MICROPROCESSOR DATA BOOK

S. A. Money

MICROPROCESSOR DATA BOOK

S. A. Money
T Eng (CEI), MITE, MBCS

GRANADA
London Toronto Sydney New York

Granada Publishing Limited – Technical Books Division
Frogmore, St Albans, Herts AL2 2NF
and
36 Golden Square, London W1R 4AH
866 United Nations Plaza, New York, NY 10017, USA
117 York Street, Sydney, NSW 2000, Australia
100 Skyway Avenue, Rexdale, Ontario, Canada M9W 3A6
61 Beach Road, Auckland, New Zealand

British Library Cataloguing in Publication Data
Money, S. A.
Microprocessor data book.
1. Microcomputers 2. Microprocessors
I. Title
621.3819'580212 QA76.5

ISBN 0-246-11531-9

First published in Great Britain 1982
by Granada Publishing Ltd

Printed and bound in Great Britain at The Pitman Press, Bath

CONTENTS

PREFACE

Advances in the techniques for manufacturing large scale integrated (LSI) circuits have, in recent years, made it feasible to incorporate most, or in some cases all, of the complex logic required for a small digital computer system on to a single silicon chip. One example of the application of these LSI techniques is in the familiar digital pocket calculator, which is in fact a specialised digital computer. In these devices all of the electronic logic is contained in a single integrated circuit package.

In designing modern electronic systems the engineer must now take into account the ready availability of microcomputer and microprocessor devices which can simplify design, making the end product more versatile or more economical to produce.

One problem which faces the designer planning to use a microprocessor is the great multiplicity of devices that have become available. Choosing a suitable microprocessor could involve collecting together and searching through a mountain of different data sheets and manuals.

In this book condensed data have been provided for most of the available types of microprocessor and microcomputer device. For each major type or series a description is given of the internal architecture, instruction set, main electrical data and package details.

Most of the popular devices are manufactured by several different suppliers, and a list of alternative sources and type numbers have been included in the data for each type. Support chips designed for that processor have also been listed.

For convenience the devices have been divided into groups covering 4, 8 and 16-bit types and other processors. It would not be practical to include full details of each type, but it is hoped that sufficient information has been provided to allow the designer to narrow down his choice to perhaps one or two types. The manufacturer's data sheets or manuals may then be consulted for more detailed operating and application information.

In order to choose a processor for a project some knowledge of the basic principles of the devices is required, and this has been covered in the introductory chapter. A general guide has also been included on the factors involved when a processor type is chosen.

A complete system normally consists of a microprocessor together with a selection of supporting devices to handle input–output, external device control and to provide memory. The number of support devices available is even greater than that of microprocessor types, so no attempt has been made to include details of all of these. Some descriptions have been included covering the major support device functions, and data have been included on some of the more popular types as a guide to the facilities provided by such devices.

At the end of the book a directory of microprocessor manufacturers has been included and there is also a glossary of some of the terminology used in the microprocessor field.

It is hoped that the information given in this book will assist designers in choosing suitable devices and that it will be generally useful to those engaged in designing or planning microprocessor based products.

One problem encountered in producing any data book which deals with a rapidly advancing field, such as microprocessors, is that new devices are continually being introduced. To deal with this situation plans are being made for the publication, from time to time, of a supplement giving data on recently introduced devices. Readers wishing to have details of these supplements should complete and mail the coupon enclosed in this book, or alternatively write to the publishers, when they will automatically receive advance details of these supplements.

The reader will notice that for a limited number of devices in this book only limited data are given. This is because at the time of compilation only preliminary information was available. The reader is referred to the supplement for full information.

Finally, I would like to express my thanks to all those manufacturers and distributors who supplied the data and other information which made it possible to compile this book.

1 INTRODUCTION

In recent years the advent of microprocessors and microcomputers has revolutionised the whole process of digital system design. Projects which, a few years ago, might have required tens or hundreds of digital logic devices can today be implemented by using perhaps one or two LSI circuits. Of course, LSI circuits have been around for some years, but economic considerations have usually limited their use to applications, such as digital calculators, where high volume production is possible and high design costs can be recovered quickly. The advantage of the microcomputer is that a standard device can be used for many applications by merely altering the program of instructions held in its memory. Thus design costs can be reduced and a variety of products may be built using perhaps a standard circuit board.

Microcomputers, however, bring with them a number of new design concepts which may be unfamiliar to the system designer used to working with conventional digital logic systems. In this introductory section we shall examine the internal organisation of microcomputer systems and their general principles of operation. Later we shall consider the various factors that are involved in choosing a suitable type of microprocessor for a design project.

ARCHITECTURE

The general organisation or architecture of a digital computer, whether it be a mainframe, a minicomputer or a microcomputer, follows the basic arrangement shown in fig. 1.1.

