rodnay zaks

microprocessors

from chips to systems

THIRD EDITION

MICROPROCESSORS

FROM CHIPS TO SYSTEMS

RODNAY ZAKS

THIRD EDITION

SYBEX

Cover Design by Daniel Le Noury

Library of Congress Card Number: 80-51035
ISBN 0-89588-042-3
First Edition Published 1977. Third Edition 1980.
Printed in the United States of America
Printing 10 9 8 7 6 5 4 3 2

CONTENTS

PREFACE

FOR WHOM?

No preliminary knowledge of computers or microprocessors is required to read this book, although a basic engineering knowledge is naturally an advantage.

Microprocessors, and other LSI chips, have made system design so simple that no significant scientific or electronic training is required.

This book is, therefore, intended for all those who have only a fragmented knowledge, or no knowledge of the microprocessor world, and who wish to understand all the concepts, techniques and components in a short time.

It can be used by non-specialists: students, hobbyists, as well as engineers. The reason is that it is complete and progressive.

A BRIEF HISTORY OF THIS BOOK

The author has been involved with industrial microprocessor designs in Silicon Valley since 1972, as well as general computer design before that date. In addition, he has taught microprocessor design to more than 2000 persons, from students to computer engineers. This book is the result of the author's experience in both design and teaching. It represents the seventh version of prior course and seminar books and publications by the author in the past few years. In particular, it includes the text of SYBEX seminars C10 and A1.

As a result of this extensive testing and debugging, the book should be completely self-sufficient. The structure of each chapter is progressive: from basic definitions to advanced applications - from principles to techniques to actual examples.

WHAT YOU WILL LEARN

Chapter 1 will introduce you to all the basic concepts and definitions.

Chapter 2 will show you, in detail, how an actual microprocessor operates.

Chapter 3 will present the other techniques and components required to implement the memory and the input-output functions.

Chapter 4 will discuss the relative merits of each major microprocessor.

Chapter 5 will show you how to assemble all the previous components into a system.

Chapter 6 presents applications: how to build them, what the differences are.

Chapter 7 is a more advanced chapter: how to interface our basic system to the usual peripherals (including the S-100 bus).

Chapter 8 presents the basic principles and techniques of programming.

Chapter 9 will show you the remaining problems, and available tools for developing an actual system.

Chapter 10 will present predictions as to future evolution.

HOW TO READ THE BOOK

Sequential reading is recommended, but not necessary. If you are not interested in, or not ready for, interfacing, you may skip Chapter 7. Similarly, if the internal operation of the microprocessor is not of interest to you, you may skip Chapter 2.

Because each chapter and each section are strongly structured, from simple to complex, it is possible to refer to random sections.

However, it is strongly recommended that the complete book be read completely once. Each chapter has been designed to teach you a specific aspect. Successive chapters build upon the knowledge presented in the preceding ones. Several exercises have no answer. You should be able to answer the questions, and feel sure you have the correct answer.

TO LEARN QUICKLY HOW A SYSTEM OPERATES

If your time is very limited, read Chapters 1-2-3 and 5, without even looking at the actual components. In a few hours, you will know how a system operates and is assembled.

MICROPROCESSORS

The microprocessor industry is probably the fastest growing one today. Its effect on the economy has been regarded as the "second industrial revolution." A complete microcomputer can be implemented now on a single chip for one dollar in large quantities. It brings a "programmed intelligence" to all the processes that surround us.

The simplicity of use is such that only two limitations remain: the ingenuity of the user, and his competence.

This book is directed at removing the second limitation.

1 FUNDAMENTAL CONCEPTS

INTRODUCTION

The fundamental concepts and definitions necessary to understand microprocessors and microcomputers will be introduced in this chapter.

The *microprocessor* is a new LSI component which implements most of the functions of a traditional processor in a single chip. New terms will be defined progressively throughout this chapter.

LSI stands for "Large-Scale Integration" and refers to the new technology by which several thousand transistors can be integrated on a single integrated circuit (IC).

A *chip* is the small rectangular piece of silicon on which this integrated circuit is implemented. The LSI manufacturing process will be described later in this chapter.

