Microprocessors in brief

by Robert C. Stanley

This paper presents a tutorial overview of the past, present, and future of microprocessors and describes the key elements of their structure and operation. It is intended to serve as a technical introduction to the rapidly expanding field of microprocessor and microcomputer technology and to provide an overview of what these elements are, what they can do, and how they do it. The origin and evolution as well as the basic principles of operation are discussed. Several different types of microprocessor are considered and examples of their application in the solution of real-world problems are given.

Introduction

The microprocessor, an invention of the early 1970s, essentially incorporates the computational power of a computer in a package that can be held in the palm of one's hand. Microprocessors are currently being used extensively in lieu of conventional logic to reduce product cost, add more functions, and increase reliability through fewer part numbers, reduced hardware, and less complex packaging.

The microprocessor is viewed differently by different people. To an end user it might be just a black box, to a programmer it might be a smaller version of the central workings of a computer, or to a control logic designer it might be a logic device that continually rewires itself according to the instructions sent to it. No matter what the point of view or intended end use, the microprocessor must contain certain elements in order to perform the required tasks. What is commonly referred to loosely as a "microprocessor" usually consists of a small group of devices that combine to make up the control nucleus of a microprocessor-based system. (See **Figure 1**.) It is this nucleus of necessary devices that we are primarily concerned with in this paper.

Basic operation of the various microprocessor system elements is covered first, followed by a step-by-step description of the execution of instructions in a simplified program. A number of more intricate concepts are discussed under *Advanced concepts*, and *The future* takes a look at where microprocessors are going from here. Example 1 is a simple cash register control, whereas Example 2 details the more sophisticated master/slave microprocessor control of a robot.

In order to shed some light on the original purpose of the device and to show how it has since grown to fill other needs, details of the origin and evolution of the microprocessor are given in Appendix 1. The physical packaging aspects of microprocessors and the details of operation of the various types of microprocessor memory devices are mentioned only briefly in the body of the paper, but are covered in some depth in Appendices 2 and 3. A bibliography has been included with references to more detailed tutorial and technical literature that covers the many aspects of microprocessors and how they are created and applied.

Basic microprocessor operation

The primary control device in a microprocessor system is the CPU (central processing unit). Here, most of the decision-making is performed through the execution of program instructions. These instructions are stored in memory and the CPU fetches them as needed to perform the required task. Most instructions are very basic in nature; they direct simple operations such as reading in data from an outside source, moving data from one storage register in the CPU to another, performing some logic or arithmetic function on

the contents of a register, or writing data out to another part of the system. There are also instructions that perform more complex functions, but regardless of the complexity of a single instruction it cannot accomplish even the simplest task entirely by itself. Individual instructions must be combined in some logical fashion to create a program to step the CPU through a series of basic operations that will, when taken as a whole, perform some useful function. Sometimes the program instructions are stored in the CPU itself, but more often they are stored in a separate memory device. A microprocessor needs memory for storing both instructions and data. The various types of solid-state memory modules used in microprocessor-based systems are described in Appendix 2.

A typical CPU chip consists of several separate logical sections as shown in **Figure 2**. There must be a control ROM (read-only memory), a PLA (programmed logic array), or a random logic decoder (or some combination of these devices) to decode the instructions one at a time and to direct the operation of the rest of the CPU chip. There is timing and sequence logic that steps each operation through in its proper order. There is an ALU (arithmetic logic unit) that performs basic arithmetic and logical operations on operands that are fed through it. There are normally a number of registers of various sizes located on the CPU chip itself. There are address pointer registers whose width depends on the size of memory the system is designed to handle and on whether the memory being addressed is in the CPU or external to it. There are data registers for storing and transferring data, and at least one of these registers is normally a special-purpose working register called an accumulator. The accumulator is involved in most of the data-oriented activity on the CPU. (The results of most of the ALU operations are sent to the accumulator, and its contents are quite often used as one of the operands.) Connecting all of these elements is a data path whose width is determined by the microprocessor word size. This data path, with bidirectional buffers at the boundary of the CPU chip, becomes the local system data bus and acts as the information path connecting all data-related elements in the system.

The contents of the active address pointer register generally follow a separate path to the boundary of the CPU chip, where it passes through buffers to become the local system address bus. A 16-bit address bus allows addressing of 65 536 (often referred to as "64K") separate memory locations, and a 20-bit address bus allows for over a million. In an effort to reduce the number of pins on the CPU package, some microprocessors multiplex some portion of the address bus and data bus on the same group of pins as they leave the CPU chip. This saves pins on the CPU, but requires that extra hardware be added to create individual address and data buses to serve the rest of the system. This is of little consequence in larger systems, however, because the

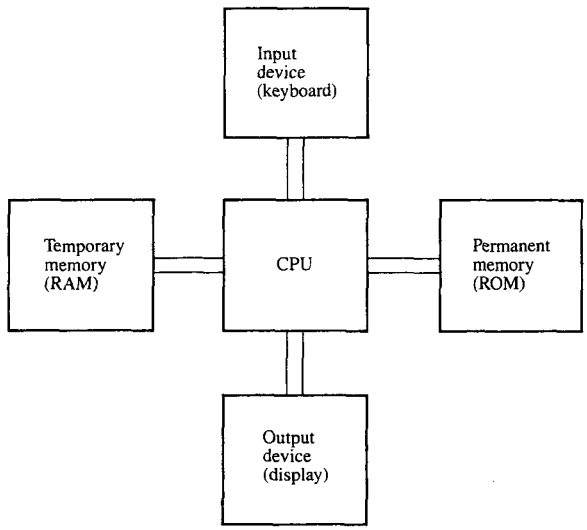

Figure 1

Simplified microprocessor block diagram.

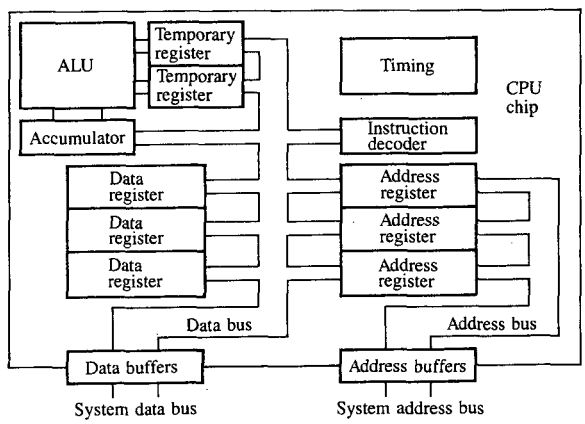

Figure 2

CPU block diagram.

local address and data buses must be buffered again before being distributed to a large number of memory and peripheral chips, and the demultiplexing and buffering can both be done by the same devices. (The address bus is unidirectional only, out of the CPU, but the data bus is bidirectional and must be buffered in both directions.)

Microprocessors are sized or classified according to the number of binary digits or bits that they are able to handle at one time. An eight-bit microprocessor generally has an

eight-bit-wide data interface to the outside world and eight-bit internal data paths and registers. When there are exceptions to this, it is generally that the internal data path is wider in order to increase the functional capability of the CPU, or the interface to the rest of the system is narrower to reduce cost or the number of package pins. (There is at least one microprocessor with a four-bit word size and a one-bit interface off the CPU.) In eight-bit microprocessors, data are moved around in groups of eight bits, which are referred to as "bytes." A byte is always eight bits, and a "nibble" is half a byte or four bits, but a word may be of any size as defined by a particular microprocessor. The most common word sizes for microprocessors today are four, eight, sixteen, and thirty-two bits.

A microprocessor needs a means of interfacing with other major elements in a control package or directly with the outside world. This is provided by interface elements or mechanisms called I/O (input/output) ports. These I/O ports may be located directly on the CPU itself or on separate chips, and can handle input or output data in either serial or parallel format. Usually separate I/O ports are required for input and output data and for serial and parallel format, but there are sophisticated devices available today that may be programmed to handle many types of I/O, and even to handle automatically much of the protocol involved in some of the more involved serial communication formats. Parallel I/O ports may be designed for data input or data output only, or may handle bidirectional data. Some allow for the selection of input or output function on a bit-by-bit basis, but most ports are configured in groups of four or eight bits. If the I/O port is located on a peripheral chip and not on the CPU, it is connected to the CPU via address, data, and control lines in a manner similar to that used for memory modules.

"Address modes" is a term which describes the different methods a CPU uses to address data stored for future use (normally in memory). Some CPUs have a number of different registers that may be used as address registers, and some of the address modes use these registers in different combinations. There are some address modes that do not use address registers as such. In "immediate" addressing, the data are included as part of the instruction and are therefore pointed to by the program counter. In "implied" addressing, the location of the data to be used is implied by the instruction itself. "Direct" addressing includes the address of the data as part of the instruction; "direct short" uses eight bits of address (near the bottom or lower addresses of memory), and "direct long" uses 16 bits of address to access data anywhere in a 64K-byte range. In "register" addressing, the data are found in a register in the CPU, and in "register indirect" addressing, the instruction refers to a register that contains the address of the data in memory. "Indexed" addressing makes use of index registers in the CPU, and there is normally a method of automatically adding an offset to the contents of the index register before it goes out on the address bus (as for indexing into a data table). The index registers may also be caused to increment or decrement automatically by a specified amount every time their contents are used on the address bus.