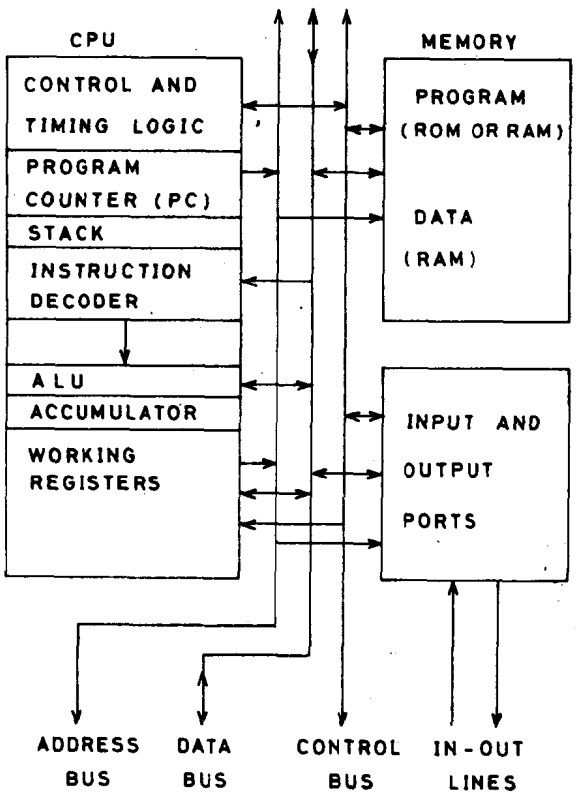

Fig. 1.1

At the heart of the system is the *central processor unit*. generally referred to as the CPU. Functionally the CPU can be broken down into two subsections, one of which is used to control the timing and sequence of operations in the system, whilst the other executes the required arithmetic and logical functions and handles the data being processed. A memory system is connected to the CPU and is used to store the list of instructions to be executed, known as the *program*, and the data being processed. In some types the data and program memories are separated but for most of the general purpose microprocessors a common memory is used. Communication with the outside world is handled by two further sections, known as the *input* and *output* ports, which allow data to be transferred to and from external devices such as keyboards, display units and printers. The various components of the microcomputer system are tied together by a system of bus lines which are common to all units. This is of course a very much simplified description of a microcomputer system and we shall now go on to look at each section in more detail.

BUS SYSTEMS

Data is transferred between the various units of the system over sets of parallel wires known as *buses*. Normally there are three buses, one for data, one for addresses and the third for control signals.

The data bus allows data to be transferred between the CPU and the memory or input–output lines. This bus is in fact bidirectional and controlled by the CPU. A read–write line, which forms part of the control bus, determines the direction of data flow along the data bus. This signal is generated by the CPU and if set to *write* allows data to be output from the CPU to other parts of the system. When the control line is set to *read* the CPU accepts data which have been placed on the data bus by one of the other units in the system. The bus may only be driven by one device at a time, although all of the other devices may read data from the bus simultaneously.

The address bus is used to provide an address signal for the memory to select one particular location within the memory for connection to the data bus. Normally the address bus is an output from the CPU and data always travels from the CPU to the memory. The address bus may also be used to select individual input or output channels when there are several of these connected to the data bus. Some of the smaller processors may have separate address lines for program memory and data memory.

The control bus provides a selection of control signals to and from the CPU to govern timing and control of transfers of data on the other bus lines. Among these may be signals for halting the operation of the CPU and perhaps disconnecting it from the bus system. The address and data buses are normally tri-state in operation. Apart from the normal 0 and 1 states the CPU, or memory, may be switched into a high impedance state which effectively disconnects it from the bus. This facility may be used when it is desired to have another device take over control of the bus system. Typical applications are in multiprocessor systems where a bus is shared between two or more CPUs, only one of which may access the bus at a time.

In a large system the output drivers of the CPU and other devices may not be capable of driving all of the loads on the bus. In such cases bus drivers or bus trans-

ceivers would be used where each device connects to the bus system. In some such systems the data signals on the data bus may be inverted by the transceivers or buffers.

CPU CONTROL

Apart from system timing and control logic the control part of the CPU normally contains a register called the *program counter*, an address register, an instruction decoder, a stack or stack pointer register and some interrupt logic.

Instructions making up the program to be executed are stored in the memory and are normally called up in sequence for execution. The function of the program counter is to give the memory address of the next instruction to be executed. At the start of a program the program counter will be loaded with the memory address of the first instruction. When the system starts running this address will be transferred to the address bus and the instruction code will be read in from memory. One part of the instruction is called the operation code or *opcode* and determines the type of operation to be performed. Typical operations might be ADD, SUBTRACT or AND, each of which will have a different opcode pattern. The opcode data read in from memory are passed to the instruction decoder, which then sets up all of the logic linkages required within the CPU to perform the desired operation. At this point the contents of the program counter will be updated to give the address of the next opcode in the memory. Some operations will require data, as well as an instruction code, and for these one or two additional words of data may follow the opcode in the memory. These data words will be dealt with as the instruction is executed. When such data exist the contents of the program counter may be incremented by two or three to provide the correct address for the next opcode. This updating is governed by the instruction decoder according to the type of opcode detected.

