MICROPROCESSOR SYSTEMS AND THEIR APPLICATION TO SIGNAL PROCESSING

C. K. Yuen

K. G. Beauchamp and
G. P. S. Robinson

MICROPROCESSOR SYSTEMS AND THEIR APPLICATION TO SIGNAL PROCESSING

C. K. Yuen

Centre of Computer Studies
University of Hong Kong

K. G. Beauchamp and G. P. S. Robinson

Computer Services Department
University of Lancaster, UK

1982
Academic Press
A Subsidiary of Harcourt Brace Jovanovich, Publishers
LONDON · NEW YORK
PARIS · SAN DIEGO · SAN FRANCISCO · SAO PAULO
SYDNEY · TOKYO · TORONTO

ACADEMIC PRESS INC. (LONDON) LTD
24/28 Oval Road,
London NW1 7DX

United States Edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue,
New York, New York 10003

British Library Cataloguing in Publication Data

Yuen, C. K.
Microprocessor systems and their application to signal processing.—(Microelectronics and signal processing)
1. Signal processing—Digital techniques
I. Title II. Beauchamp, K. G.
III. Robinson, G. P. S IV. Series
621.38'043'02854044 TK5102.5

ISBN 0-12-774950-0

Typeset by Preface Ltd, Salisbury, Wiltshire
and printed in Great Britain by
Thomson Litho, East Kilbride, Scotland

Contents

Preface

The present book is aimed primarily at scientists and engineers who wish to acquire a basic knowledge of microprocessor hardware and software, in order to apply microelectronics in the design and development of instruments and systems related to the measurement, acquisition and analysis of signals.

In the first part of the book we provide a fairly general introduction to the various aspects of microprocessor systems technology, including the electronics of digital hardware and its logical design, microprocessor instruction sets, input/output handling, and the use of software and hardware development tools. The second part illustrates the earlier material by showing the use of microprocessors and related devices in signal processing and data acquisition. It should be noted that as the material in the first part has a fairly general orientation, it should be useful as an aid to system development in other fields. We have not assumed prior knowledge of either microprocessors or signal processing applications in writing the first part. However, some familiarity with computing and programming will be helpful. In contrast, the second part assumes that the reader is already acquainted with various aspects of computer signal processing and data acquisition, such as Fourier transformation, correlation and filtering, and only wishes to learn the part microprocessor sytems may be able to play. Readers without such knowledge should consult some of the textbooks on signal processing and data acquisition listed in the reference sections of the book. While many readers may wish to apply microprocessors in areas other than signal handling, it is nevertheless suggested that the examples we include here provide very useful pointers to the potential of microprocessors and well illustrate their value as a system tool.

There are six chapters in Part 1 of the book. Chapter 1 gives a general introduction to microprocessor systems, including their hardware and software structures and a sketch of microprocessor systems development. It will be realized from this chapter that while microcomputer hardware systems are simpler than mainframe computers, many of the user facilities found in mainframes (e.g. operating systems and file management) have their coun-

terparts in microprocessor systems design and that this leads to a complexity in software development for the microprocessor user.

Chapter 2 provides a brief introduction to digital electronic devices, mainly to familiarize the reader with the common terminology and the range of components employed in hardware development. Chapter 3 is concerned with the structure of common hardware units of microprocessor systems together with some brief discussion of design techniques.

Chapter 4 introduces the reader to microprocessor instruction sets and show their use in programming simple problems. Many basic ideas of computing constructs are introduced in this chapter. Chapter 5 discusses the structure of the Input/Output subsystem of microprocessors, including interrupt handling, direct memory access, interfacing, and programming considerations. Chapter 6 presents an overview of tools commonly employed in the development and testing of microprocessor software and hardware. These play an important role in making microprocessor systems accessible to the non-expert user.

Part 2 of the book is concerned with applications. Chapters 7 and 8 consider the overall aspects of microprocessor applications in signal processing and data acquisition. Components and systems which contain, or are used with, microprocessors and which perform various signal handling functions are discussed and their important properties noted. In Chapter 9 examples are given of complete microprocessor systems used to solve particular signal processing applications.