Accessing program instructions is not usually considered one of the address modes in a microprocessor, since instructions are normally pointed to by a special-purpose address register on the CPU called a "program counter." The program counter points only at instructions and is automatically incremented to the next instruction every time it is used. The program counter may also be force-loaded with an address, as in the case of program jumps where the next instruction to be executed is not the next one stored sequentially in memory. The program counter might also be referred to as an instruction pointer register.

Another register not included in the address modes is the stack pointer. This is an address register that is normally loaded in the IPL (initial program load) routine to point to an area in RAM that has been set aside to store return addresses and miscellaneous data. This RAM area is referred to as a "stack," and its size determines the number of subroutines and interrupts that may be nested or overlapped. On a "call to subroutine" instruction, the CPU saves or pushes the address of the next instruction onto the stack by using the stack pointer as an address pointer and the contents of the program counter as the data to be written in memory. Next, the program counter is loaded with the address of the first instruction in the subroutine being called and this causes a jump to that subroutine. (Subroutines are used when there is a series of instructions that are to be used over and over again in a program, as for a routine to read data in from a keyboard every time a key is pressed.) The last instruction in a subroutine is "return from subroutine" and it merely pops the return address off the top of the stack and places it in the program counter. Operation is then resumed immediately following the point in the program where the subroutine was last called. The stack pointer register is automatically incremented and decremented by the CPU so that it always points to the last entry or next available space. Push and pop instructions are available that allow the programmer to store temporary data on the stack or to save the contents of registers when a subroutine is called or an interrupt occurs. (The "push" instruction writes to the stack and "pop" reads from it.) The stack acts as a LIFO (last-in first-out) register, so any data must be popped or retrieved in the reverse order from that in which they were saved or pushed.

Some microprocessors, referred to as register-based, have a large number of registers on the CPU, and the instructions specialize in operating efficiently on data in these registers. Another type, referred to as memory-based, has fewer registers on the CPU, and the instructions are oriented more toward operating on data in memory. Still a third variety has

very few, if any, data registers on the CPU. Instead, there are pointers to memory where the "register" space actually resides. This fact is almost transparent to the programmer, except that there appear to be a number of complete sets of working registers that are easily accessible.

Tying together all the basic components in a microprocessor system generally involves a number of SSI and MSI logic chips. Buffers are required on the data bus to buffer the CPU from the electrically harsh environment on the system data bus and to increase the drive capability of the bus. Address decoders decode the higher-order address lines and provide chip select signals to activate the proper memory or I/O chips in response to a certain range of addresses. Multiplexing, or time-sharing, of address information and data on the same CPU pins requires that the system provide address latches to hold the address bus active throughout the entire memory read or write cycle. The basic system timing in a CPU is often derived from a crystal, and this usually requires a separate clock oscillator chip to convert the crystal frequency into clock signals suitable for the CPU chip. Some CPUs generate their own bus control signals, but more complex systems often utilize bus controllers to handle the read and write timing and other bus-related functions. What is really a fairly simple microprocessor system can easily result in a printed circuit card with 20 or more devices on it. What is often referred to as a "computer on a chip" today is frequently in reality more of a "CPU on a chip." There is a constant effort, however, to reduce the number of chips required to form a working system. (Refer to "microcontrollers," discussed in the section on advanced concepts.)

Instruction execution

In order to see just what a CPU does in carrying out its duties, we explore a short scenario of reading in some instructions and following the CPU through its paces. We assume that there is a program already in program memory to read the data at an I/O port, mask off some of the bits, and store the results in data memory. We assume also that this program starts at memory address 0, and that after it has been reset, this is where the CPU will look for its first instruction. For our example we assume that our microprocessor is accumulator-based with isolated I/O and uses eight data bits and 16 address bits. **Figure 3** is a memory map for this program.

The first instruction in our program is a two-byte INPUT instruction. The first byte is the op code, or operation code, that calls for the reading of data from an I/O port. The second byte of the INPUT instruction is the number of the port that is to be read. In our example it is port 0. The second instruction is the AND instruction, which performs the masking of the bits. This also is a two-byte instruction, with the first byte being the op code, and the second byte being the immediate data to be used in the logical "and"

0007	HALT
0006	34
0005	12
0004	MOVE
0003	07
0002	AND
0001	00
0000	INPUT

Figure 3

Sample program memory contents.

operation. In our example we wish to mask off, or disregard, the high-order five bits of the input data, so the immediate data byte in the instruction is 07 (binary pattern 00000111). The third instruction is a MOVE instruction that takes our masked data and moves them out to memory. This instruction is three bytes long, with the first byte, as always, being the op code. The second and third bytes contain the 16-bit address of the memory location in which we wish to store the data. For our example we use memory address 1234. To prevent the CPU from continuing and doing anything after it has finished our example, we end the program with a HALT instruction that stops the CPU from executing any more instructions.

After the CPU has been reset by an external reset, the program counter contains all zeros, and its contents travel through address latches and out on the address bus to access memory location 0000. (Memory addresses are to be specified in our example by using four hexadecimal digits.) The CPU also sends out a memory read command so that the contents of the selected memory location are placed on the data bus by the memory module. The CPU reads the data bus, which now has the op code for the first instruction. Since this is the first operation after a reset, the CPU knows it must be reading in an op code so it places the first byte in the instruction decoder to be decoded. Upon decoding the instruction, the CPU discovers that it is a two-byte instruction and it must fetch the second byte. The program counter was automatically incremented after its contents were saved in the address latches, so it now contains address 0001. This address goes through the address latches and out to access memory location 0001 for the second byte. The

second byte of the first instruction, which is 00 to represent port 0, goes back to the CPU by way of the data bus. Since this is an INPUT instruction, the CPU places the second byte of the instruction in the address latches as the lower half of the address of the I/O port. (The upper eight bits of address are zeros for all I/O operations.) Address 0000 goes out on the address bus, but this time the CPU sends out an I/O read command because it is reading an input port and not a memory location. The combination of address 0000 and the I/O read line being active causes input port 0 to place its contents on the data bus. (We might assume that the port is wired to some switches that we wish to monitor in a control application.) The CPU reads the data bus and places the data in the accumulator. (The accumulator is the destination for all INPUT instruction data and the source for all OUTPUT instruction data.) This concludes the execution of the first instruction, so the CPU prepares to read in the next op code.

The contents of the program counter again go out on the address bus, this time with address 0002 and a memory read signal. This causes the op code for the second instruction to go to the instruction decoder for decoding. The CPU discovers that this is a two-byte AND instruction and it sends the program counter contents of 0003 out on the address bus to fetch the second byte. The data in the second byte of the AND instruction are to be logically "anded" with the contents of the accumulator, so they are placed in a temporary holding register. The contents of the accumulator are copied into a temporary accumulator and then both temporary registers are sent through the ALU, where they are "anded" together. The results of the ALU operation are returned to the accumulator, which now contains the low-order three bits of the data that were read in from port 0. "Anding" a number with 0 gives a result of 0. "Anding" a number with 1 leaves the number itself. Thus, our "anding" the accumulator with immediate data of 07 (00000111 in binary) has had the effect of masking the high-order five bits to zeros and leaving the low-order three bits unchanged. This concludes the execution of the second instruction, so the CPU prepares to read in the next op code.

The contents of the program counter go through the address latches and out on the address bus with address 0004 and a memory read signal. This causes the op code for the third instruction to follow the data bus from memory into the CPU and then to the instruction decoder for decoding. The CPU discovers that it is a three-byte MOVE instruction, and it sends the program counter contents of 0005 out on the address bus to fetch the second byte. The data in the second byte of the MOVE instruction are to be part of a memory address, so they are placed in half of a temporary address register in the CPU. The program counter was automatically incremented after its contents were saved, so it now contains address 0006. This address goes through the address latches and out to access memory location 0006 for the third byte of the instruction. The third byte of the MOVE instruction is the rest of the 16-bit memory address, so it goes to the other half of the temporary address register. The MOVE instruction is being used to move the contents of the accumulator out to data memory, so the temporary address register goes through the address latches to access memory location 1234. In this case we are moving data out of the CPU, so the contents of the accumulator are placed on the data bus and the CPU issues a memory write command. This causes the data on the data bus to be latched into the memory location that is being addressed by the address bus and concludes the execution of the third instruction.

The HALT instruction is then read in from memory location 0007 and decoded. The CPU now remains in an idle state until it is given an external reset or an interrupt. No matter how long or involved a program may be, it is always executed one step at a time, as we have just seen (though for ease of exposition we did leave out a number of the details). There are microprocessors that can operate on more than one instruction at a time, and many of the instructions cause more activity than those we looked at, but it is always done one step at a time.

Programs to control microprocessors may be written in a number of different ways. The most basic of these is "machine language," which comprises the actual binary zeros and ones that the microprocessor responds to when it reads them in as instructions. The zeros and ones are fine for the microprocessor, but it is not comfortable for programmers to have to write programs using only numbers, so this is not done very often unless the application is expected to sell in extremely high volume.

