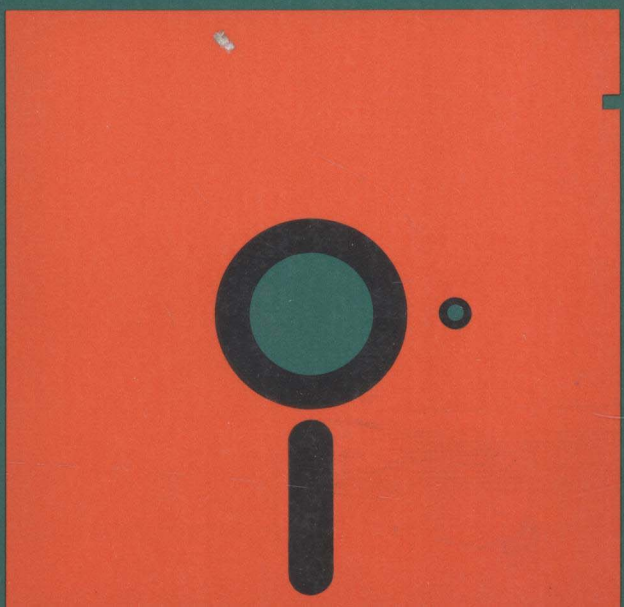
Microprocessors and microcomputers

for engineering students and technicians

BARRY G. WOOLLARD



6
5

Microprocessors and microcomputers for engineering students and technicians

Barry G. Woollard C Eng, MIERE, M Inst MC

Senior Lecturer, Walsall College of Technology

McGraw-Hill Book Company (UK) Limited

London · New York · St Louis · San Francisco · Auckland · Bogotá
Guatemala · Hamburg · Johannesburg · Lisbon · Madrid · Mexico
Montreal · New Delhi · Panama · Paris · San Juan · São Paulo
Singapore · Sydney · Tokyo · Toronto

Published by
McGRAW-HILL Book Company (UK) Limited
MAIDENHEAD · BERKSHIRE · ENGLAND

British Library Cataloguing in Publication Data

Woollard, Barry
Microprocessors and microcomputers for engineering students and technicians.
1. Microprocessors
I. Title
621.3819'58'35 TK7895.M5 80-41033

ISBN 0-07-084640-5

12345 AP 84321

Printed and bound in Great Britain at the Alden Press, Oxford.

Preface

Recent years have seen considerable excitement concerning the 'silicon chip', so much so that it has become a household term. It is unfortunate that with all the publicity there is grave danger of the *silicon chip* being taken out of context. This so-called 'silicon chip' has been with us for about a quarter of a century and, as we know it today, is a technological development of the transistor—which, after more than a decade of using other methods, was first manufactured in a slice (or wafer) of silicon in 1961. Several hundred similar devices were processed simultaneously—each device being a *chip* from the slice. About the same time, it was realised that other electronic components could be manufactured simultaneously into the 'chip' to produce an *integrated* circuit (IC). During the ensuing years, the technology and manufacturing equipment was progressively developed to the extent that more complex mathematical (and electronic) functions could be contained into a single IC (or 'chip'). By 1971, *Intel* had produced the first 8-bit *microprocessor* using similar techniques. Improvements have continued to be made to these devices and other IC's which may be used with them, e.g., storage elements (memories) and interface circuits, thus enabling *microcomputers* to be assembled and put to a wide variety of applications. It is to this latter area that the term *silicon chip* has really been aimed—mainly because we now have the hardware to enable large-scale automation to be an economically viable proposition. A descriptive phrase recently stated that the microprocessor 'is a solution in search of a problem!'.

Many people talk about the *microprocessor*—or indeed, the *silicon chip*, when in fact, they really mean the *microcomputer*. Alone, the microprocessor (μP) is a virtually useless piece of electronic hardware! To enable the μP to be put into effective use, it must be connected with additional associated IC's and electronic components to make up a microcomputer (μCP). The degree of complexity is related to cost in meeting a particular specification. Costs vary from less than £100 for a very simple single-board μCP to well over £10 000 for a sophisticated professional development system.