A *microcomputer* is a computer whose Central Processing Unit (CPU) has been implemented using an LSI microprocessor. The progress of LSI technology now allows the implementation of a complete simple computer on a single chip ("microcomputer-on-a-chip").

The first microprocessor was introduced on the market at the end of 1971 (I4004) and there are more than 30 different types of microprocessors on the market now. Only a few, however, sell in significant quantities. The typical sales price as of this writing is $10.00 in quantities of 100 and more (such a price has indicative value only and may change rapidly with market conditions).

A *microprocessor system* is a complete system including processing functions, memory, and input and output facilities. The microcomputer is naturally a microprocessor system. However, one generally thinks of a microcomputer as a desk-top computer, i.e., a system with a cabinet, front-panel and power supply intended as a general-purpose computer.

PRINCIPLES OF OPERATION

The principles of microprocessor systems' operation are naturally identical to those of any computer's operation. However, the new systems implemented with microprocessors often differ significantly from traditional systems. The fundamental reason is that *the processor itself has become one of the most inexpensive resources* in the system. As such, the processing element may be conceptually considered as a peripheral to other system components, in particular input-output sections. In short, one can expect that in the near future most of the so-called peripheral chips now available for use in microprocessor systems will incorporate a significant *processing module.* Peripheral chips are becoming "processor-equipped" and, therefore, programmable.

The fundamental concepts of the operation of a computer system will be reviewed here. In Chapter 3, the new LSI components available for building a microprocessor system, or a microcomputer, will be reviewed and examined in detail. Finally, in Chapter 5, these components will be interconnected into a complete working system.

The structure of a basic computer system is illustrated in Fig. 1-1. A system includes 5 fundamental units. At the center of the illustration appears the *Central Processing Unit* abbreviated CPU. The CPU includes 2 units: the *Control Unit* (CU), and the *Arithmetic-Logical-Unit* (ALU).

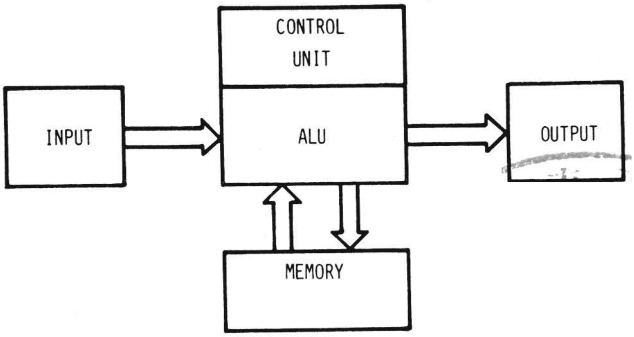

Fig. 1-1: A computer system includes 5 functional units

The function of the *Arithmetic-Logical-Unit* is to perform arithmetic and logical operations on the data passing through it. Typical arithmetic functions include addition and subtraction. Typical logical operations include "logical and", "logical or", and shift operations.

The responsibility of the *Control Unit* is to sequence the operation of the entire system. In particular, it will generate and manage all control signals necessary to synchronize operations and the flow of data within the Arithmetic-Logical-Unit, as well as outside it. It will control the flow of data on the address-bus and on the data-bus, and will manage or interpret the signals presented on the control-bus of the system. (A *bus* is a set of signals, or lines, grouped by function – the three standard buses in a microprocessor system are the data bus, the address bus, and the control bus.) One of the essential roles of the control unit is to *fetch, decode, and execute* successive instructions stored in the memory system. Such a sequence of instructions is called a *program*. The CU is generally physically associated with the ALU that it controls, and the combination of the CU and ALU is called a *Central Processing Unit* or CPU. A microprocessor is basically a CPU on a single chip.

The CPU need not necessarily be implemented as a *single* component, and the CU can be separated from the ALU. In particular, *bit-slices* now available implement the ALU section of a traditional computer, exclusive of the control section, which must be designed and assembled separately. Bit-slice architecture will be examined in Chapter 5.