The next level of programming is that of using "assembly language," where all of the binary machine instructions have been assigned names that represent their operation. Examples of these are AND, INPUT, ADD, JUMP, SUBTRACT, etc. The programmer writes a program using the individual instructions by name. (These names are referred to as mnemonics.) The mnemonics are then entered into a computer along with a program called an assembler, and the mnemonics are converted into the binary numbers that the microprocessor actually operates on. Assembly-language programs can be very efficient because the instructions are selected one at a time, but they also take a long time to write, and the programmer must be familiar with all the details of how each particular microprocessor handles each instruction. Assembly language is used today for most programs that are relatively short and are stored in ROMs or EPROMs and sold in large volume.

Higher-level languages are available which make it easier to write programs because the language sounds more like basic English statements than elemental operation steps in a computer. After a program has been written in a high-level language, it is entered into a computer along with a program

called a compiler. There are different compilers for each of the high-level languages (such as BASIC, Pascal, FORTRAN, etc.), and there are specific versions for each of the target microprocessors. The compiler lists or compiles the individual mnemonics required to execute each of the general program statements, and then its output is run through an assembler to create the actual machine code or instructions. High-level languages save time for the programmer, but quite often make poor use of the microprocessor time and memory because the compiler does not always make optimum use of the instructions. Some programs are written in a high-level language and then time-sensitive portions are rewritten in assembly language to optimize the operation of the overall system.

Advanced concepts

There are many different types of microprocessors available today. Some of them do not have all the features mentioned in the section on basic operation, but others have most or all of these features plus many more. Sometimes a number of powerful features are built into the CPU itself, or in some cases a rather ordinary CPU is applied in a system incorporating very sophisticated peripheral chips. The applications run the gamut from, for example, washing machine control or children's toys to word processing to real-time image processing and continuous-path machining.

One feature that might be incorporated in a CPU to increase its performance is pipelining. An example of this may be seen in instruction decoding. In our simple example we saw the instructions going directly from the data bus to the instruction decoder one byte at a time for decoding. Some microprocessors have instructions that are over half a dozen bytes long, and the CPU might have a queue for storing several bytes of instruction before they go to the instruction decoder. This saves CPU execution time in two ways. First, separate bus control logic on the CPU can be dedicated to keeping the instruction queue full so that the rest of the CPU may concentrate on executing the instructions, and not on trying to make optimum use of the bus. (Simple microprocessors spend most of the time fetching instructions from memory and in the read/write data operations on the bus, and not on actual arithmetic and logical operations that are performed within the CPU.) Second, the instructions may be pre-decoded to a certain extent in the queue so that some operations may overlap in time if they do not require the same portion of the CPU logic or depend on the results of a preceding instruction. Pipelining may be as simple as allowing the execution of one instruction to overlap the fetching of the next op code or as advanced as operating on several instructions at the same time. The objective, of course, is to execute instructions as rapidly as possible and to speed up the overall system operation. There are microprocessors today that do a high-speed fetch of a number of instructions and place them in an instruction cache or temporary storage location on the CPU chip itself. There they can be accessed more rapidly by the CPU than if they were left in system memory. This speeds up the overall system operation considerably.

One of the trends today is toward more and more complex peripheral chips to handle such things as serial communication protocols, display screen control, parallel I/O with handshaking, DMA (direct memory access), floppy disk controllers, and multifunction timer/counters. These chips normally have a number of control registers that dictate how the chip functions in the overall operation of the system. When electric power is first turned on, these chips are typically unable to function without first receiving "operating instructions" from the CPU. The CPU, in its own IPL (initial program load) routine, encounters a series of instructions that tell it how to configure the peripheral devices in the system. It does this by sending a series of command words to the control registers in the peripheral devices themselves. These command words, which must normally be sent to the peripheral controller chip in a special sequence, are used to "customize" the operation of the chip to fit the application. After this initialization, the I/O chips automatically handle the data in the prescribed manner every time they are called upon by the CPU or by the peripheral device they are supporting.

The term *direct memory access* refers to an operation that transfers data directly to or from memory without the data being handled by the microprocessor chip itself. The DMA controller is an intelligent peripheral chip that must be initialized by the CPU and told where the data are to be moved from and where they are to be moved to. Once so initialized, the DMA controller takes control of the data bus, the address bus, and the control lines that handle memory read and write operations. The CPU is temporarily forced off the bus and must wait until the DMA operation is completed before it can regain control of the bus. DMA operations save time when large amounts of data are to be moved, because the DMA controller can automatically transfer data at the full bandwidth of the bus and does not need to keep reading in program instructions to tell it what to do next.

The microprocessors we have been talking about until now might be called "conventional" microprocessors. There is a type of microprocessor that is generally referred to as a single-chip microprocessor, or microcontroller. The main feature of these devices is that an attempt has been made to place the entire system on one chip. (This includes the CPU, RAM, ROM, clock oscillator, and I/O ports.) A microcontroller has enough on-chip ROM to hold a control program of reasonable size (i.e., 1K or 2K bytes). Some microcontrollers are available in both masked ROM and UV-erasable EPROM versions (as described in the section on memory), and some have provisions for additional off-chip ROM or EPROM to allow for future expansion of an

application program. Some microcontrollers also provide for off-chip data RAM in addition to the on-chip registers and scratchpad RAM. Some of the other features incorporated into microcontrollers to increase their control capability include Boolean processing, timers and counters (cascadable or with optional prescalers), A/D (analog-to-digital) conversion, PLLs (phase-locked loops), and the ability to handle serial I/O. Microcontroller applications today are expanding rapidly and will continue to do so in both high-end and low-end applications. Typical applications of the newer high-end versions are analog data processing or signal processing; many of the lower-end versions will be buried within electrical equipment where the presence of a CPU will probably not be recognized by the end user.

The coprocessor is another type of processor that has been developed to enhance the capabilities of a microprocessor CPU. An example of this is the NDP (numeric data processor). The NDP cannot operate in a stand-alone situation as the microprocessor CPU does, but typically is connected to the same address and data bus as the CPU. The NDP monitors the instructions as they go to the CPU, and when an NDP instruction is encountered, the CPU releases the bus and allows the NDP to take over the bus and execute the instruction. The CPU may wait while the NDP performs some floating-point operation, or it may continue on its own and then at a later time query the NDP for the results of the operation. The NDP instructions are intermixed with the CPU instructions, so the programmer need not be overly concerned that some instructions are being executed by the CPU and some by the NDP. Coprocessors speed up processing time because they are dedicated to performing specific tasks that would take much longer in the more versatile but slower CPU. They also save time by being able to operate in parallel with the main system CPU.

Interrupts are far from new (they have been available on microprocessors since the 8008), but they do require some attention to detail in both the hardware and the software design. The simple approach to interfacing with a peripheral device would be for the CPU to poll its controller. (Polling is software that periodically checks to see whether the device requires service.) The device might be a printer that is ready for more data to print or a tape reader that has data for the CPU. This method is all right for applications where time is not critical and the peripherals can wait for service. (The printer could theoretically wait all day for more data to print without causing a system problem.) The reader, on the other hand, could cause a problem. If the data were not read from it before more data were received, the new data would be written over the old data and would thus destroy the old data. In this case, the reader should be able to interrupt the CPU in whatever it is doing and request that the data be read in a timely manner.

Interrupts may come from just one device in a system, or from a number of devices. In any case, the CPU must know where in memory to find the service routine for the particular device that has caused the interrupt. It is normally up to the requesting device or an interrupt controller chip to place this information on the bus. The CPU saves, on the stack, the address of the instruction it was about to read in when the interrupt occurred. It then picks up the interrupt information from the bus and calls the proper routine to service the device. The last instruction in the service routine is "return from interrupt," which gets the return address from the stack, and the CPU returns to what it was doing when the interrupt occurred. Some CPUs have duplicate sets of registers so that an entire new set of working registers may be switched to rapidly when an interrupt occurs. If there are to be many interrupt sources in a system, a priority interrupt controller chip is used. This device keeps track of the devices that have requested service and allows interrupts of a higher priority to interrupt the CPU while it is servicing devices of lower priority. Some of the newer microprocessors and microcontrollers have priority interrupt hardware and microcode built into the CPU itself in order to simplify system design and eliminate the need for a separate interrupt controller chip.

Bit-slice microprocessors are a specialized type of microprocessor used in very high speed applications or where a specialized instruction set is required. The faster bit-slice microprocessors employ ECL (emitter-coupled logic) because this configuration of bipolar transistors can provide switching speeds in the subnanosecond range. The bit-slice microprocessor instruction set is determined by the system designer and is stored in a block of high-speed memory. (This technique is often referred to as microprogramming.) With microprogramming it is possible to optimize an instruction set for any required application, or to emulate the operation of existing microcomputers. The ALU portion of the processor is available in "slices" that are typically four bits wide. A number of these slices may be combined to build a processor that operates on larger word sizes (i.e., 12, 16, 20, or 32 bits). The advantages of a bit-slice microprocessor are the high speeds possible and the flexibility of being able to design your own customized instruction set. The disadvantage is the fact that the hardware and software must be defined in great detail by the system architect and cannot merely be picked off the shelf and used as can be done with the conventional microprocessor that has everything predefined by the manufacturer. The lower sales volumes of bit-slice microprocessors also tend to raise the overall cost of these systems.