Due to the economic advantages offered by today's microelectronics, the application interests have increased and overlapped into a wide range of disciplines. Those requiring some form of educational insight into the field of microprocessors and microcomputers include: managers, accountants, businessmen, administrators, stock-control personnel; mechanical, production, electrical and

electronic, chemical and physical engineers and technicians. Such a broad spectrum obviously requires different emphasis and consideration.

This book is designed to give an introduction to the field of microprocessors and microcomputers for technicians and engineers, and may be used as a text to enable readers to gain invaluable 'hands-on' experience using an inexpensive microcomputer system. Particular emphasis is with regard to the applications of a microcomputer as an 'industrial controller'.

Based on a standard *Intel* product, I have designed the Microcomputer Educational Development System, using the *Intel* 8085A microprocessor. This system is shown in the frontispiece, and is manufactured by *Beal-Davis Electronics Ltd.*, The Old Post Office, Whittington, Worcester. A 'bread-board' is included to enable digital electronics and interface circuits to be developed—so that control applications can easily be performed.

It is appreciated that the most difficult level of programming to learn is at the machine-code level, and this is developed progressively, so that the newcomer to this field may be introduced relatively painlessly to this task.

The text is developed to suit technicians and engineers, with sufficient details given on hardware relating to 8-bit microprocessors—in particular the *Intel* 8080A and 8085A—to enable a reasonable understanding of these, and microcomputer systems based on them, to be achieved. Areas of the text can be omitted by the reader on the first reading—depending on the reader's objectives, e.g., the hardware details may be omitted if the reader initially wishes to become proficient at programming such a machine as the MEDS-85. The text, developed in this way, should be of particular interest to students following the TEC/D and HTC/D courses.

Many programming exercises are included, commencing with very simple problems requiring only a few instructions, progressing to relatively more complex problems in an effort to develop the reader's skill in handling the Instruction Set. Solutions to these programming exercises are included in the Appendices. The concluding chapter gives complete details for the solution of two problems: the mathematical solution for the multiplication of two 8-bit numbers, and the practical application solution (including hardware) for the speed (or position) and direction of rotation control of a stepper motor.

I sincerely acknowledge my gratitude to all those who have contributed to the realisation of this book, in particular to *Intel* Corporation for their guidance and permission to reproduce diagrams and sections of text from their manuals. Finally, a special thanks to my wife, for her patience and efforts during the preparation and typing of the manuscript.

Barry G. Woollard
Longdon, Staffs.

Contents

1 Introduction to microprocessors

1.1 The digital computer

A digital computer performs a sequence of arithmetic operations on data by means of a stored program of instructions, each of which defines each step in the sequence.

A computer generally follows human behaviour when dealing with a problem. The information, recorded in one way or another, is read into the machine—*input*. The completed result is read out of the machine—*output*. In between these two is the 'work area' which we call the *central processing unit* (CPU), the *central processor*, or simply, the *processor*, in which reference is made to the stored data and the stored program—*storage*, and also in which the calculations are carried out—*arithmetic and logic unit* (ALU), and the whole process is supervised by the *control*.

A complete computer system basically consists of a CPU with a number of devices surrounding it called *peripherals*, as shown in Fig. 1.1.

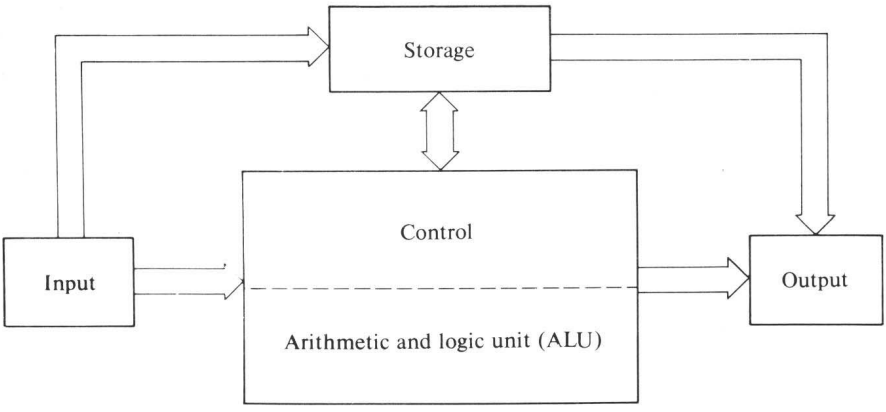


Fig. 1.1. Elements of the digital computer.