The authors take great pleasure in acknowledging the assistance of the following individuals towards this book: Professor Arthur Sale of the University of Tasmania, Dr Garth Wolfendale of CSIRO, Canberra, Dr John Hargreaves of the University of Lancaster, Terry Kennair of the University of Newcastle upon Tyne, and to the following organisations; Intel Corp., Advanced Micro Devices Ltd., TRW Products Inc., Base Ten Systems Ltd., National Semiconductor Corp., Mostek Corp., Analog Devices BV, and Plessey Ltd.

Hongkong &
Lancaster, Spring
1982

C. K. Yuen
K. G. Beauchamp
G. P. S. Robinson

Part One

Microprocessor Systems

1 Introduction to Microprocessor Systems

1.1 INTRODUCTION TO MICROPROCESSOR APPLICATIONS

The microprocessor represents one of the more recent developments in digital electronics and integrated circuit technology. First appearing in the early seventies in the form of crude devices with limited instruction sets and interfacing capabilities, by the end of the decade the microprocessor had become a highly complex and sophisticated unit with capabilities rivalling minicomputers and even large mainframes. Microprocessors are now employed in a vast array of applications. Simple processors are found in household appliances to implement intelligent and flexible machine control, while the larger and newer processors appear increasingly in personal and business computing systems. Few of today's machines and instruments are without a microprocessor somewhere within the whole unit, and even more widespread application may be expected as microprocessors continue to become cheaper, more powerful and easier to use.

However, by itself, a microprocessor is a device of very restricted capabilities. It can do little more than accept a set of numbers (*input*) perform some arithmetic or other manipulative operations on them (*processing*), and either temporarily hold the results in its internal storage (*registers*) or *output* the results to the other parts of the microprocessor system. Moreover, it can only handle data of a very special form, namely *binary* data—numbers made up of strings of ON/OFF electrical pulses (ones and zeros). Thus, before information produced by humans or instruments may be processed by a microprocessor, it must first be converted into binary form by the *input/output* (I/O) subsystem attached to the microprocessor, and stored as strings of zeros and ones in the *memory subsystem*. Appropriate interconnection (*interfaces*) must be established between the processor, the memory and the I/O units, such that the processor may cause the I/O units to acquire the needed data and place them into suitable locations in the memory. We shall consider

various aspects of the hardware of microprocessor systems in the following section and in Chapters 2 and 3.

It may also be said that the *kind* of data manipulations a microprocessor can perform is also highly restricted—such as copying a number from one storage location to another, adding or subtracting two numbers, or testing whether a number is positive, zero or negative, or larger or smaller than another. Before a new microprocessor is produced, its designers must first decide on a list of operations the processor should be able to perform, and this list is called the processor's *instruction set*. The details of the processor's internal circuits are then worked out to make it possible to perform such operations (or in computing terms, to *execute* such instructions).

We may ask: What accounts for the power and versatility of microprocessors? The answer lies in their great speed. They can accept, process and return hundreds of thousands or even millions of items of data per second. By performing many individually simple operations in rapid succession and in a coherent fashion, a microprocessor is able to accomplish many complex tasks, even those which apparently have little to do with arithmetic or number testing.

Take, for example, the control of a washing machine. The following is a possible sequence of operations performed by the machine controller (which may be a microprocessor, a mechanical controller, or even a person operating knobs and switches) after the machine is turned on.

1. Reset all parts of machine to idle.
2. Turn on tap.
3. Test water level indicator. If "full" go on to step 4. Otherwise repeat step 3.
4. Turn off tap. Turn to "wash".
5. Wait 10 minutes.
6. Turn off wash. Turn on pump to empty.
7. Test water level indicator. If empty go on to step 8. Otherwise repeat step 7.
8. Turn to "spin".
9. Wait three minutes.
10. Turn off spin . . .