Bus arbitration chips are available to help control access to the bus in multiprocessing operations, and memory management units allow a microprocessor access to more physical memory than its number of address lines would normally allow. The smaller device geometries used in the newer microprocessors are allowing them to operate at

higher and higher speeds, and features such as memory management are being incorporated into the CPU chip itself. Separate user and supervisor modes for system control, as well as support of trap instructions for error detection, have been available for several years, but the newer devices are providing much more of this type of support. The user/supervisor separation has recently been increased to four levels of protection. The concept of virtual memory, in which the program is unaware that some memory space is in main storage and some is in external storage, has been extended to the concept of a virtual machine. There is better support for coprocessors in the newer devices, and the register sets and instructions are becoming better suited to compiler-generated code. Most of the new processors utilize memory segmentation, but some still provide for a form of linear addressing. Many data types are supported, and some devices go so far as to provide a descriptor architecture in which the information stored is only one step from being self-identifying. The end result is that the newer microprocessors are taking on more and more of the attributes of the traditional mainframe computer.

The entire microprocessor field has been expanding at such a rapid rate that it is difficult, if not impossible, to keep up to date on all aspects of it. One area that has suffered has been that of terminology. New products have been introduced at a rate that has prevented the literature from ever completely catching up; as a result, much of the terminology in use today is imprecise and inconsistent. One cannot fall back on the conventional field of electricity and electronics as it existed and was taught in universities prior to the 1970s. Today a student's knowledge can become obsolete in the same year he graduates. The differences in documentation also hinder systems designers when they must work with literature and data sheets from a number of different manufacturers and combine different technologies from different vendors into a working and reliable system. Another problem in the microprocessor field is that its early development was driven to an extent by hobbyists and toymakers as well as by industrial designers and programmers, and some of the effects of their influence can still be seen today.

The future

Microprocessors have already changed the way most of us live our daily lives at home, at work, and at play. Just over a dozen years ago microprocessors did not exist, and today there are millions of them in operation in such unlikely places as a child's toy or the engine of a car or an electric mixer in the kitchen. To state that they will continue to change our lives is like pointing out that the sun will come up tomorrow. What nobody knows today is where this will take us before the end of the century, because by then it is very likely that more microprocessors will be used in applications that do not even exist today than all of the microprocessors currently in use. The world is still trying to adjust to living with computers, and now we have a new breed that will fit under a thumbnail.

Some of the changes currently taking place are quite obvious. More peripheral functions are being added to the CPUs and more intelligence is being added to the peripheral devices to the extent that very soon there will be cases where it will be difficult to distinguish one from the other. Higher-level language translators are being incorporated in silicon and will facilitate human communication with the new devices, and the devices themselves are being designed to operate more efficiently with the higher-level programming languages. Microcontrollers are being improved in function and in bit-handling ability; this will ease their introduction into an entirely new field of low-level applications such as "smart" traffic lights, vending machines, and electrical appliances which will become a natural part of our everyday lives. Microcontrollers are also being improved to handle higher-end applications such as signal processing and specialized industrial controllers, and are taking on many of the same features that are being added to the newer CPU chips.

Many different names are applied in reference to the technology used today for the silicon chips themselves, but most are variations of NMOS. (Refer to Appendix 3.) CMOS is now becoming faster in operating speed and smaller in size, and is increasing in popularity to the point where some manufacturers are introducing new parts only in CMOS. Another recent development is the radiation-hardening of the parts in order to increase their reliability in certain harsh environments being encountered in some applications.

While microprocessor use will be quite visible in some areas, many of the new applications will not be as noticeable to the end user. (The motorist will be aware of the processor in the dashboard of his new car, but he may not know that there is actually a small CPU in the windshield wiper motor, fuel injector, voltage regulator, headlights, heater, and in each of the turn signal and brake lamps.)

The higher-end applications such as data processing and industrial control (and even some of the personal computers) will be much more visible to the end user. A number of thirty-two-bit devices are already available and even larger word sizes will surely follow. This does not mean, however, that there will be a mass movement to the larger word sizes. The first microprocessors were four-bit devices, and although they did not find their way into a large number of different applications, they still remain as the largest-selling microprocessors. Eight-bit microprocessors come in second as far as volumes shipped, but they do appear in the largest number of different applications today. Sixteen-bit microprocessors are now reaching their maturity and appear in an ever-increasing number of applications. The thirty-two-bit devices are being well received, but faster and more

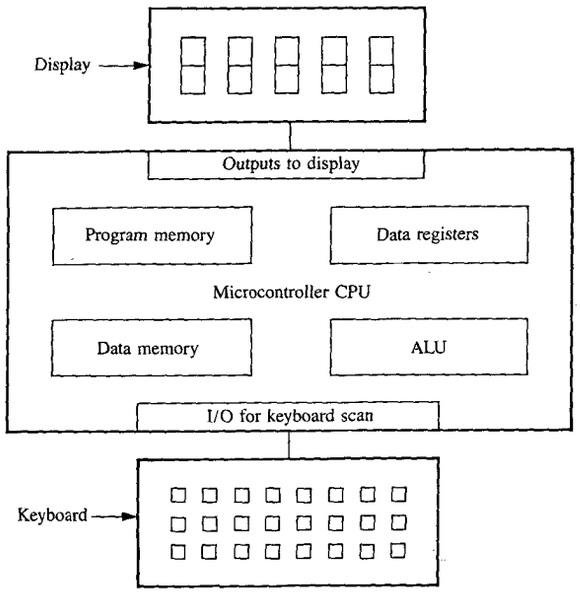

Figure 4

Cash register block diagram.

versatile eight- and sixteen-bit devices will remain as the backbone of the microprocessor industry for some time to come (at least until there is a revolutionary breakthrough in the technology required to interconnect multiple devices).

To state simply where microprocessors are going from here: They are becoming more powerful and complex in function, faster in operating speed, and smaller in size; they will soon be used everywhere.

Example 1: A cash register control

Our first example is a simple system using a microcontroller to operate a cash register. The cash register was chosen as an example because until recently it was a purely mechanical device, and its function is well understood. The marketplace today requires that mechanical devices be simplified or eliminated in order for a product to remain competitively priced. The use of a touch-pad keypad and LED (light-emitting diode) display eliminates the mechanically operated keys and price display of the traditional cash register. A microcontroller was chosen because the application is a high-volume one and unit cost is of prime concern. Also, the application itself is well suited to the capabilities of a microcontroller. The cash register, which is used by a fast-food chain, has a special keypad for entering items by name, and a five-digit display for showing the total price of an order, the amount tendered, and the change due. (See **Figure 4**.) As an added feature that would not have been possible with the old-style cash register, a running total of receipts is kept for transmitting to a host computer at the end of the day.

The microcontroller chosen for this application has 1K bytes of onboard masked ROM for storing the application program and 64 bytes of onboard RAM for data storage. For prototyping, there is a second version available that has UV-erasable EPROM in place of the ROM and is compatible pin for pin with the masked ROM version. There are three I/O ports of eight bits each that provide 24 bits of I/O directly on the CPU chip. There is an accumulator, an eight-bit timer, eight eight-bit working registers and an onboard stack that stores eight return addresses. No clock oscillator chip is required, as there are two pins on the CPU that accept a crystal directly as the timing element. There is also an interrupt pin which may be used to interrupt the CPU or which may be programmed to act as a polled input without actually interrupting the program in progress. Two "test" pins are available, one of which may be used either as an input or an output. A single five-volt power supply is required.

The LEDs for the display are standard seven-segment displays (so named because of the seven individual segments that are used to form the number 8). The simple approach would be to use displays that accept BCD numbers from 0 to 9 and display the corresponding digits. This approach would require 16 of our 24 I/O lines (four display digits at four bits each). We have chosen instead to use LED displays that provide separate inputs for each of the seven segments plus one for the period (or decimal point). This means that we need eight data lines going to each display digit; but as well as displaying numbers, we will be able to form words such as "Err.," "CASH," "DUE," etc. by selecting proper combinations of the segments. The LEDs need not be energized 100 percent of the time to remain visible, so we multiplex them by connecting all similar inputs on a bus. (All segment 1s are tied together, as are all segment 2s, all periods, etc.) We illuminate the displays by strobing power to them one at a time, using four more of our 24 I/O lines. Going from four to five digits of display would add the possibility of words such as "HELLO" and "Error" to our vocabulary and would still only require a total of 13 I/O lines for the display. By multiplexing the display LEDs, we have saved three I/O lines and added a display digit, as well as reducing the total amount of power consumed by the display. We have also added a level of difficulty to the programming. Instead of writing to the display digits and leaving the data there until they are to change, we must now continually change the data and select lines in the proper sequence in order to keep the display lit. If we multiplex more than five display digits and also have to watch for new inputs from the keypad, the display will start to get dim.

The keypad for the cash register is of the EDS (elastic diaphragm switch) type and essentially has no moving parts

to wear out. (It is also impervious to dirt and liquid spills.) There are 25 "keys," 22 of which are designated by the names of the products available. This reduces the possibility for operator error, as the inputs are by item name and the prices are automatically assigned by the microprocessor. If the prices change, a table in the program must be updated; if the items change, a new template may be placed over the keypad. The keys are arranged electrically in a 5 × 5 matrix and are strobed by ten of the I/O lines. We have chosen to have the depression of any key cause an interrupt so that we scan the keypad only when necessary.