A typical instruction execution sequence consists of one or more instruction cycles. The first cycle normally calls in the opcode from memory and is called the *instruction fetch cycle*. On simple instructions this will be followed by an execution cycle, when the required operation is carried out. If data are required extra cycles are required to read in the data words following the opcode. In some cases these words may be a memory address, and once in the CPU will be transferred to the address register and address bus during the execution cycle.

Timing varies from one type of processor to another. In some types, such as the 6800 and 6502 series the timing is relatively simple with each instruction cycle divided into a pair of time periods called $\phi 1$ and $\phi 2$. Internal CPU functions occur during $\phi 1$, such as execution, and memory access occurs during $\phi 2$. Thus a simple instruction calls in the opcode on $\phi 2$ and executes the instruction during the following $\phi 1$ cycle. Other processors may divide up the instruction cycle into many different time periods, which may vary according to the type of operation being performed. Manufacturer's data sheets should be consulted for detailed instruction timing for any given processor type.

We shall look at the stack and interrupt functions in more detail later. The timing clock for the CPU will often be a multiphase type, but in many of the newer types of device it will be generated by an on-chip oscillator which merely needs an external quartz crystal for timing. Older types often do require multiphase clocks but usually a special clock chip is available to generate these, using a crystal for timing. In some cases the system clock will also be needed to control the timing of the memory and input–output channels.

CPU EXECUTION

In the execution section of a microcomputer CPU there will be an arithmetic and logic unit or *ALU*, which is a complex logic array to carry out the desired arithmetic or logic function. Associated with the ALU is a special register called the *accumulator*. Normally the ALU has two data inputs and one output which may each be 4, 8 or 16 bits wide. Data for one of the ALU inputs are provided by the accumulator and the results of the operation are placed in the accumulator after execution.

Apart from the ALU and the accumulator the execution unit may also contain a number of general storage registers, which may be used to hold data or intermediate results. In some types of processor these additional general purpose registers may be used as accumulators. Data may be transferred to and from the accumulator, ALU input and general registers and memory via the data bus.

Using the internal registers for data storage can speed up the operation of the program, since these registers can be specified by the opcode, thus avoiding the need for extra memory access cycles when the instruction is executed. Some CPUs, such as the 6800 and 6500 series, use memory locations as general purpose registers allowing operations such as shifts and incrementing or complementing of data directly within the memory.

Many CPUs have data pointer or index registers within the register bank. These registers may be used to hold memory addresses rather than data and for certain instructions their contents may be placed on the address bus to select data in the memory. Such registers may be incremented or decremented to allow tables of data to be handled in the memory.

Typical functions provided by the ALU and accumulator are ADD, SUBTRACT, AND, OR, EXCLUSIVE OR, complement and clear. It is also possible to shift data left or right by one bit in the registers and to rotate data where the bit pushed out at one end of the register may be inserted again at the other end to give a continuous loop of data moving through the register. Other operations include incrementing and decrementing contents of registers. Normally arithmetic uses pure binary numbers, but for some applications binary coded decimal (BCD) format may be used where groups of four binary bits are used to represent decimal digits.

On a few more sophisticated processors multiplication and division instructions may be provided, but for most types these operations must be implemented by a small sequence of instructions. Many processors with a multiply or divide will use an internally stored program to carry out the operation which will be relatively slow to execute. Some types, however, use a hardware logic array within the chip to give very fast multiplication execution times.

A very important facility in microprocessors is the

ability of the CPU to test the results of an operation and to take alternative courses of action depending upon those results. Typical tests provided are zero, minus, carry and overflow. Carry indicates that a carry out of the accumulator was generated by the operation and overflow indicates that the result is a number larger than the register can handle. As a result of these tests the CPU may be made to skip, branch or jump instead of executing the next instruction.

A skip operation causes the CPU to skip over the next instruction in sequence and execute the following one. In a branch or jump instruction a new value is placed in the program counter and execution is transferred to another point in the program. For jumps the new instruction address follows the jump instruction as data in memory. Branch instructions often use relative addressing, where the data following the branch instruction are added to the current contents of the program counter to generate the new instruction address. In a conditional branch the program will branch to a new point if the condition of the branch is true, whilst execution will continue with the next instruction if the condition is not true. When an unconditional branch or jump occurs the program will always go to the new point specified by the branch or jump.