We may see that some of the operations involved are input operations, which bring data (e.g. water level) about the state of the machine into the control unit for analysis. Some are tests of data values (water full, empty, or neither full nor empty). Yet others are output operations, e.g. sending an On or Off pulse to the motor switch. Arithmetic may also be required, such as step 5, which requires the controller to keep count of elapsed time.

The above illustrates the idea of *programming*. As explained earlier, a computer processor has a number of in-built functions that it can be made to perform, which make up its instruction set. Programming is the process of specifying a sequence of instructions which would together perform a particular task. A computer program is just a set of logically related instructions. It is written in a computer *language*, which is a system of notations for defining the program, by specifying what operations are required on what data. As each individual microprocessor model has its own set of operations, it has to have its own machine language, though there is a great deal of similarity between the languages of different processors. Also, each problem may be programmed in many languages. A *high-level* or *problem-oriented* language provides a notation particularly related to a class of problems and therefore gives concise and comprehensible descriptions of their solutions. The machine language of a normal microprocessor, on the other hand, is not designed to suit particular applications, but relates directly to the hardware design. In consequence, specifying the solution of a problem in machine language instructions is usually a complex and tedious task. Even a simple task like washing machine control requires a program containing many instructions, all of which must be correctly specified to make the program work. As we shall discuss later, there are programming tools which help to reduce the difficulty of software development. In particular, it is possible to specify the solution of one's problem in a high-level language and then use a computer to translate the program into the machine language for one's microprocessor.

Regardless of the actual software development process involved, the final product is always a machine language program consisting of strings of ones and zeros, which will be accepted by the microprocessor and cause it to perform the set of operations specified by the program. Because of the large number of instructions involved, a microprocessor program cannot be stored in the microprocessor itself, but is kept in the memory subsystem, which also makes space available for data. The program will co-exist with data of various types including constants, reference tables, data received from input devices but as yet unprocessed, results returned from the processor but not yet sent to output devices, and intermediary results which will be used in later processing and then erased. Programs and the various types of data may reside in the same memory modules or separately depending on conditions (see next section).

A memory module comprises semiconductor storage "cells" (whose structure we shall study in Chapter 2), each of which can have two possible states, 0 or 1 (ON or OFF). The module also contains some control mechanism which can select a particular group of cells and reproduce the cells' contents as electrical pulses, which may then be sent to the microprocessor as

data to be worked on. This operation is called a memory READ. The same control mechanism can also receive electrical pulses and store their binary values in a designated part of the memory, and this is called a memory WRITE. Each individual cell is said to contain one *bit* of data.

During the operation of a microprocessor system, the processor *executes* a program by reading the instructions in the program from memory one by one, analysing each instruction to see what operation is to be performed on which data, and then fetches the data from the memory, manipulates them as required, and returns the new data to the memory, all as specified by the instruction. The first step is called an instruction *fetch*, the second step an instruction *decode*, and the last step the actual *execution* of the instruction. Most instruction executions involve passing data from and to the memory, some affect only data already inside the processor, while others, the I/O instructions, pass data to or from I/O devices, e.g. reading the setting of a switch or sending a string of numbers to a printing wheel. As we shall see later, special programming techniques are required for the purpose of controlling I/O devices and effecting data transfers between them and other parts of the microprocessor system. The interconnection between the various parts of the system is also a fairly complex task if proper data and control signal transmission is to be ensured.

We have identified three main components of a microprocessor system: The processor (or processors, since a system may well have multiple processing units sharing common memory and I/O devices), one or more memory modules with a control mechanism making up the memory subsystem, and the I/O devices with their interfacing hardware making up the I/O subsystem. Generally speaking, as far as hardware cost is concerned the I/O subsystem constitutes the largest expenditure. Whereas processors and memory modules handle only digital data in the form of binary electrical pulses of specific values and durations, I/O devices function to convert data between binary digital form and other forms, such as manually controlled switches, printed or other visually displayed text, analog (i.e. continuously variable) electric voltages or currents, magnetically recorded information, temperature, pressure, etc., and thus contain a wider variety of circuit elements. The present semiconductor technology allows us to implement circuits of great complexity on a small piece of silicon and produce elaborate microprocessors and memory modules at low cost.