As the total price for each transaction is displayed, it is also added to a running total for the day that is being kept in RAM. If desired, a count can also be kept for any or all of the various items being sold. (A total of 255 can be accumulated in one byte, but two bytes allows the daily total to reach 65 535.) At the close of business (when the phone rates happen to be cheaper), the cash register may be connected to a long-distance line through a modem (modulator/demodulator). The daily records may then be transmitted to the home office, where they may be used for bookkeeping and/or inventory control purposes.

Example 2: Control of a robot

For our second example we look at a more sophisticated system that uses a number of 16-bit microprocessors to control an industrial robot. The robot under consideration has eight axes or degrees of freedom. Each axis is driven by an electric motor and uses a microprocessor to control the speed of the motor. A ninth microprocessor acts as an overall supervisor to send data to the individual axis controllers, interface with the machine operator, and communicate with a host computer that oversees the operation of several robots. Each axis-control microprocessor is located on a separate printed-circuit card and is referred to as a slave. The master is located on a larger printed-circuit card, sends commands to the slaves, and also handles the miscellaneous robot control functions. **Figure 5** shows the basic components in the system. We look first at a typical slave card and then at the master.

Each slave card has a 16-bit microprocessor with its associated logic and memory. (Refer to **Figure 6**.) Two EPROM sockets allow for up to 4K bytes of program memory. Four 1K × 4 RAM modules provide 2K bytes of scratchpad RAM for storing program variables and a return address stack. There is an eight-bit DAC (digital-to-analog converter) to supply an analog (continuously variable) output used to control the speed of the axis drive motor. A 16-bit ADC (analog-to-digital converter) converts an analog input signal ranging from −10 volts to +10 volts into a 16-bit digital word representing the position of the robot axis in its range of travel. The microprocessor program on the slave card continually reads the axis position from the ADC. If the axis position does not match the last command position

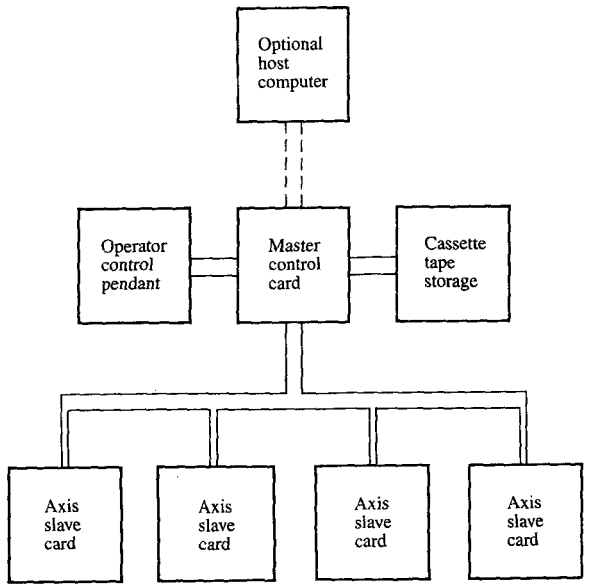

Figure 5

Robot simplified block diagram.

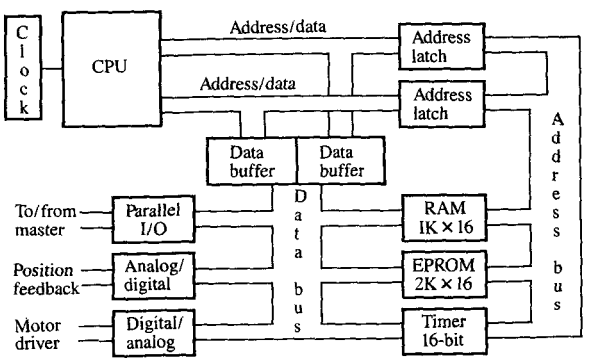

Figure 6

Slave card block diagram.

from the master, the microprocessor program causes the DAC to send a voltage to the axis drive motor until the actual position matches the commanded position.

The slave card does not really have much to keep track of (the commanded position from the master, the actual position from the ADC, and the voltage to send out to the drive motor). It must, however, operate at a high rate of speed because when the robot axis is moving at high speed, the ADC input changes rapidly and the motor signal must

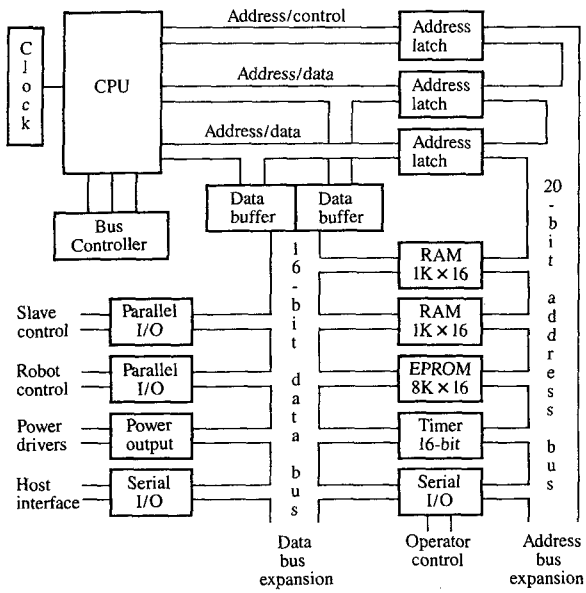

Figure 7

Master card block diagram.

be tuned to this for smooth operation of the robot. The portion of the program that handles the motor drive signal is quite concise and is written in assembly language for optimum use of execution time. The rest of the program handles communication with the master, system diagnostics, etc. and is not as critical for execution time or memory space. To save programming time, this portion of the slave program can be written in a higher-level language.

The master card is physically larger than a slave card and contains a 16-bit microprocessor with its own associated control logic and memory. **Figure 7** is a simplified block diagram of the master card. The two EPROM sockets on the master card are larger than those on the slave card and allow for up to 16K bytes of program memory. Eight 1K × 4 RAM modules provide 4K bytes of scratchpad RAM for storing data tables and a return address stack. Some additional control chips are used with the microprocessor on the master card to allow for a DMA operation with the RAM on the card and for the use of a numeric processor to perform high-level math functions. The application table that determines the various positions of each axis is stored in RAM on the master card and may be brought in slowly from a tape cassette or rapidly from another CPU by using DMA. The numeric data processor is used when the robot is not just moving from one point to another, as in a pick and place operation, but must maintain a specific path and speed, as in a welding operation. A high-level language can be used to write the program for the master card except for the routines that handle communications with the slaves. These routines determine the sequence of events on the communications bus and are written in assembly language.

The robot can be "taught" to perform an operation by leading it through the process steps with an operator control pendant. Each step is recorded in a table that is later followed by the machine as it performs the operation. This table is in RAM on the master card, so it may be changed at any time. The data are lost when power is turned off, so the information is stored by sending it out serially on a communication link to a tape drive or a host computer. This also allows the host computer to build up a table of robot operation steps and download it to the master card for execution.

A multi-microprocessor-based control for the robot was chosen as a cost-saving measure. The functions provided by the slave cards could have been implemented in conventional hardware, but the cost would have been greater because a large portion of the control function was eliminated from the hardware and performed with software. The functions provided by the master card could have been accomplished by a minicomputer. Here again there is a cost saving, because a minicomputer is stand-alone and general-purpose and the master card was designed specifically to control a multi-axis machine so that there is no unused hardware. Also, the master card is packaged on an 8 × 15-in. (20.32 × 38.10-cm) printed-circuit card inside the robot control unit and the minicomputer would be in a moderately sized box of its own.

Appendix 1: Evolution

The introduction of the transistor changed forever the physical aspects of computer circuitry. At first there was a constant effort to build smaller and smaller transistors until their size was reduced to the point where they became difficult to handle in manufacturing operations. Development effort was then directed toward placing as many transistors as possible on one piece of silicon, called a die or chip. This allowed circuit designers to create individual logic chips with ever-increasing levels of sophistication, and the resulting SSI (small-scale integration), MSI (medium-scale integration), and LSI (large-scale integration) random-logic devices were widely used by system designers in the proliferation of solid-state digital controls that were introduced in the late 1960s and early 1970s [1]. The point was then reached where LSI devices were becoming less general-purpose and more dedicated to one specific application. At this point the introduction of new LSI devices began to slow down due to the diminishing return on investment required to develop and introduce new devices.

This was the environment in which the first microprocessor, or "CPU on a chip," was born. The calculators being built at the time each used several logic chips containing up to 1000 transistors apiece. Busicom, a

Japanese manufacturer of calculators, approached Intel in mid-1969 with a request for 12 LSI chips for a new family of high-performance programmable printing calculators. Intel at the time was producing MOS (metal oxide semiconductor) and bipolar RAMs (random-access memories) and could place up to 2000 transistors on a single chip. The Busicom design required 3000 to 5000 transistors on each chip and packages with 36 to 40 leads each.

The logic to control the printer, scan the keyboard, and update the display, as well as to perform the actual calculations, was to be incorporated on separate chips. The elemental instructions for performing the calculations were quite complex and were executed more in random logic than in ROM (read-only memory). In an effort to simplify the design, it was decided to reduce the complexity of the elemental instructions and thus make them more general-purpose. With programs stored in ROM, these simple instructions could be used repetitively not only to perform the required calculations but also to perform the logic functions required for the overall operation of the calculator.