Typical branch conditions are Branch if Zero, Branch if Not Zero, Branch if Plus, Branch if Minus and so on. Results of the various tests are normally stored as flag bits in a status register. Often these flags can be manipulated by the program so that if desired the carry bit could be set or reset directly by the program.

SUBROUTINES AND STACKS

Often there will be frequently repeated sequences of instructions in a program. Typical of these might be the set of instructions to read data from a keyboard or to carry out a multiplication. Whilst these instructions could simply be repeated as desired throughout the program a more convenient technique is to make use of a subroutine. In a subroutine the small sequence of instructions is treated as a small separate sub-program and is stored once in the memory. When the subroutine is to be used a special jump to subroutine or CALL instruction is used. This causes the contents of the program counter to be temporarily stored away and then a jump is made to the start of the subroutine instructions. At the end of the subroutine a RETURN instruction is executed which causes the old program counter contents to be restored and the program will then continue to execute with the instruction immediately after the CALL.

In simple processors a single temporary register is used for saving the program counter, but with such a scheme it will not be possible to call a subroutine from within another subroutine. Some processors may have several save registers, which are called the *stack*, and as each register is filled a new one is selected for loading. In such a system the last register loaded will be the first to be read out, and this type of memory is sometimes called a first in last out or FILO type. In computers this type of memory arrangement is called a stack and in most of the general purpose processors the stack is built up in the main memory. To keep track of the current stack top location a data pointer register known as the *stack pointer* is used and this is normally part of the CPU control section. On each subroutine call the contents of the program counter are written to the top of the stack in memory and the stack pointer is automatically updated to point to the next free location in the stack. On a return from subroutine the stack pointer is moved down to point to the last data written to the stack and then those data are transferred to the program counter. Most general purpose processors will allow data to be moved to and from the stack by using PUSH and PULL (or POP) instructions with the stack pointer being updated as required. Some of the more advanced processors have two stack pointers, one for storage of subroutine addresses and generally called the *system* or *supervisor stack*, whilst the other may be used for data storage and is called the *user stack*.

When a stack is implemented in memory subroutines may call other subroutines, a process known as *nesting*. As the inner subroutines complete execution the program transfers to the next subroutine up until eventually a return is made to the main program. When nesting is used the size of the stack space available will determine the number of levels to which subroutines may be nested. In some processors the return from a subroutine may be made conditional so that returns may be made from different points in a subroutine, according to the results of operations carried out within the subroutine.

Sometimes it may be desirable to transfer data from the main program into a subroutine. This can be done by pushing the data on to the stack before the subroutine call. Once inside the subroutine the stack pointer may be manipulated to allow access to the data and perhaps to replace them with data to be transferred back to the main program. The only critical point here is that the stack pointer must be restored to its proper position pointing to the return address before the return from subroutine is executed.

INTERRUPTS

When communicating with the outside world there will often be occasions where the processor is ready to transfer data but the external device isn't or *vice versa*. These situations can be overcome by placing the processor in a testing loop, where it waits for a ready signal from the external device. This is inefficient, however, since the processor is not processing data whilst it is waiting for the external device. A technique for overcoming this problem is to use an interrupt system. In an interrupt system the external device applies a pulse to a special interrupt request input on the CPU, and this sets a flip flop within the CPU. When an interrupt occurs the CPU will complete the execution of the current instruction and then branch to an interrupt service routine which will deal with the external device. The service routine is in fact very much like a subroutine. When the interrupt is acted upon the contents of the program counter are stored as for a subroutine, but usually the status register contents will also be saved on the stack as well. Some processors, such as the Motorola 6800, will save the contents of all of the internal registers when an interrupt occurs. As with a subroutine at the end of the interrupt service routine the program counter and any other saved registers will have their contents restored and program

execution continues with the next instruction in the program.

The simple interrupt described above is usually called a *non-masked interrupt* or NMI. For some purposes it is useful to be able to inhibit interrupts and this can be done by using a maskable interrupt. By setting or resetting a bit in the status register the interrupt may be inhibited or activated as desired. Interrupts may be nested as for subroutines if desired.

A further type of interrupt is the software interrupt, which is triggered by an instruction rather than by a pulse from external hardware. This type of interrupt is generally used for debugging programs, but may also be used to indicate a fault condition thrown up by a program test.

Most processors have only one interrupt request (IRQ) line for masked interrupts and one for non-masked interrupts. Normally the non-masked and software interrupts will have priority over masked interrupts and are thus dealt with first if both types occur together. There may be several external devices sharing the common interrupt line, and some form of priority of service may be desired between the various devices. In such cases priority will usually be determined by external logic, but some processors such as the Texas 9900 series do allow for priority decoding within the CPU.

Most general purpose processors use what are known as *vectored interrupts*. Here each type of interrupt will cause the CPU to go to a different address which will contain the interrupt vector which is the start address for the service routine corresponding to that particular interrupt. In some types the vector address of the service routine must be fed in via the data bus from the external interrupting device when the CPU acknowledges acceptance of the interrupt.