Back to Fig. 1-1, the *memory module* appears underneath the CPU, at the bottom of the illustration. The memory of any computer system is used to memorize information. Physically, several kinds of memories can be distinguished, depending on whether one can write information in the memory, and then read it, or whether one can only read from it. These two basic types of memory are called respectively *RAM* (Random Access Memory) and *ROM* (Read-Only Memory).

An *RAM* is a memory in which data can be either written or read. The typical cycle time of such a memory is 500 nanoseconds (ns) to 1 microsecond (us) for MOS LSI RAM's. The *ROM* is a memory which can only be read once data has been deposited in it. Its main advantage is to be *non-volatile*, whereas standard RAM's are volatile, i.e., lose their data once power disappears. Functionally, the memory contains two types of information: *programs* and *data*.

A *program* is a sequence of instructions which has been written by the user, then coded into the binary system, so that it could reside in an electronic memory. Each of the successive *instructions* of a program will be fetched, under the supervision of the control unit, and deposited in a special register of this control unit, where it will be decoded and executed. For example, a typical instruction might add the contents of two registers together, and deposit the results back in a third register. This process will be studied in Chapter 2.

The *data* contained in the memory will be processed by the Arithmetic-Logical Unit. Data may have a variety of formats. The most important ones will be described in Chapter 8. Typically, data are numbers or characters, represented in the binary system.

The *structure* of a memory will be described in the next section. The *technical characteristics* of LSI memories will be examined in Chapter 3.

Back to Fig. 1–1, the two remaining modules are respectively the *input-module* and the *output-module.*

The *input-module* appears on the left of the illustration and supplies data to the ALU. It can be, for example, a keyboard, or a sensor (temperature sensor, presence detector, pressure sensor). In a typical microcomputer, the usual input device is a keyboard.

The *output-module* will display to the outside world the data coming out of the ALU. Typical output modules are LED's (Light-Emitting-Diodes) or LCD's (Liquid-Crystal-Displays), which are extensively used on digital watches and pocket calculators, printers, lamps, or any control mechanism (a motor or a relay, for example).

The input and the output modules are normally not connected directly to the ALU but to the *system's buses,* i.e., to the address-bus, the data-bus, and the control-bus. For this reason, the data coming in from the input module does not necessarily *need* to transit through the ALU to reach the memory or perhaps the output module. However, in the architecture of most systems and, in particular, of microprocessor systems, this is indeed the case. All data in a typical microprocessor system usually transits through a special register of the ALU, called the *accumulator.* However, the possibility exists to implement other transmission schemes. This will be used, in particular, in the case of the *DMA* (Direct-Memory-Access-Controller), to be studied in Chapter 3.

BUSES

A bus has been defined as a set of signals grouped by function. The elements of a system are interconnected by means of three buses: the data-bus, the address-bus, and the control-bus.

The *data bus* transmits *data* between units. An 8-bit microprocessor requires an 8-bit data-bus in order to transmit 8 bits of data in parallel. The data-bus is bidirectional (transmits in both directions).

The *address-bus* is used to *select* the origin or destination of signals transmitted on one of the other buses. Typically, it is used to select a register within one of the system units, which will be used as a source or destination of data. Traditionally, a standard address-bus has 16 lines, i.e., can address 2^{16} = 64K devices (1K = 1024 in computer jargon).

The *control-bus* is used for the *synchronization* of the system. It carries status and control information both to and from the microprocessor unit (the standard abbreviation for the microprocessor unit has now become MPU). A minimal control-bus requires at least 10 or preferably more control lines to be useful.

EXAMPLE: A POCKET CALCULATOR

As an example of a simple computing system, let us examine the operation of a pocket calculator. The modules appearing on Fig. 1-2 have been grouped differently than in the previous illustration. The *microprocessor* implements the CPU, i.e., the control functions and the computing functions. The *memory* stores the program and the data. The *input-output* module is the special interface required to connect the input module and the output module to the microprocessor. The input module here is a hexadecimal keyboard, i.e., a 16-key keyboard. The output module is an LED display.