To operate a computer installation we must have a CPU and all the peripherals to support it—this is commonly known as the *hardware* of an installation. In addition, we need a *program* to direct the computer and the systems on which it will work—collectively known as the *software*.

The CPU is functionally organised into four basic sections, as shown in Fig. 1.2, and the binary information utilised in its operation may be separated into control information and data. The control signals are generated by a control unit to dictate the operation of the ALU, interface and storage (memory) sections and the flow of information among them via several links called data paths (or buses). Control information also determines the future operation of the control unit itself.

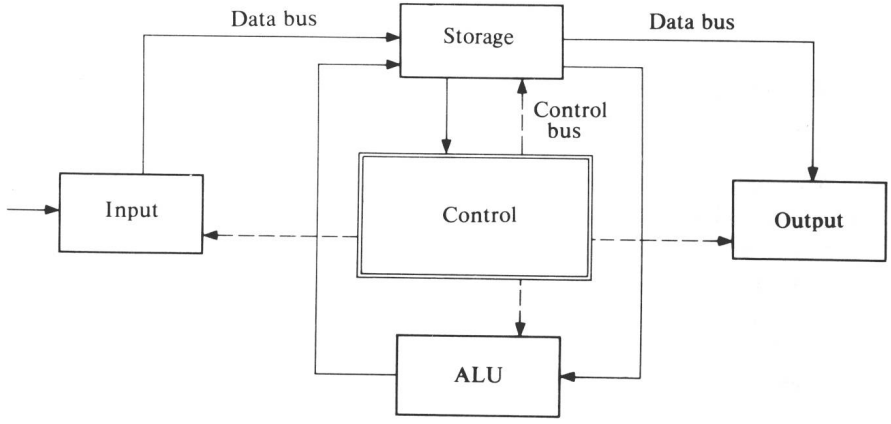


Fig. 1.2. Functional organisation of a simple computer.

Information not treated as control information may be regarded as data. So-called *machine language instructions* are a special type of data that are used by the control section to determine the proper generation of control signals to effect a program. In most systems, machine language instructions are indistinguishable from other types of data stored in memory.

1.2 What is a microprocessor?

The terms *microprocessor* and *microcomputer* are often confused and have been used interchangeably. The trend in terminology currently associates the term 'microprocessor' with the component product and the term 'microcomputer' with the board product.

A *microprocessor* (μP) may be a self-contained single-chip component, or a collection of unassembled processor-related components consisting of a CPU chip, memories, peripheral and I/O chips, clock, and interface chips. The composition varies, and is application and user dependent.

A *microcomputer* (μCP) is an assembly of microprocessor components mounted on a printed circuit board. The board contains all the components required to make a working computer, and may have resident software. These basic systems are generally expandable into larger systems. Variations of the basic component requirements are usually limited to a few basic types.

1.3 Microprocessor evolution

Intel produced the first general-purpose 8-bit μP in December 1971 in the form of the *Intel 8008*. This was manufactured using the p-channel MOS technology, and was packaged in a single 18-pin D.I.L. package. The 8008 used standard semiconductor random-access memory (RAM) and read-only memory (ROM), and (generally) TTL components for input/output (I/O) and general interface. It immediately found applications in byte-oriented end-products such as terminals and computer peripherals where its instruction execution (20 μs), general-purpose organisation, and instruction set matched the requirements of these products.