Power on reset or initialisation sequence is a special form of interrupt. Normally this causes the program counter to be loaded from either the top location in the memory or from the bottom location. The CPU then executes a start up routine which resets the internal logic and starts execution of the program at the reset vector address stored at the top or bottom of memory.

MEMORY

The microcomputer memory is used to store both program and data. Characteristics of various types of memory device are discussed in Section 8. Normally the program will be held in some form of read only memory for a dedicated system, whilst the data are held in read–write memory, often referred to as RAM. Some data, such as constants, may be stored in ROM along with the program. In a general purpose computer system, such as a development system or a personal computer, the user program will often be stored in RAM but the operating system and its associated programs are often stored in ROM. In most systems an initialisation program and various utility routines may be stored in ROM to form what is generally called a *monitor program*.

For large amounts of data storage special memories, such as magnetic bubble types, may be used or the data may be held on a floppy disk or a magnetic cassette tape. Microcomputers generally have a small amount of scratchpad RAM for data and a medium size ROM for program storage. Most single-chip types use mask programmed ROMs, which makes them best suited to very large production levels. Some types may use programmable ROMs for the program storage and a few have erasable PROMs. These types are best suited to specials and limited production runs.

INPUT–OUTPUT

To be of any use the microcomputer must be able to communicate with the outside world. Often this may be via a keyboard or keypad for input and visual display or printer for output. Digital data in and out are transferred via data ports, which are basically latched registers connected to the data bus and selectable by the CPU for data transfers. In some cases the signals may be converted to or from analogue signals before reaching the outside world. Switches, solenoids and servos can be driven from the CPU via suitable interface circuits.

Apart from the data lines each input or output port will have a pair of control lines, often referred to as *handshake lines*. One of these lines is an output from the computer to indicate to the external device that it is ready to transfer data via the port. The second line is an input to the computer system from the external device and may be used to indicate either that the peripheral unit is ready to accept data or that it has placed data on the port lines for the computer to read. These signals may also be looked upon as a request for action in one direction and an acknowledgement signal in the other direction, or alternatively they may simply be used to indicate that the device producing an output handshake signal is busy. Handshake signals are particularly important where the microprocessor and the external device operate at different speeds.

WORD LENGTH

Microprocessors and microcomputers work with binary data, which are handled as groups of binary digits or *bits* and these groups are called data words. Typically a word may contain 4, 8, 12, 16 or 32 bits of data according to the type of system being considered.

The earlier microprocessors, such as the Intel 4004 or 4040, and many of the modern single-chip microcomputers use 4-bit data words, which are sometimes called *nibbles*. In these systems the data bus, ALU and data registers are all 4-bits wide. The numerical value of a 4-bit word can range from 0 to 15, and often a system known as *binary coded decimal* (BCD) may be used where each word is used to represent a decimal digit having a value from 0 to 9. Large binary numbers have to be processed in 4-bit segments.

Most of the popular general purpose microprocessors use an 8-bit data word, which is referred to as a *byte*. A byte may have a numerical value in the range 0 to 255 and is also convenient as a means of encoding text character information, where perhaps 96 different upper and lower case letters and various punctuation signs and numbers have to be identified. In an 8-bit system the data paths, ALU and registers are normally 8 bits wide. A data byte may be used to hold two BCD digits when that type of data is to be handled.

For large systems the word length and data path size

may be 16 bits, allowing data values from 0 to 65535. Some of the 16-bit processors may include registers 32 bits long, so that 32-bit wide data can be handled inside the CPU, but at the time of writing no 32-bit systems are available although some minicomputers can handle 32-bit wide data.

In most cases the data bus will define the word size of the processor system, but there are a few processors, such as the Intel 8088, which although having a 16-bit CPU architecture, use an 8-bit data bus and each data word is transferred as a pair of successive data bytes.

With a 4-bit processor the data word is not wide enough for use as an instruction opcode since it would allow only 15 different instructions to be used. It is normal therefore for 4-bit processors to use an 8-bit instruction word and to hold the program instructions in a separate 8-bit wide memory. A few types may use a 10-bit instruction word.

In industrial systems it is common to find 12-bit data, giving a range of values from 0 to 4095. A few microprocessor types have been produced for this word length but generally 12-bit data will be handled by an 8-bit or 16-bit CPU system.

Address systems for the 8-bit processors are usually 16 bits wide, allowing up to 65536 memory locations to be selected. For the larger 16-bit processors an address of 20 or 24 bits may be used which will allow several megabytes of memory to be directly accessed.