They may be mass produced because the same microprocessor and memory components may be applied to different tasks by different programming—a factor that makes processors and memory modules cheap. In contrast, I/O devices usually contain mechanical or other non-electrical mechanisms, and there is great variation between devices in the type of information they handle, the data rates involved, and the control mechanism. This makes

component fabrication more difficult, especially as some components have to handle fairly large currents to drive the I/O mechanisms. Further, as each microprocessor system has its particular configuration of devices to suit the problem it is to solve, the I/O subsystem needs to be specially put together for it, and one is less able to take advantage of mass production methods.

However, by far the greatest individual cost item in the development of a microprocessor system is the program or software. Not only is the final running program itself a highly complex product (even for relatively simple applications like washing machine control), and therefore takes much expensive skilled labour to design, test and modify before it may be made to work, but a great many costly facilities will have to be made available to make the software development process possible. Indeed, it often requires a computer system to develop programs for a microprocessor system, as we shall discuss in Chapter 6.

We now begin to see that, whereas microprocessors are cheap, a working microprocessor *system* seldom is. We also see that the economic advantage of employing a microprocessor system for any particular application is dependent on the following factors.

1. The application should preferably be one that occurs widely in the same form or several fairly standard forms, so that a single system configuration with a standard program may be implemented in many replicated units and the cost of the hardware design and software development may be divided among many individual units, resulting in a low development cost overhead per unit.
2. The application should involve simple I/O devices of standard types and have standard data rates. The processing involved should be reasonably simple, so that it is within the capability of standard, mass produced processors executing relatively simple user-developed software together with standard system software.

While microprocessor systems can be, and indeed are, widely applied to problems which do not meet these requirements, the development cost involved is likely to be considerably higher.

The economic factors are largely responsible for the high degree of user involvement in the hardware and system software aspects of microprocessor systems. Whereas users of large computer systems usually need just to know how to write their programs, with microprocessor systems the user often has to design and implement hardware and system software himself, because it would be far too expensive to have the work done for him by others. Again, whereas large computer systems always have an elaborate system of manufacturer support for both hardware and software, it is not economical for micro-

processor producers to provide the same degree of support. Thus, not only has the development of microprocessors had a major impact on the application areas (including our domestic life, through such things as microprocessor-controlled appliances), it has also affected the computing profession, by once again requiring it to train people in all aspects of computer systems rather than just as narrow specialists. For this reason also, the present book will introduce the reader to both the hardware and the software aspects of microprocessor systems development.

1.2 THE STRUCTURE OF MICROPROCESSOR SYSTEMS: HARDWARE

In the present section we give the reader an overview of the hardware structure of microprocessor systems, discussing the basic components of these systems as well as the interconnection of these to form a system configuration. The discussion is general and centres on the "functional" aspects, rather than details of circuit design or construction. However, first we must make some remarks on current digital hardware technology.

Virtually all the electronic parts of current microprocessor systems are fabricated by *integrated circuit* (IC) techniques, in which a large number of electronic components (transistors, together with diodes, resistors, capacitors, etc.) are constructed and interconnected on a single piece of silicon. The resulting IC "chip" can have very great internal complexity, e.g. 2^{16} memory cells or a complete microprocessor contained on a piece of silicon of about one centimetre square. Although these techniques enable continually increasing circuit complexity to be contained on a single chip, they do give rise to a "switching problem". It is only possibly to connect so many wires to a small piece of silicon, the maximum at present being about 64. Thus, information to be handled by the IC devices, and results produced from them, all have to pass over these small number of connections, so that, despite the high speed and internal elaboration of the devices, there are limitations in our ability to take advantage of them in data handling. In fact, a great proportion of the circuits are included in each IC chip to direct information to and from its complex internal structure through the small number of connections. The various individual IC components of a microprocessor system have to be designed to operate together with only relative small numbers of interconnections between them.