With the advent of high-production n-channel MOS RAM memories and 40-pin packaging, *Intel* produced the 8080A μP in December 1973. The 8080A utilises LSI techniques—having the equivalent of about 20 000 electronic components. It was designed to be software-compatible with the 8008, so that existing users of the 8008 could preserve their investment in software and at the same time provide a dramatically increased performance (2 μs instruction execution), whilst reducing the number of components necessary to implement a μCP system. Additions were made to the basic instruction set to take advantage of this performance and large-system type features were included on-chip such as DMA, 16-bit addressing, and external stack memory so that the total spectrum of applications could be significantly increased. Since 1973, the 8080A has become an accepted industry standard and, as such, is one of the most widely used μP's. Also accepted as industry standards are the *Motorola M6800* and the *Zilog Z80* μP.

1.4 The basic microcomputer system

The components of the basic microcomputer system are shown in Fig. 1.3, and can be seen to comprise:

(a) *Microprocessor*. This is the CPU, which performs all the control and timing functions in response to a bit pattern fed into it on a group of parallel lines. The bit pattern is fed into the CPU in groups of bits called *words*, and the number of bits in a word varies between μP's. The *Intel 8080A* (and *8085A*), *Motorola M6800*, and *Zilog Z80* μP use word lengths of 8-bits, i.e., *bytes*. A whole series of operations may be initiated by the CPU in recognition of a word, including:

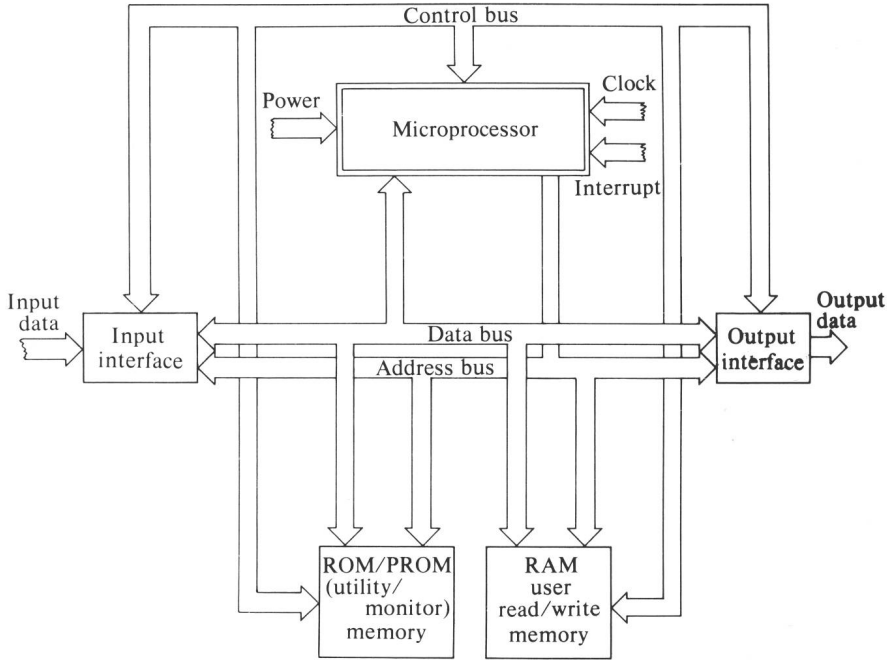


Fig. 1.3. The basic microcomputer system.

(i) Arithmetic, logic, and transfer operations;
(ii) instruction and data modification;
(iii) program transfer control.

(b) *I/O interface system*. This enables the μP to exchange data with a real-world device or system such as a teletypewriter (TTY), visual display unit (VDU), or the industrial process being controlled.

(c) *Memory*. This stores the program instructions and data. The machine operating instructions (editor, monitor, or utility programs) are usually stored in non-volatile storage—which is read-only memory (ROM)—in which the information remains even when the power is switched off. The user memory is generally random-access memory (RAM) which is volatile, i.e., the contents will be lost when the power is switched off. All memory cells may be considered as either *static*, in which the stored information is maintained as long as the power is on, or *dynamic*, in which the information is retained as a charge on a capacitor, but must periodically be subjected to a *refresh cycle* to compensate for the leakage of charge from the capacitors. Dynamic memory allows a greater packing density (16-Kbit dynamic RAM's are widely available) than static RAM's (4-Kbit static RAM's are widely available).