Sometimes in order to reduce the number of leads needed on the processor package the data and address buses may be multiplexed over a common set of lead out pins. Although this allows the use of a smaller package and may produce a less expensive CPU device, it can lead to extra complication in the external logic circuitry, which may offset the advantages gained by producing a smaller CPU.

TYPES OF DEVICE

Three basic types of device are available for use in the construction of a microcomputer system.

Firstly there are the microprocessors which contain all, or most, of the CPU section on a single silicon chip. Some types such as the 8080 do require additional external logic to provide a system clock and to control the data and address buses but the newer microprocessor types include these functions on the single chip.

For small dedicated systems it is possible to obtain a single-chip microcomputer which has the CPU, memory and input–output ports on a single chip. Such a device can provide a complete microcomputer system with just a few external discrete components for timing. Most of these devices use mask programming for the program memory where the pattern of instructions is built into the chip as it is being manufactured. A few types do have programmable read only memories for the computer program which can be set up in the field.

The third major type of processor device is the *bit slice*, where each of the CPU functions is implemented as a slice of logic typically four bits wide. Normally there will be an ALU slice and some form of program sequencer or control slice to make up the main functions of the CPU. Other slices may be used for timing, stack functions and data control. If an 8-bit wide system is needed then two 4-bit ALU slices would be operated in parallel and the program control slices may be dealt with in a similar way. These devices are generally very fast in operation and would be used for special purpose processors where high speed or some special internal organisation is desired.

FABRICATION TECHNOLOGY

Devices may be built using a wide variety of semiconductor fabrication technologies but generally these can be grouped as four major types of fabrication.

Most of the early microprocessors used *p*MOS, where the logic is implemented using *p* channel field effect transistors (FETs). These early devices normally operate from a negative voltage supply and will often have several different voltage supply rails. Their main advantage is that *p*MOS is an older and hence more established form of construction and generally devices using it may be cheaper to produce than other types.

The popular modern processors use *n*MOS technology, where the internal logic uses *n* channel FETs and these devices will normally be designed to operate from a single +5 V supply rail. *n*MOS devices are usually faster than *p*MOS versions, but they do require well stabilised supplies of typically ±5% tolerance. Because of high production volume their cost may not be a lot different from that of similar *p*MOS types.

The third major type of device is made using CMOS, where a combination of both *n* and *p* type FETs is used. Perhaps the most important feature of CMOS is its very low power consumption compared with other types of fabrication. This makes the CMOS microprocessors ideally suited to applications where battery operation and hence low power demand are needed. CMOS also has the advantage of being relatively unaffected by variations in the power supply voltage and is less sensitive to noise on its input lines. A slight disadvantage of CMOS is that it tends to be a little slower in operation than *p*MOS or *n*MOS, although some new types of CMOS device built on a sapphire substrate can achieve speeds as high as for *n*MOS systems.

Finally there are the processors that use bipolar devices for the logic. These may use *transistor transistor logic* (TTL), *emitter coupled logic* (ECL) or *integrated injection logic* (I^2L) for the actual logic circuits inside the chip. The main feature of bipolar devices is that they are very fast in operation and will normally be chosen for special functions where high speed is essential. Bipolar circuits generally use up more space on the silicon chip, so these types tend to be less complex than the types using MOS technology and bipolar devices also tend to require more power.

One factor to be considered is that all of the MOS types, *p*MOS, *n*MOS and CMOS, are susceptible to damage from static electric charges at their very high impedance inputs. Most of the devices, however, include diode protection at the inputs to reduce the risk of damage by static charges, but nevertheless some care may be required when handling this type of device.

CHOOSING A MICROPROCESSOR OR MICROCOMPUTER

Unfortunately for the system designer there is no convenient magic formula by which the optimum micro-

processor or microcomputer device can be selected for a particular application. It is, of course, fairly easy to select a device which may be technically best suited for a particular project but often it will be aspects of software design or economic considerations which will dictate the type of device used.

In the following notes the general procedure involved in choosing a suitable processor for a project will be discussed and the factors which may influence the decisions at each of the stages will be considered. Although these guidelines will, it is hoped, prove useful the final decisions will usually depend very much upon the particular circumstances existing when the choice is made. In many cases the decision may well require the use of plain commonsense judgements based upon the available facts.

Basically the process of choosing a suitable processor or system can be broken down into the following stages:

(1) Define exactly what the system must do.
(2) Choose whether a microprocessor, single-chip microcomputer or bit slice system is to be used.
(3) Choose the word length.
(4) Consider the hardware factors such as speed, power requirements and availability of existing hardware.
(5) Consider software design with particular reference to available expertise and development aids.
(6) Examine the economic factors.

At each of these stages it should be possible to eliminate a number of the available types of processor until there are perhaps two or three potential devices from which the final selection can be made.