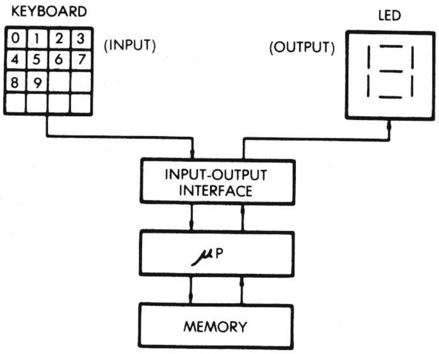

Fig. 1-2: A pocket calculator is a simplified microcomputer

A typical sequence is the following:

1. The user types in a number.

2. This number will be stored in a register of the system, contained in the microprocessor. It is assumed here that this is a standard model containing two internal registers. Each register may store an 8-digit decimal number, plus sign. These two registers are sufficient in the case of simple operations, such as "+" and "–", and no data need be stored in the memory. At this point, the memory would be required only to store the *program* . In fact, this memory is not necessarily *external* to the microprocessor itself. For clarity, on the illustration, the memory appears as distinct from the microprocessor itself.

3. Once a complete decimal number has been stored in a register (up to 8 decimal digits), an operation will be specified at the keyboard. This operation specified will be memorized in a special microprocessor register until it can be executed. Let us assume that a "+" has been specified.

4. The second operand required by the specified operation is then typed in by the user at the keyboard. Again, a complete decimal number is accumulated, until the user hits another operation key such as "=".

5. Specifying the "=" operation is equivalent to ordering the execution of an *arithmetic program* determined by the *operator* which had been saved in a special register ("+" in our case). An addition program is then executed: the instructions of that program, stored in the memory are executed. As a result, the addition is performed and the result is deposited in one of the initial operand registers, called the *accumulator.* This result can now be displayed to the user.

6. The result is sent from the accumulator (contained in the microprocessor) to the LED display unit, where it is displayed to the user. This result remains stored in the internal accumulator until it is cleared explicitly through the use of the "C" key, or else until the user specifies a new operation.

This simple sequence illustrates the use of the main modules of a computing system: the input and output devices, the memory (which here was storing only a program) and the Central Processing Unit, which performs operations and sequences the system.

Most CPU's incorporate a small amount of memory implemented in fast internal registers, for the sake of efficiency. In the above example, the processing unit incorporates at least two registers capable of storing decimal numbers, and a small register capable of storing the operation specified. Conceptually, these registers can be considered as part of the memory of the system. Several "levels" of memory are available in a system. Registers are the fastest memory level.

The techniques for reading information from a keyboard, or transforming a decimal number into a code suitable for driving LED displays will be studied in detail in Chapter 7 (Interfacing Techniques).

THE MEMORY

The memory of the system is used to store both the programs which will execute on the processor, and the data which will be manipulated by the system. The memory can be implemented in at least three ways:

1. **The internal registers** which are usually part of the ALU module provide the fastest level of data memory available to the system. Typically, the contents of internal registers can be accessed by the system in less than 100 ns (1 nanosecond = 10^{-9} s).

2. **The main memory** of the system is usually implemented in one or several components and has a size typically ranging from 1 to 64K words (1K = 1,024). In short, the main memory of a system is normally called "the memory." In the case of microprocessor systems, this memory is implemented in MOS/LSI technology, and can even reside directly on the microprocessor chip itself. The device is then called a *microcomputer-on-a-chip,* as it incorporates all the logical elements of a computer on a single chip. "Standard" microprocessors require an external memory. Typical speed is 300 - 600 ns.

3. **The mass memory.** In view of the relatively high cost of the fast main memory, it is impractical to consider a system using more than 32K or 64K of main memory. For this reason, special peripheral devices are used to store large quantities of data on inexpensive supports. The two most used peripherals for microprocessor systems are the magnetic tape and the floppy disk. The magnetic tape is usually a tape cassette. A floppy disk can be either a regular floppy disk or a micro-floppy. These devices will be described in Chapter 7 (Interfacing Techniques).