Several different forms of ROM are currently available:

(i) *Mask-programmable*—a standard or customer (debugged) specification is programmed during the manufacturing process, so that

once programmed this form of ROM cannot be altered. Although this is expensive it is well suited for large production runs.

(ii) *Fusible-link PROM*—this is supplied in 'blank' form, and customers program their own requirements by applying high-voltage pulses to the data output pins of the device. There are two types of fusible-link PROM: one has all its cells connected and the programming pulse is used to 'blow the fuse'; the other has all its cells open-circuit and the programming pulses are used to break down a reverse biased diode to give the required connection. Once programmed, these devices cannot be re-programmed, so that they become ROM's, and are useful in development systems in which the program has been well proven.

(iii) *EPROM's and EAROM's*—the program can be completely erased in the EPROM by exposing it, through the window, to strong UV light for about 20 minutes. It can then be re-programmed in a similar way to that used for the fusible-link PROM. The EPROM is ideal for use in prototype systems. In the Electrically Alterable ROM (EAROM), data can be erased by applying a high voltage pulse to the programming pins. This type of device is word-programmable, i.e. one word can be erased and re-written without affecting the rest of the contents. However, this means that it is time-consuming to program the whole of the device initially.

(d) *Bus system*. This is the network of paths which facilitate the flow of data. The important buses in a microprocessor system are identified as the data bus, address bus, and the control bus.

1.5 Microprocessor architecture

It is not possible to draw the circuit diagram of an integrated circuit (IC), even if we should want to do so, due to the short-cuts that the technology has allowed, *but*, it *is* possible to draw a diagram of the *functional* organisation (architecture) of the microprocessor IC chip, as shown in simplified form in Fig. 1.4.

A typical microprocessor organisation comprises the following units:

(a) *Arithmetic and logic unit (ALU)* performs various forms of addition and subtraction, and the logic mode performs such logic operations as ANDing the contents of two registers, or masking the contents of a register. The ALU also contains the *accumulator*, which is the main working register into which data and results are written and processed, and from which they are subsequently transferred to memory and output. The accumulator generally operates with one of the internal 'scratchpad' registers or with a memory location.

(b) *Working (scratchpad) registers* are used to store information during program execution. These registers can be accessed more easily and quickly than

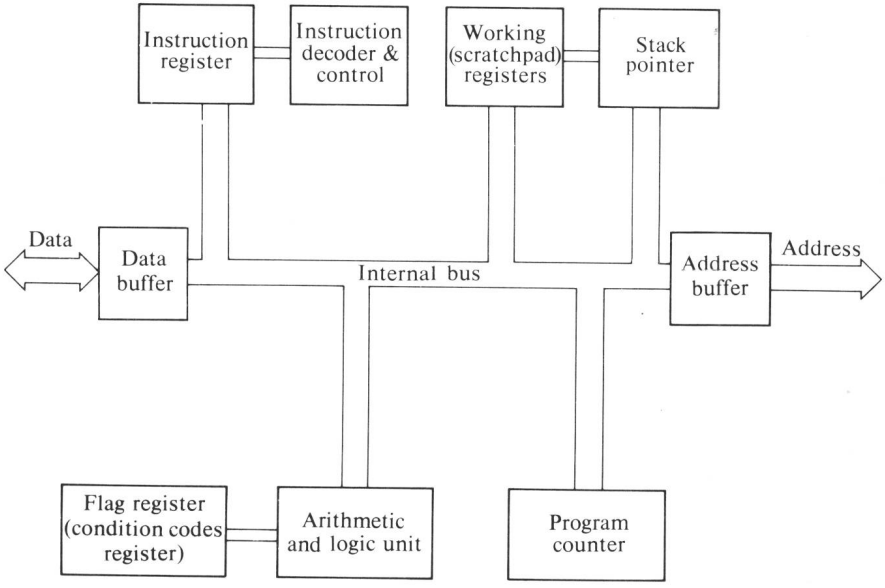


Fig. 1.4. Microprocessor architecture.

external storage, and increase the computational capabilities of the microprocessor.

(c) *Flag register* is used in conjunction with the ALU, and is sometimes called the Condition Codes Register. It comprises several flip-flops (bistable elements) whose states are used to indicate the condition of a particular operation.