The first stage may seem very obvious and yet it is surprising how many system designers will progress to detailed design before they have defined exactly what is required. At this stage the system specification can be divided into two broad areas. First there are those requirements which are absolutely essential, and secondly there are the features which, whilst not essential, may be desirable since they may make a more versatile or attractive product. These features may be arranged in a list with some form of priority or value rating given to each. These secondary features of the system requirements may be helpful in the later stages of selection where two more or less equally attractive devices are being considered.

Once the system requirements have been defined it is important to ask the question 'Is a microprocessor needed at all?'. Consider what is involved in meeting the system specification using conventional logic, programmed arrays or off the shelf dedicated circuits. In many cases these other approaches will be impractical and a microprocessor type system is inevitable. Where the other approaches might be practical alternatives they should be considered along with the microprocessor solution. It would be ridiculous to use a microprocessor system when simple logic could provide a cheaper or simpler solution.

Choice between bit slice, microcomputer and processor will usually be dictated by the technical requirements of the system. In general the bit slice approach would be used where very high speed or some special processing function is needed. Choice between a single-chip microcomputer device and a multi-chip system based around a microprocessor is likely to prove a little more difficult.

Generally a single-chip microcomputer device would be chosen where space is limited or where a dedicated system is being produced for which the level of production is likely to be very high. Most of the single-chip microcomputers use mask programming and production runs of at least some 5000–10000 units will probably be required in order to justify the cost of designing the mask. In mass production, however, the cost per unit of a single-chip design is likely to be very much less than that of a multi-chip arrangement. An important factor here is that assembly and circuit board costs will be greatly reduced and these can contribute a large proportion of the final production cost of each unit. Another factor is that a single-chip design with fewer lead connections will be potentially more reliable than a multi-chip system.

For prototype and small batch production runs the mask programmed microcomputer becomes uneconomical, but devices such as the Intel 8748 or Motorola MC68705, with a programmable on-chip ROM, may be used. The unit cost of such devices can be fairly high and some labour will be involved in programming the on-chip ROM for each individual unit. However, a system using such devices does have many of the advantages of the single-chip approach, such as lower assembly cost, smaller size and potentially better reliability.

When the system is required to perform a variety of tasks or the production quantity is low, perhaps one off, a multi-chip system using a microprocessor with external I/O and memory devices may be used. The cost of the individual devices will generally be low since the microprocessor itself is likely to be a standard popular type, such as the 8080 or 6800, which is being mass produced. The assembly costs for such a system are likely to be relatively high and the complex interconnections on the circuit board may lead to potential unreliability.

The word length is usually fairly easy to resolve from the system technical requirements, although at this stage it may be permissible to consider alternative word lengths and leave the final elimination to a later stage.

Generally the 4-bit word length is appropriate for the smaller systems, particularly where binary coded decimal data is being handled. This is particularly true for controller type devices where the input is applied from a keypad and the output data are displayed on some form of decimal display.

Where text data are being handled and for general number processing the 8-bit word length is highly suitable. The normal set of text symbols can readily be accommodated within an 8-bit binary code. Most of the personal computer systems and small business machines use the 8-bit word length.

For large processing tasks the 16-bit word length may be desirable. In general a 16-bit machine will be faster in operation than an equivalent 8-bit type since instructions will generally be contained in a single data word and fewer memory access cycles will be required for each instruction. In a 16-bit system alphanumeric characters may be packed two to a word, again saving memory space and memory accesses. In general 16-bit types may well be used to perform the tasks that hitherto would have used a small minicomputer system.

A study of the hardware requirements, such as speed, power demand and compatibility with other systems, will often resolve the choice of fabrication technology. For battery operated and portable equipment CMOS type devices will usually be chosen, since they have very

low power requirements and will tolerate wide variations in the supply voltage. For most of the other applications the *n*MOS type of device will probably be the best choice, unless the system must be interfaced with existing *p*MOS type equipment. Generally the choice of bipolar technology, such as I^2L and ECL or TTL, will be reserved for applications where very high speed operation is required.

When considering the hardware aspects, especially for multi-chip systems, the availability of ready made circuit boards should be taken into account, since their use could greatly simplify hardware design and reduce design costs. In most cases these boards may contain some redundant circuits not needed for the current application, but they are usually less expensive than specially designed boards when only limited production is envisaged.

Finally we come to the software, or computer program, which can represent a major part of the cost and design effort needed for developing a microprocessor based product. Unlike conventional logic systems the microprocessor will do nothing without software.

An important factor to be considered is the availability of in house software expertise. It may well be better to use a technically less attractive processor with which the in house software team are familiar, than go to the trouble and expense of retraining the team in the software required for a newer type of processor. In some cases, however, training of the software team may be justified if the new processor is likely to be widely used in future projects. The final decision in this matter will depend upon the possible long term future benefits that might be gained by using a new processor type.