A four-bit-wide data path was chosen because it worked out well with the BCD (binary coded decimal) coding being used in calculators at the time and allowed addressing of up to 16 digit positions for display or printing. The final version of the calculator control consisted of a four-chip set that eventually was called the MCS-4. It consisted of a four-bit CPU chip, a ROM chip for program memory, a RAM chip for data memory, and a shift register chip for output expansion. The CPU chip ended up with about 2300 transistors on a 16-pin substrate, becoming the 4004 that was advertised as a "micro-programmable computer on a chip" in November of 1971. The term "microprocessor" was first used for this device in 1972.

The CPU chip contained a four-bit parallel adder, 16 index registers, an accumulator, and a push-down stack for return addresses. There were 46 instructions which included conditional branching, jump-to-subroutine, indirect fetching, and support for both binary and decimal arithmetic. The 4004 had a four-bit parallel bidirectional data bus that multiplexed address information and data. An eight-step instruction cycle was used to handle eight-bit instructions, and a 12-bit instruction address space allowed addressing of up to 4096 bytes of program memory.

The microprocessor was therefore born out of necessity as a natural development in the progression of logic design and was initially intended more for the replacement of random logic than as an attempt to put a computer on a single chip.

Late in 1969, Computer Terminals Corporation requested that Intel develop LSI chips for the registers of a new intelligent terminal. The 4004 instruction set had just been defined and Intel proposed to CTC that the terminal control logic be implemented on one chip as a complete eight-bit processor. This design was not used in the CTC terminal, but it was implemented in silicon and became the first parallel eight-bit microprocessor. The CPU was introduced by Intel early in 1972 as the 8008. It had 45 instructions oriented toward the terminal requirement of character string handling and also provided interrupt capability, a return address stack, and six general-purpose registers.

Late in 1972 Rockwell entered the microprocessor market with a four-bit parallel processor called the PPS-4. The CPU had 50 instructions and was implemented in a PMOS (p-channel metal oxide semiconductor) chip on a 42-pin substrate. Several other manufacturers introduced microprocessors in 1973 and 1974. Among these were Texas Instruments' TMS-1000 family of four-bit PMOS processors, RCA's CMOS (complementary metal oxide semiconductor) 1802, and National Semiconductor's IMP-16, which was a 16-bit five-chip bit-slice CPU.

The first three microprocessors marketed by Intel were implemented in PMOS. (There had also been a follow-on improvement to the 4004 called the 4040. It had 14 more instructions, a deeper stack, and more registers and memory addressing ability, as well as interrupt capability, all on a 24-pin substrate.) In 1974 Intel released an upgraded version of the 8008 and called it the 8080, the first of the second-generation microprocessors. The new processor consisted of 5000 transistors implemented in faster NMOS (n-channel metal oxide semiconductor) and had 30 more instructions than the 8008. The 8080 utilized a 40-pin package that made interfacing to it somewhat easier than with the 18-pin 8008, which required an additional 20 TTL (transistor-transistor logic) packages for memory and I/O interface. The 8080 had an eight-bit data bus and a 16-bit address bus, and the return-address stack was removed from the CPU and placed in RAM to allow virtually unlimited subroutine nesting. Decimal and double-precision arithmetic instructions missing from the 8008 were added to the instruction set, and the hardware provided the ability to handle vectored interrupts. (The interrupting device essentially tells the CPU where in program memory to find the proper interrupt service routine.)

Following the introduction of the 8080 in 1974, many new microprocessors have been introduced each year. (By 1976 the total was already up to 54.) The 6800 from Motorola in 1974 was the first to use a single +5-volt power supply, and Intel's 8048 in 1976 was the first eight-bit microprocessor to have a CPU, RAM, ROM, and I/O all on one chip.

Sixteen-bit microprocessors have been around in one form or another since 1974, but the first one to gain wide acceptance in the industry was the 8086 that Intel introduced in 1978. Texas Instruments' 9980, Intel's 8088, and Motorola's 6809 featured higher performance by utilizing 16-bit internal data processing, but allowed simpler interfacing to the rest of the system by retaining an eight-bit external data bus. The trend today is to go to wider data paths (16-bit microprocessors are in widespread use and 32-

bit microprocessors are being introduced). There is also a strong effort to place more of the overall system function on the CPU in the form of operating systems in silicon as well as auxiliary functions such as timers, interrupt controllers, communications controllers, etc., on the chip. Coprocessors are being used for high-level mathematical functions, such as floating-point operations, as well as intelligent I/O.

Some of the faster, special-purpose microprocessors have been implemented with bipolar transistors, but photolithographic and semiconductor processes have improved at a rapid rate and most of today's microprocessors still utilize NMOS in one form or another. There is an increasing trend, however, to utilize CMOS in order to improve noise immunity and reduce power consumption.

The first microprocessors were used mainly for the replacement of random logic, as in the case of the printing calculator and the intelligent terminal previously discussed. They continue to be widely used in this type of application and have also branched into two other main areas. One is that of reducing the physical size of actual computers, as is evidenced by the rapidly growing personal computer market. The other is the introduction of intelligence into areas where it was not previously possible or practical. Some of the more popular examples of this may be seen in electronic toys, arcade games, automobiles, household appliances, robotics, instrumentation, and the ever-present video games.

In the ten years since the introduction of the 8080 there have been improvements in speed and functionality, and instruction sets have been expanded to include more data-handling and program-control instructions, but the rudiments of the basic 8080 itself can still be seen in the majority of today's processors.

Appendix 2: Memory

A microprocessor system requires some type of memory for storing program instructions and also for storing constants and variable data. Instructions are often stored in ROM (read-only memory) and data are often stored in RAM (random access memory). The information stored in a ROM is not volatile; that is, it is retained even when all electric power to the device is turned off. RAM, on the other hand, retains its data only while it is connected to a voltage source. ROM and RAM each come in several different types, which we now explore.

ROM may have its contents established during the manufacturing process, in which case it is called "masked" ROM. The user must specify the desired contents of the memory to the ROM manufacturer, who creates a photolithographic mask with the correct bit pattern. The masking of the data into the ROM is an expensive and time-consuming process, but the finished parts themselves are relatively inexpensive. Masked ROM is therefore generally used in applications involving high-production items where many identical copies are required and where the program or data being stored are not likely to change.

Another type of ROM is the PROM (programmable read-only memory). The PROM is more expensive than masked ROM in large quantities, but it has an advantage in that it can be programmed by the manufacturer of the finished product in his plant and does not require that the information be essentially built into the parts by the semiconductor manufacturer, as is the case with the masked ROMs. The actual programming which loads the data or instructions into the PROMs is done on a device called a PROM programmer. PROM programmers are now available to meet a variety of needs. A simple and inexpensive one might be able to program a specific type of PROM or copy its contents into another one of the same type, where a more sophisticated and expensive model might be able to work with different types of PROMs and perhaps communicate with a terminal or a computer. Some microprocessor development systems have PROM programmers built in, so that after a program has been developed it can be loaded directly into a memory chip for debugging in the target system.

PROMs themselves come in several varieties. First is the "fusible-link" PROM that is personalized by selectively fusing metal links to change the bit pattern stored in the device. The memory chip as it is manufactured contains all zeros or all ones and the links are opened up one at a time to change the original data in one bit location at a time. This is a nonreversible process, so if changes must be made to the data, it generally means starting over again with a new PROM chip. Fusible-link PROMs do not typically have a large storage capacity and are used more in logic applications than for storing programs or data.

The second, and today the most widely used, type of PROM is the UV-erasable EPROM (erasable programmable read-only memory). As the name implies, a "UV-erasable" EPROM may be erased by a prolonged exposure to an ultraviolet light source. (There is a transparent window on the top of the package to allow the erasing light to reach the actual memory chip.) An EPROM may be programmed and erased many times before it finally fails to accept new data. The actual programming of the EPROM is done with a PROM programmer. In large quantities, the individual piece price of an EPROM is greater than for a masked ROM, but it does have a number of advantages. Because an EPROM is erasable, it may be used in prototypes while the final code is being developed for eventual entry into a masked ROM. EPROMs may be gang-programmed in quantity and are therefore suited for applications that require repeat build but do not see enough replication to justify the cost of a masked ROM. Also, in the case of upgrading a product with new software or expanded capability, an EPROM can be exchanged and the old one sent back to be reprogrammed with the new code.

A third type of PROM is the EEPROM, which stands for "electrically erasable programmable read-only memory." The EEPROM is similar to the UV-erasable EPROM except that it may be programmed and erased electrically while in place in a system and need not be removed for the UV light and PROM programmer process. An EEPROM is also referred to as an EAROM (electrically alterable read-only memory).

RAM memory is used to hold information that is subject to change during the normal operation of a system. RAM is available in two basic types known as "static" and "dynamic." In both cases, the information in the memory is lost when power is removed from the module and must be written back in each time power is applied. The difference between static and dynamic RAM is in how the data are actually stored in the memory chip. A static RAM has flip-flop-type storage locations for each bit that is to be stored and these flip-flops retain their information until it is written over or until power is removed from the device. In a dynamic RAM, the storage mechanism consists of a single transistor for each storage location. The information is stored as zeros or ones by charging or discharging a capacitor on the base of the storage transistor. Dynamic RAM is less expensive than static RAM, but the capacitors leak and must be repeatedly recharged. Circuitry known as "memory refresh logic" is required to keep recharging the proper capacitors. This is done automatically, but the logic takes up space on the circuit board and adds cycles to the process time of the system.