(d) *Program counter* contains the address of the next program instruction. The activities of the CPU are cyclical; the processor *fetches* an instruction, *decodes* it, *executes* it, then fetches the next instruction, and so on. The processor updates the program counter by adding '1' to the program counter each time it fetches an instruction, so that the program counter is always current.

(e) *Instruction register and decoder*. The processor fetches an instruction in two distinct operations: firstly, the CPU transmits the address in its program counter to the memory; then the memory returns the addressed byte to the CPU. The CPU stores this instruction byte in the instruction register, and uses it to direct activities during the remainder of the instruction execution. The translation of this instruction code into processing action is performed by the instruction decoder and by the associated control circuitry.

(f) *Control*. The control circuitry is the primary functional unit within the CPU.

Using clock inputs, the control maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control issues the appropriate signals (to devices both internal and external to the CPU) for initiating the proper processing action. The control unit is often capable of responding to external signals, such as an interrupt or wait request. An *interrupt* request will cause the control to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program. A *wait* request is often issued by a memory or I/O element that operates more slowly than the CPU. The control causes the CPU to idle until the memory or I/O port is ready with the data.

(g) *Stack pointer.* The stack is a block of successive memory locations which is accessible from one end (LIFO). The stack is coordinated with the stack pointer to keep track of the storage and retrieval of each type of information in the stack. The stack pointer contains the address of the memory location at the top of the stack.

(h) *Internal bus* is shared by all the internal registers and the ALU and carries the data and timing information dictated by the control unit.

(i) *Data bus buffer.* This bidirectional buffer is used to isolate the CPU's internal bus from the external data bus.

1.6 Word—byte—nibble

These terms are frequently misused in describing microprocessor data. For a specific microprocessor, a *word* is the number of bits associated with the instruction or data length. This may be 4, 8 or 16 bits, etc., depending on the machine. A *byte* specifically refers to an 8-bit word, and can be manipulated by a 4-, 8-, or 16-bit microprocessor. For example, instructions are often provided to deal with byte data in 4-bit or 16-bit processors. This is called byte handling, and is independent of the natural word sizes of the machine.

A *nibble* is four bits, and it takes two nibbles to make a byte. Nibble, or 4-bit control, can be found on many 8-bit word machines, as well as some 16-bit machines. Four-bit operations are usually associated with hexadecimal or binary-coded decimal (BCD) operations.

1.7 I/O lines/ports

Microprocessor input/output (I/O) ranges from the complex floppy-disc control peripheral IC chips to simple 8-bit I/O latches. The input and output facilities are the devices by which the CPU communicates with the external environment.

If the microprocessor has an 8-bit internal structure, the I/O bus and each respective I/O port will also be of eight bits. A *port* is a collection of individual

I/O lines, the number of which is equal to the basic microprocessor word size. A port can be either input or output, depending on the nature of the I/O command—if it is bidirectional.

The majority of microprocessor applications have I/O requirements for between 32 and 64 I/O lines. This includes simple I/O as well as complex I/O peripherals, such as communication devices, CRT, controllers.

1.8 Instruction cycle

An instruction cycle is the periodic process of fetching an instruction from the microprocessor memory. This can consist of one access to memory or several accesses, depending on the type of instruction and the data required by the instruction. These are called memory cycles, which are the subset of instruction cycles.

Literal type instructions generally require one cycle, since the instruction data is included in the instruction word. A conditional jump may require one cycle to fetch the direct or indirect address.

When a microprocessor manufacturer specifies a cycle time, the user must consider the following three quantities: minimum cycle time, typical cycle time (the number often quoted), and maximum cycle time. Typical cycle time tells you what the overall periodic time is for an instruction looking at a typical mix of instructions. This is the best measure of instruction cycle time. Maximum cycle time is the period of the largest instruction. A machine is efficient when the typical and minimum cycle times are close.