Both the software and hardware development will require some form of development aids. For small projects and as an initial training aid the simple evaluation type board is a useful development aid. However for any serious work such a system is likely to be inadequate and a full scale system with editor, assembler and debug facilities is essential. A disk based operating system is desirable since this allows rapid operation and an efficient filing system. Some systems, such as the Motorola 6800 PDS, use a cassette filing system but it is very much slower in operation than a disk based system. Most disk systems use either standard or mini floppy disks giving perhaps 100 to 200 kbyte of storage per disk drive. It is advisable to have two disk drives since this makes copying of files much faster and simpler. One disk may be used for the operating system whilst the second is used for working files.

Debug facilities normally allow the program to be run and breakpoints to be inserted as desired. At these points the program may be stopped and data and status within the system may be examined. Also the program may be traced one instruction at a time to see exactly what is happening during its operation. These facilities allow errors in the program logic or coding to be detected and corrected.

For hardware testing an in circuit emulator may be used. Here a special probe is inserted into the processor socket on the hardware board being tested. The emulator allows the main development system to provide the processor function in the final product board and the full debug facilities may then be used to discover the cause of system failures.

In many cases the development system will be provided by the microprocessor manufacturer and will usually handle only processors made by that manufacturer. Sometimes different cards will need to be used in the system according to the actual type of processor being used.

Some development systems, such as the Futuredata AMDS produced by General Radio and the Tektronix 8000 series microprocessor lab system, are designed to handle a wide range of different manufacturers' devices. An advantage of this type of system is that the basic operating system will be the same for all processors, whereas with other systems each type has its own distinctive operating system. These multiprocessor systems generally use different processor boards for each processor type when emulation is required. These systems may also provide cross assemblers for a wide range of device types whilst using one control processor in the development system. This allows machine code programs to be produced for a range of device types without the need for board changes.

Some of the more popular processor types may have their software developed on larger computers such as the PDP11 minicomputer by using cross assembler programs.

The choice of microprocessor for a project is likely to be very much influenced by the availability of some suitable development system in house. The purchase of a development system can be a major item of expenditure and must be taken into account when choosing a processor. Here the future use of the system will be an important factor in deciding whether to use a particular processor. For a single project it may well be economical to rent a development system specifically for the project.

The cost of producing the software and the availability of a suitable development system will generally resolve the final choice of processor to be used for a project. The final decision may well be made on political or economic grounds rather than on purely technical factors. Obviously the aim should always be to choose the best technical device that can be justified.

It is hoped that these notes, used in combination with the data on microprocessor devices given later in the book, will assist readers in choosing suitable processor devices. In the final analysis, however, the decision made will often depend upon the application of common sense, past experience and perhaps a little intuitive judgement.

MICROPROCESSOR PRICES

One factor which may influence the choice of processor for a project is the unit price. No attempt will be made to give actual prices for the various types, since these are likely to vary from time to time according to the popularity of the device and the level of production. Price will also change according to the quantity purchased. However the price of one type relative to another may influence the final choice, so a guide has been provided where the prices are based upon those current in the UK at the time of preparing the book but are presented as units to a normalised scale. Although this will perhaps give some guidance to possible price relativities it is essential that the manufacturer or supplier be consulted to get realistic prices before the final choice is made if the price is likely to be a deciding

factor. Generally the price will not be important until the choice has been narrowed to perhaps two or three types of device.

Relative price data

4-bit processors

Generally the 4-bit devices are microcomputer chips which are mask programmed and likely to be used in very large quantities. Final prices will depend upon negotiations with the actual chip manufacturer and the number purchased.

AMD 2900 series	8 – 20
AMI 2000 series	2 – 5
National COPS2 series	1 – 2.5
N.E.C. μCOM42 to μCOM45 series	1.5 – 3
Rockwell PPS4/1	2 – 5
Texas TMS1000	1 – 4 according to type

8-bit processors

Apart from the single-chip 8-bit microcomputers most of the 8-bit types are likely to be used in small to medium quantity and prices are based around the 100+ level.

Intel 8080A	8 – 12
Intel 8085A	9 – 14
Motorola 6800	8 – 12
Motorola 6802	10 – 15
Motorola 6809	25 – 40
MOS Tech 6502	10 – 15
RCA 1802P	10 – 15
National 8060(SCMP2)	12 – 18
Zilog Z80	8 – 12
Signetics 2650A	15 – 20
NEC μPD780C	9 – 15
Fairchild F8 CPU	10 – 15
Intel 8748	about 30

16-bit and other processors

Intel 8086	100 – 120
Motorola 68000	about 300 – 400 at present
Intersil 6100	12 – 15
Texas 9900	70 – 100
Zilog Z8000 series	150 – 200
Fairchild 9440	180 – 250
Intel 2920	200 – 250

2 4-BIT MICROPROCESSORS AND MICROCOMPUTERS