computed results and communicates them to the operator or to another system. The input and output devices are known as *peripherals*. Examples are paper-tape readers/punches and typewriters. The places of contact between the input/output devices and the microprocessor are called the *input/output ports*. The input and output ports are also addressable so that several input and output devices can be connected to the microprocessor.

It is characteristic of *digital* computer operation that all information is handled in discrete form, i.e., with finite numbers. Frequently a digital computer must communicate with another system not capable of handling discrete information. Nondiscrete information is termed *analog* or *continuous*. In such a case there must be a transformation between continuous and discrete information. The input and output devices capable of performing this type of transformation are called *analog-to-digital converters* and *digital-to-analog converters*.

Figure 1.1 indicates that the input and output units directly communicate with the arithmetic unit. This is not the only possible organization. To achieve greater overall speed in a computer, it is frequently desirable to allow the input and output units to access memory directly rather than via the arithmetic unit. This is known as *direct memory access* (DMA). The direct memory access concept will be developed in Chap. 9.

Buses

In a microcomputer the various units are connected by *buses*. A bus is a set of lines over which information is transferred from any of several sources to any of several destinations. One common bus structure of a microcomputer is shown in Fig. 1.2.

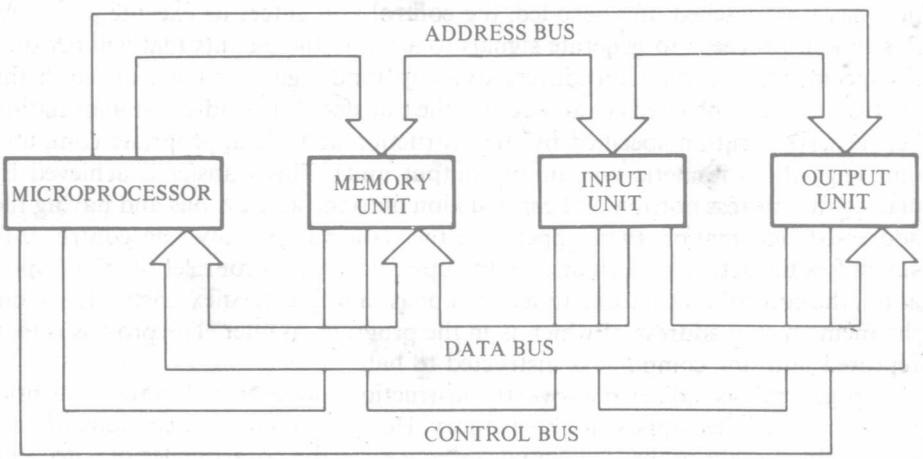

Figure 1.2 The bus structure of a typical microcomputer.

This structure consists of three buses. The *address bus* is unidirectional; i.e., information flows in only one direction. This bus is used to transmit an address from the microprocessor to the memory, input, or output unit. The *data bus* is bidirectional; i.e., information can flow in either direction on these lines, and is the path for data flow. Finally, the *control bus* is the set of lines over which signals travel to maintain timing and status information. Some of these lines are bidirectional, while others are unidirectional. For this reason, no direction for this bus has been indicated in the figure.

1.4 MICROCOMPUTER OPERATION

In the previous section, the functions of the various computer units were explained. In this section, let us consider the way in which the units interact and the dynamics of information flow.

The control unit progresses through three phases of operation: fetch, decode, and execute. After the program and data have been entered into the memory, the address of the first instruction to be executed is placed into the program counter, and the control unit is set to its fetch phase. At this time, the content of the program counter is placed onto the address bus so that the corresponding instruction can be retrieved from memory. The instruction stored at the location specified by the address in the program counter is sent to the instruction register in the control unit via the data bus. Since the instructions are stored in sequential order, the program counter is then incremented by 1 so that it will contain the address of the next word in the program. The control unit now decodes the operation code in the instruction word just received. If it is determined that the instruction consists of more than one word, the fetch phase is repeated a sufficient number of times to retrieve the entire instruction. Each time the fetch phase is performed, the program counter is incremented. After the entire instruction has been fetched and decoded, the control unit enters its execute phase. At this time it proceeds to generate signals to activate the circuits that will perform the specified operation. If an address for an operand is given in the instruction, the control unit then proceeds to arrange for the transfer of the addressed information between the location specified by the instruction and the appropriate computer unit (e.g., the arithmetic unit or the output unit). This transfer is achieved by placing the address portion of the instruction onto the address bus and having the addressed information then appear on the data bus. Finally, the control unit supervises the actual performance of the operation. Upon completion of the operation, the control unit returns to its fetch phase and gets the next instruction from the memory, the address of which is in the program counter. The process is then repeated until the computer is instructed to halt.

In general, as indicated above, the instructions are executed in the sequential order in which they appear in the memory. However, there are occasions when it is desirable to ignore this sequential order, e.g., as the consequence of a decision based on a flag bit. In such a case the location of the next instruction may be other than the location immediately following that of the instruction currently being

executed. Instructions that permit this deviation are called *jump* or *branch instructions*. In the case of a jump instruction, the address portion of the instruction contains the location of the next instruction to be fetched by the control unit if the change in sequential order is to be carried out. Thus, upon decoding a jump instruction and establishing that a deviation in the instruction sequence is to occur, the control unit places that part of the current instruction having the address of the next instruction into the program counter. In this way, when the control unit enters the fetch phase, the appropriate next instruction is obtained.

1.5 TWO ADDITIONAL COMPUTER CONCEPTS

Before we conclude this overview of microcomputers, two additional concepts that increase the computer's versatility should be introduced. These are the *stack* and *program interrupts*.

The Stack

It has already been mentioned that jump instructions are used to break away from the sequential ordering of instructions currently being executed. A type of jump instruction, which differs from those previously discussed, allows the repeated use of a specific subset of instructions, called a *subroutine*.

During the course of a (main program) computation, a subcomputation, written as a subroutine, may need to be performed. The computer must jump from the main program to the subroutine, perform the necessary subcomputation, and then return to the point of exit in the main program. A record must be kept of the program counter prior to the jump to ensure the correct return of the computer. This is precisely what is done by a *jump to subroutine* instruction. In particular, the content of the program counter is stored, and then the jump is executed. A special *return from subroutine* instruction is used to retrieve the stored program counter so that reentry into the main program can be made. The place of storage for the program counter is typically in a *stack*.

In general, a stack is a set of registers that accept and return information on a *last-in first-out* (LIFO) basis. This means that only the top of the stack (last-in) is immediately accessible. Depending upon the design of the computer, the stack may appear in the control unit or as part of the memory unit.

By allowing the stack to hold several addresses, the nesting of subroutines becomes possible. That is, a jump from a subroutine to another subroutine can be performed, and even a jump from a subroutine to itself (called *recursion*). Clearly, the degree of nesting is dependent upon the storage capacity of the stack.

Program Interrupts

In many microprocessor applications, it is desirable to interrupt the computing process to service an external device. An interrupt can be performed by having the external device requesting attention transmit a signal to the microprocessor. The