MEMORY ORGANIZATION

A memory is logically organized in words. A word is one logical unit of information, consisting of 4, 8, 12, or 16 bits (a *bit* is a *binary digit*). An 8-bit microprocessor requires 8 bits of data, and the word size for an 8-bit microprocessor is therefore 8 bits. Its memory will be logically structured in 8-bit words.

8 bits are called a *byte;* 4 bits are called a *nibble.* There is no other specific word for other sizes. The word size of an 8-bit microprocessor is a byte. For a 16-bit processor, it is 2 bytes.

The logical organization of a typical memory appears in Fig. 1-3. The width of a memory is its number of bits, i.e., its word-size. Bits are normally numbered from 0 to n. In the case of an 8-bit microprocessor (which will be our "standard microprocessor" from now on) the position of the bit within the memory will, therefore, be referenced by a digit from 0 to 7.

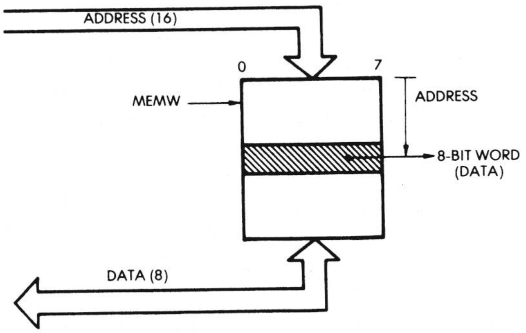

Fig. 1-3: Logical organization of a memory

QUESTION: *Why are bit positions referenced by digits from 0 to 7, instead of 1 to 8?*

The answer is that each digit from 0 to 7 represents the *binary position* of the bit in the word. In the binary system, the right-most digit (bit 0) represents 2^0. The next left-most bit, bit 1, represents 2^1, and so on.

Vertically, the height of a memory is its size, or its number of words. The position of the word within the memory is called its *address.* The first word of the memory has address 0, the next one address 1, and so on. For decoding reasons, the size of the memory is normally a power of 2. For example: 256, 512, 1K, 2K, 4K words.

QUESTION: *What is the address of the nth word of the memory?*

ANSWER: n–1.

In order to read the contents of a word within the memory, it is necessary to specify its address. Each memory module is therefore connected to the address bus. A typical address bus includes 16 lines, so that it may specify up to 2^{16} = 64K device locations. If the actual memory size is smaller than 64K, fewer lines would be required. In order to reference one word in the memory, a bit pattern will be sent on the address bus, which will specify the address of the desired word in the memory. The bits coming from the address bus are directed toward decoders. A *decoder* selects a word within the memory. In response to a control signal, such as READ, for a read operation, or WRITE, for a write operation, a word will be read or written from or into the memory. In the case of a read operation, a word will be fetched from the memory; after a time called *access-time,* it will become available on the output pins of the memory chip. These pins are connected to the data bus (8 bits, since words are 8 bits wide in our example).

QUESTION: *Why are only 8 bits of data coming out of a memory which is receiving 16 bits of address?*

ANSWER: This is an important point to clarify. There is no direct relationship between the number of bits coming out of the memory on the data and the number of address bits. The address bits specify a position within the memory and are used to select one word location through special decoders. The data word corresponding to the selected position can be of arbitrary length, from 1 to p bits. As an example, a small memory chip might include only 64 words x 8 bits. In this case, the address bus required to select a word in this memory would require only 6 lines (2^6 = 64). However, for each of the specified 64 addresses which might come in on this reduced address bus, there would still be 8 bits of data going out of the memory during a read operation. Conversely, if the memory should be large, such as 64K locations, 16 bits of address would be required, and there would still be 8 bits of data on the data bus.

In the case of a WRITE operation, the sequence is analogous: an address is specified on the address bus and data are presented on the data bus. An order is specified to the memory (through the control bus), and the memory writes in the specified memory location the contents presented on the data bus (8 bits in our example). The time required to write data in the memory is called the *memory cycle time.*

In practice, most microprocessor programs require fewer than 4K words. Addressing 4K words requires 12 bits. 4 to 5 lines of the address bus are, therefore, not used for memory addressing. This feature will