This has several implications. First, data transfers take place in units of small size, such as 4, 8, 16, or occasionally, 32 bits. Further, it is insufficient to transfer data by themselves. There has to be accompanying information about which component as well as *where within the component* the data have to

be directed to or accessed from precisely because each component may have considerable internal complexity. Also, it is necessary to specify what action the receiving device is to perform on the data. We see that a data transfer in a computer system has to be in a "package" containing three different types of information; a *unit of data, addressing information* (where), and *control information* (what to do). And there are different levels of elaboration in the way the "packages" are conveyed. On larger systems the different parts may travel simultaneously over connections provided separately, while on smaller systems they may travel successively over the same set of wires. Obviously, the latter arrangement reduces system speed, since each data transfer operation takes several times as long. Consequently, the "bus width", i.e. the total number of separately provided electronic pulse paths, has a major effect on overall system speed. On the most sophisticated systems a large bus width is provided, but data may be passed in units of several sizes, which requires yet more control information to indicate how long the data item is.

A second implication of restricted chip connection is that data processing has to be predominantly serial even where parallel data manipulation hardware could have been provided at competitive cost. Take, for example, the addition of two vectors. Although we can fabricate IC chips capable of adding simultaneously, say, 256 pairs of numbers, few microprocessor systems take advantage of such devices because we cannot transfer 256 pairs of values simultaneously between the data stores and the parallel addition device. Instead, values have to be sent a few at a time, and the high speed potentially available cannot be utilized. The result is little better that the serial method of the processor fetching a pair of numbers at a time, adding them, and returning the sum before proceeding to fetch the next pair.

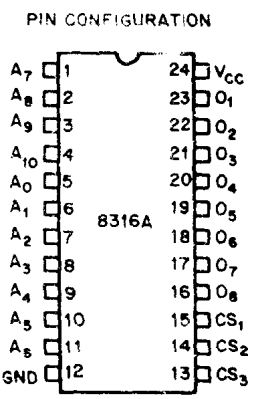
We are now ready to discuss individual system components, and commence with the memory.

1.2.1 Memory

As remarked earlier, it is now common to have IC chips containing 2^{16} individual memory cells, grouped into units of 8 or 16 bits. Each unit is termed a memory *word* or a memory *location*, and its length is the *word-length* of the memory. An 8-bit unit is given the special name of a *byte*. Processors and computers that manipulate data in 8-bit units are called byte machines. It is, of course, possible for a machine with a particular word-length to have also the capability of handling data of other sizes. A common size for memory modules is 64K or 2^{16} bits, although 256K modules are becoming readily available.

A 2^{16}-bit memory chip with word-length 8 would have 2^{13} individual memory locations. Each of these is identified by an *address*, from 0 to $2^{13} - 1$, or in

binary form, 0000000000000 to 1111111111111. Thus, the chip would have a maximum of 13 pins or connectors for addressing and 8 connectors for the data. Figure 1.1 shows the pin connections for the Intel 8316A Memory Chip. It will be seen that 3 connectors are allocated for select pulses. These are required to indicate to the memory control circuits whether one wishes to place data in a memory location or take data out of it, i.e. to carry out a READ or a WRITE operation. To WRITE into a memory location it is necessary to place the data digits on the data lines of the memory module, place the desired address of the memory location on the address lines, and then send a pulse along the WRITE control line. This will cause the data direction circuits to accept the data from the data lines and place their values into the specified memory location, erasing the values previously stored there. To READ from a memory location it is only necessary to place the address on the address lines and send a pulse along the READ control line. This will cause the data direction circuits to copy the bit values stored in the location addressed onto the data lines, ready for transmission elsewhere in the system. The READ operation does not change the values stored in a memory location, and the same contents can be read as many times as needed during various stages of processing.



PIN NAMES

A_0-A_{10}	Address inputs
O_1-O_8	Data outputs
CS_1,CS_3	Programmable chip select inputs

Fig. 1.1. Pin assignments of a 16K memory chip.