1.9 Interrupt

Many CPU's have interrupt facilities in order to improve the efficiency of response to real-time demands. In an interrupt, an I/O device attracts the CPU's attention by activating an interrupt line, to indicate that some external activity is either nearing completion, or about to be initiated. The interrupt, if enabled by the CPU, causes the CPU to suspend temporarily the operation it is performing and effect a jump to an interrupt subroutine.

A system may have several sources of interrupt, so the microprocessor must determine which of them is requesting service, since each requires different attention. This may be handled in two ways by the CPU. Some CPU's require that the interrupt device provide an address at the time of interrupt acknowledge. This address is used to compute the effective address of the interrupt routine of interest. Alternatively, the CPU can be caused to jump to a fixed interrupt service location, and have a small subroutine to sort out which device initiated the interrupt. When this sorting is done, a software jump to a subroutine to service the interrupt will ensue.

1.10 Software and firmware

The microprocessor generally forms part of a stored program computer. The collection of programs and instructional procedures is referred to as *software*. The software is designed to make the microprocessor perform some functional system-related task such that the operation of the program is performed by the hardware. In a fixed-instruction microprocessor, a fixed number of instructions or operations is defined with fixed word lengths that exercises the CPU independently of the data.

Software is accessible to and alterable by the user, and programs can be stored in RAM. *Firmware* is generally committed to a permanent storage medium such as ROM, and is not accessible to the user.

Firmware, or microprogramming, affects how the actual hardware operates. Its effects on hardware are less noticeable than software. A firmware program may be used to implement a software instruction set.

1.11 Simulators and emulators

A *simulator* is a device used to imitate one system with another, using a software program written in Assembler or a high-level language. The simulator accepts the same data, executes the programs, and accomplishes the same results as the system being imitated. The simulator is generally much slower than the machine simulated, and bears no physical resemblance to it. Simulators may be used to get a finer insight into the workings of the imitated system. Generally, larger microcomputers (or minicomputers), like software development systems, will be used to simulate smaller microprocessors to develop and debug software.

An *emulator* differs from a simulator in that the latter uses software to imitate the other system. The emulator uses a microprogram and specific hardware to imitate the desired system at, or faster than, the imitating system's cycle time. Slice microcomputers are used to emulate large microcomputers (or minicomputers) with fewer components and faster cycle time. The emulator can be made to physically resemble the imitated machine.

1.12 Comparison of microprocessors

A wide range of 8-bit microprocessors have been produced by many manufacturers. Some of these have evolved as being more popular than others, as shown in Fig. 1.5. The most popular of those listed in Fig. 1.5 are the *Motorola* MC6800, *Intel* 8080A and 8085A, and the *Zilog* Z80A—and these have become adopted as industry standards. Many 16-bit microprocessors are becoming readily available. As costs tend to fall, it is expected that 16-bit microcomputers will increase in popularity—those shown in Fig. 1.5 currently being the most popular.

8-bit:

Manufacturer	Type	Supply	Instruction set	Clock (MHz)	Instruction cycle (μs)	Interrupts	Pins
Fairchild	F8	+5 V	76	2	2	1	40
Intel	8080A	±5 V, +12 V	78	2	2	1 vectored	40
Intel	8085A	+5 V	80	3	1.3	5 vectored	40
Motorola	MC6800	+5 V	72	1	2	2	40
National	SC/MP	+10 to +14 V	46	1	20	1	40
RCA	CDP1802	+4 to +10 V	91	6.4	2.5	1	40
Rockwell	6502	+5 V	56	2	1.5	2	40
Signetics	2650	+5 V	75	1.25	2.4	1 vectored	40
Zilog	Z80A	+5 V	158	4	1.25	4 vectored	40

16-bit:

Manufacturer	Type	Supply	Instruction set	Clock (MHz)	Instruction cycle (μs)	Interrupts	Pins
Intel	8086	+5 V	133 basic	5	2	256 vectored	40
Motorola	MC68000	+5 V	61 types	4	1.5	8 levels	64
Texas	TMS9900	+5 V, ±12 V	69	3	5.9	16 levels	64
Zilog	Z8000	+5 V	110 basic	4	2.5	256 vectored	48

Fig. 1.5. Comparison of a selection of Microprocessors.