A newer type of RAM is the NOVRAM (nonvolatile random access memory). These devices contain an area of high-speed static RAM that is backed up on the same device by an identical array of EEPROM. The static RAM is used for normal read and write operations, and its contents may be transferred to the EEPROM very quickly if a power failure occurs. The data may then be retrieved from the EEPROM when the system is again operational. These devices provide the benefits of both normal high-speed read and write operations and nonvolatile storage in the same package.

"Byte-wide memory" is a packaging scheme that is becoming popular in microprocessor-based systems. This scheme uses identical package pinouts for the same sizes of RAM, ROM, and EPROM. This allows EPROM to be used in the development stage of a project, and when the program has been finalized and debugged, ROMs may be ordered that directly replace the EPROMs with no wiring changes required to the system. Another advantage of having identical pinouts is that printed-circuit cards may be fabricated with standard prewired memory sockets that have been assigned addresses in the memory map. The end user may then populate the card with the type of memory needed for the particular system at hand, whether it is RAM, ROM, or EPROM. In byte-wide memory systems, package pinout is also considered when different sizes of memory are utilized. Smaller memory modules may be plugged into sockets intended for larger-capacity memory. In this case, care must be used to ensure that the proper pins on the module interface with the corresponding socket pins because the socket, which is able to support a larger memory, has more pins than the module.

Memory modules are available in many different sizes and are generally categorized according to the total number of data bits that may be stored. (A 4K RAM, for instance, has 4096 individual bit positions.) The number of data lines on a memory chip is determined by the size of the words that may be stored, and the number of address lines is determined by the number of words. Ten address lines allow addressing of 1024 words and eight data lines allow for an eight-bit word size. A memory manufacturer may refer to this as an 8K or 8192-bit device, but in a microprocessor system it is more likely to be called a 1K × 8 memory (for the 1024 words of eight bits each). For larger word sizes, the address lines of several memory chips may be wired in parallel. For more words, the data lines and the lower-order address lines may each be wired in parallel and the higher-order address lines are used to select or activate the proper memory chip. System memory may thus be tailored to almost any application by the proper selection and interconnection of memory modules.

Appendix 3: Packaging

The overall package size of a microprocessor is determined not by its computing power or the amount of on-chip logic it contains, but rather by the number of lines needed to interface the microprocessor itself with the rest of the system it is controlling or operating in. A microprocessor CPU consists of a small square of silicon, referred to as a die or chip, which contains the necessary solid-state logic circuitry. This chip is mounted on a substrate that acts as a chip carrier and contains the physical interface to the rest of the system. A typical microprocessor chip is about 0.2 mm thick and 2 or 3 mm square. The chips are fabricated from silicon wafers that measure up to several inches in diameter. The wafers go through a number of photolithographic and chemical process steps before finally being diced into individual chips. Each chip or die is then mounted on a substrate that is suitable for handling and for final assembly onto a card or board.

Individual microprocessor package sizes vary, but a typical one today has 40 pins, measures 1.5 cm wide by 5 cm long, and has two parallel rows of 20 pins each. The double row of pins is evidenced in the name DIP (dual in-line package) that has been given to this particular type of package (also referred to as a DIL for "dual in-line"). The individual pins are spaced 0.1 in. (0.254 cm) apart and the rows are spaced 0.6 in. (1.524 cm) apart. The inputs and outputs on the chip itself consist of metal pads that are spaced only mils apart

around the periphery of the chip. Very fine wires are bonded to these pads to connect with metal traces on the substrate. These metal traces connect with the pins on the substrate, and the number of pins and traces needed is the determining factor in the size of the substrate. (In some of the newer, more sophisticated designs, especially those with 32-bit architectures, the size of the silicon chip is also becoming a determining factor.) The substrates may be made of ceramic or hard plastic and are hermetically sealed to prevent contaminants from contacting the silicon chip or interface wiring. The DIP packages are often referred to as modules and, to add to the confusion, quite often are loosely referred to as chips.

Some of the newer microprocessors are getting away from the DIP packaging in order to save printed-circuit board space and have gone to a square substrate that has rows of pins fastened to the bottom of the substrate. This type of package is called a pin grid array. There are also leadless chip carriers, which are square ceramic substrates that have no protruding pins. Connections are made to this device by contacting metal pads located around the periphery of the substrate in a process referred to as "surface-mount technology," or SMT. (Leadless chip carriers are sometimes imprecisely referred to as flat-packs.) Another method of fastening substrates to printed-circuit boards is via J-shaped leads that allow for different thermal coefficients of expansion for the substrate and the board.

SSI, *MSI*, and *LSI* are terms that relate to the amount of logic or number of "equivalent gates" that are present on one chip. A chip having fewer than ten equivalent logic gates is referred to as SSI. MSI denotes chips with between ten and 100 gates, and anything over 100 gates is known as LSI. VLSI is sometimes used for chips containing over 1000 gates in equivalent logic.

There are many names for the technology used in placing digital logic on semiconductor chips, but most are variations of bipolar and MOS. Bipolar transistors are typically fast and use more power than the field-effect transistors that make up the MOS devices. PMOS transistors use holes as majority carriers and are thus slower than the more popular NMOS transistors that use electrons as the majority carriers. CMOS chips utilize both NMOS and PMOS transistors on the same silicon substrate with the result being a device that uses very little power and has a high immunity to electrical noise.

Acknowledgments

The author wishes to thank Rex Dixon, Ray Floyd, Ed Galli, and Will Tracz for their support, encouragement, and suggestions in the preparation of this paper.

Reference

1. Marcian E. Hoff, Jr. and Robert N. Noyce, "A History of Microprocessor Development at Intel," *IEEE Micro* **1**, 8–11, 13–21 (February 1981). Note: This article contains a wealth of information concerning the environment into which the microprocessor was born; it includes history on all early publicized microprocessors, not just those developed by Intel. This article has been reprinted by Intel as "AR-173" (available from Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051).

Bibliography

This bibliography has been selected as a small sampling from the author's personal library in an attempt to touch upon all areas of the broad field of microprocessors. There are selections specifically to cover history, and some on technical details, but the majority were selected with the tutorial theme of this paper in mind. A few references to basic computer architecture have been included as a bridge from computers to microcomputers to microprocessors, but programming references have been eliminated except for those directly applicable specifically to microprocessors. The bridge from digital discrete logic is covered quite thoroughly in the early literature on microprocessors. The listing is far from being complete or exhaustive, but it should contain material that will be helpful to readers of virtually any background. The arrangement is chronological, with the titles preceding the names of the authors. The earlier entries show how the industry got started in a particular area, and the more recent ones describe the latest developments. A number of books are included as overviews, along with a sampling of manufacturers' data books for exacting details. The main thrust, however, is on articles and papers, as they are generally short and to the point in a particular area, and one may scan the titles for subjects of particular interest.

1971–1973

"Microprogramming Handbook," 2nd ed., Microdata Corporation, 644 E. Young St., Santa Ana, CA, November 1971.

"MOS Integrated Circuits," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, February 1972.

"Microprogrammable Arithmetic Processor System—Orientation," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, May 1972.

"MCS-8 Microcomputer Set—8008 Users Manual," Rev. 4, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, November 1973.

1974

"IMP-16C Application Manual," *Publication No. 4200021C*, National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, January 1974.

"MCS-4 Microcomputer Set Users Manual," Rev. 5, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, March 1974.

"MPS Microprocessor Series Users Handbook," *Manual No. DEC-08-UMPHA-A-PA*, Digital Equipment Corporation, Maynard, MA 01754, July 1974.

"Microprocessor Design Series" (four reprints from *EDN* magazine, Vols. 18 and 19), R. H. Cushman (offered by *Design News* magazine, July 22, 1974).

"Distributed Function Microprocessor Architecture," A. J. Weissberger, *Computer Design* **13**, 77–83 (November 1974).

"PACE Users Manual," *Order No. IPC-16P/968X*, National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, December 1974.

1975

"An Introduction to Microcomputers," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1975.

"ALTAIR 8800 Theory of Operation Manual and Schematics," Micro Instrumentation and Telemetry Systems, 2450 Alamo SE, Albuquerque, NM 87106, 1975.

"F8 Circuit Data Book," Fairchild Semiconductor, Fairchild Camera and Instrument Corporation, 464 Ellis St., Mountain View, CA 94042, 1975.

"How to Select and Use Microprocessors," *Pro-Log Microprocessor Users Guide*, Pro-Log Corporation, 2411 Garden Rd., Monterey, CA 93940, 1975.

"LSI-11/PDP-11/03 Processor Handbook," Digital Equipment Corporation, Maynard, MA 01754, 1975.

"Microprocessors," L. Altman, Ed., McGraw-Hill Book Co., Inc., New York, 1971.

"M6800 Microprocessor Applications Manual," Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1975.

"M6800 Microprocessor Programming Manual," Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1975.

"Series 1600 Microprocessor System," General Instrument Corporation Microelectronics, 600 W. John St., Hicksville, NY 11802, 1975.

"Series 3000 Reference Manual," *Manual No. 98-221A*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1975.

"The Bugbook III—Microcomputer Interfacing," P. R. Rony, D. G. Larsen, and J. A. Titus, E & L Instruments Inc., 61 First St., Derby, CT 06418, 1975.

"TMS 1000 Series MOS/LSI One-Chip Microcomputers," Texas Instruments Inc., P. O. Box 225012, MS-54, Dallas, TX 75265, 1975.

"μC Systems Directory" (fold-out wall chart), R. M. Grossman, *EDN* **20** (1975).

"Understanding Microprocessors," D. Queyssac, published by Unwin Brothers Ltd., Old Woking, Surrey (undated); distributed by Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721.

"990 Computer Family Systems Handbook," *Manual No. 945250-9701*, Texas Instruments Inc., P. O. Box 225012, MS-54, Dallas, TX 75265, 1975.

"8080 Microcomputer Systems User's Manual," *Order No. 98-153C*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, September 1975.

1976

"An Introduction to Microcomputers—Volume I—Basic Concepts," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1976.

"An Introduction to Microcomputers—Volume II—Some Real Products," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1976.

"Dr. Dobbs' Journal of Computer Calisthenics and Orthodontia—Running Light Without Overbyte," Volume 1 (1976—entire year), J. C. Warren, Jr., Ed., People's Computer Company, P. O. Box E, Menlo Park, CA 94025, 1976.

"F8 User's Guide," *Manual No. 67095665*, Fairchild Micro Systems, Fairchild Camera and Instrument Corporation, 464 Ellis St., Mountain View, CA 94042, 1976.

"Introduction to Microcomputers and Microprocessors," A. Barna and D. I. Porat, John Wiley & Sons, Inc., New York, 1976.

"Microcomputer Design," D. P. Martin, Martin Research, 3336 Commercial Ave., Northbrook, IL 60062, 1976.

"Microcomputers/Microprocessors: Hardware, Software, and Applications," J. L. Hilburn and P. N. Julich, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976.

"Microprocessors and Minicomputers," B. Soucek, John Wiley & Sons, Inc., New York, 1976.

"M10800 High Performance MECL LSI Processor Family," Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1976.

"Scelbi '6800' Software Gourmet Guide and Cook Book," R. Findley, Scelbi Computer Consulting, Inc., 1322 Boston Post Road Rear, Milford, CT 06460, 1976.

"Scelbi '8080' Software Gourmet Guide and Cook Book," R. Findley, Scelbi Computer Consulting, Inc., 1322 Boston Post Road Rear, Milford, CT 06460, 1976.

"Software Design for Microprocessors," J. G. Wester and W. D. Simpson, Texas Instruments Learning Center, Texas Instruments Inc., P. O. Box 225012, MS-54, Dallas, TX 75265, 1976.

"The ia7301 Computer in a Book," D. Guzeman, IASIS, Inc., 815 W. Maude Ave., Sunnyvale, CA 94086, 1976.

"8080 Programming for Logic Design," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1976.

"MCS6500 Microcomputer Family Programming Manual," *Publication No. 6500-50A*, MOS Technology, Inc., Valley Forge Corporate Center, Norristown, PA 19401, January 1976.

"Microprocessor-Controlled Video Game Can Be Adjusted to Player's Skill," S. Davis, *Electronic Engineering Times*, pp. 25–27 (April 12, 1976).

"Microprocessor Basics" (25-part series), *Electronic Design* **24–27** (April 26, 1976, through June 7, 1979).

"16-Bit Processor Performs Like Minicomputer," A. Lofthus and D. Ogden, *Electronics* **49** (May 27, 1976).

"When Programming Microprocessors, Use Your Hardware Background," E. Lee, *Electronics* **49** (July 8, 1976).

"A Microcomputer Tailored for Multiprocessor Control Applications," G. Adams, T. Morgan, and J. Zarrella, *Control Engineering* **23**, 58–60 (September 1976).

"'Universal' Development System Is Aim of Master-Slave Processors," R. D. Catterton and G. S. Casilli, *Electronics* **49** (September 16, 1976).

"EDN μC Design Course," C. A. Ogdin, *EDN* **21**, 127–316 (November 20, 1976).

"An Introduction to Microcomputer Software," M. G. Leonard, *Machine Design* **48**, 70–76 (November 25, 1976).

"Putting a Microcomputer on a Single Chip," H. A. Raphael, *Computer Design* **15**, 59–65 (December 1976).

"TMS 1000 Series Data Manual," Texas Instruments Inc., P. O. Box 225012, MS-54, Dallas, TX 75265, December 1976.

1977

"An Introduction to Microcomputers—Volume 0—The Beginner's Book," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1977.

"Evaluation Kit Manual for the RCA CDP1802 COSMAC Microprocessors," *Manual No. MPM-203A*, RCA Solid State, 1998 Springdale Rd., Cherry Hill, NJ 07066, 1977.

"How to Program Microcomputers," W. Barden, Jr., Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1977.

"Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing," D. G. Larsen, P. R. Rony, and J. A. Titus, Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1977.

"Microcomputer Handbook," C. J. Sippl, Petrocelli/Charter, New York, 1977.

"Microcomputer-Based Design," J. B. Peatman, McGraw-Hill Book Co., Inc., New York, 1977.

"Microprocessor Applications in Business, Science and Industry" (42 article reprints from a number of magazines), National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1977.

"Microprocessor Architecture," M. Biewer, Pro-Log Corporation, 2411 Garden Rd., Monterey, CA 93940, undated.

"Microprocessor Systems Handbook," Dr. D. P. Burton and Dr. A. L. Dexter, Analog Devices, Inc., P. O. Box 796, Norwood, MA 02062, 1977.

"Microprocessors and Microcomputer Systems," D. H. Sawin III, Lexington Books, D. C. Heath and Co., Lexington, MA, 1977.

"Microprocessors in Systems," R. H. Fish III, System Insights, P. O. Box 1, Austin, TX 78767, 1977.

"Parallel Processing System (PPS) Microcomputers," Rockwell International Microelectronic Devices, P. O. Box 3669, Anaheim, CA 92803, 1977.

"Practical Microcomputer Programming: The M6800," W. J. Weller, Northern Technology Books, P. O. Box 62, Evanston, IL 60204, 1977.

"The 8080A Bugbook—Microcomputer Interfacing and Programming," P. R. Rony, D. G. Larsen, and J. A. Titus, Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1977.

"TMS9940 Single Chip Microcomputer," J. D. Bryant and R. Longley (for presentation at Electro '77), Texas Instruments Inc., P. O. Box 225012, MS-54, Dallas, TX 75265, 1977.

"Z80-Assembly Language Programming Manual," Zilog, Inc., 1315 Dell Ave., Campbell, CA 95008, 1977.

"6800 Programming for Logic Design," A. Osborne, Adam Osborne and Associates, Inc., P. O. Box 2036, Berkeley, CA 94702, 1977.

"A Logic State Analyzer for Microprocessor Systems," J. H. Smith, *Hewlett-Packard Journal*, pp. 2–11, 1820 Embarcadero Rd., Palo Alto, CA 94303 (January 1977).

"Firmware for a Microprocessor Analyzer," T. A. Saponas, *Hewlett-Packard Journal*, pp. 12–15, 1820 Embarcadero Rd., Palo Alto, CA 94303 (January 1977).

"Software for Microprocessors" (seven-part series), *Electronic Design* **25** (January 4, 1977 through June 7, 1977).

"Taking the Mystery out of Micros," P. Roybal, *Machine Design* **49**, 80–83 (March 24, 1977).

"Third-Generation Microcomputer Set Packs It All into 3 Chips," D. W. Sohn and A. Volk, *Electronics* **50**, 109–113 (May 12, 1977).

"Analysis of Multiple-Microprocessor System Architectures," A. J. Weissberger, *Computer Design* **16**, 151–163 (June 1977).

"16-Bit Microcomputer Is Seeking a Big Bite of Low-Cost Controller Tasks," J. D. Bryant, *Electronics* **50** (June 23, 1977).

"Unraveling the Mystery in User Microprogramming," R. Frankenberg, Part 1, *Mini-Micro Systems* **10**, 28–33 (June 1977); Part 2, *Mini-Micro Systems* **10**, 46–50 (July 1977); Part 3, *Mini-Micro Systems* **10**, 54–60 (September 1977).

"EDN Software Design Course," C. A. Ogdin, *EDN* **22**, 67–200 (June 5, 1977).

"Microprocessor Networks," W. L. Spetz, *Computer*, pp. 64–70 (July 1977).

"The First of the Third Generation Microcomputers," L. Goss, *Solid State Technology*, pp. 42–45 (July 1977).

"Before Buying a Micro, Read the Menu," M. Biewer, *Machine Design* **49**, 68–74 (July 7, 1977).

"A Microcomputer Course for Professionals," G. K. Holt, *Mini-Micro Systems* **10**, 36–40 (September 1977).

"Acquiring Microcomputer Skills," C. A. Ogdin, *Mini-Micro Systems* **10**, 42–48 (September 1977).

"Comparisons and Trends in Microprocessor Architecture," E. E. Klingman, *Computer Design* **16**, 83–91 (September 1977).

"Microelectronics" (reprint of eleven articles from *Scientific American*), W. H. Freeman and Co., 660 Market St., San Francisco, CA 94104, September 1977.

"Making the Transition to Micros," C. A. Ogdin, *Mini-Micro Systems* **10**, 32–37 (October 1977).

"8X300 Reference Manual," Signetics Corporation, 811 E. Arques Ave., P.O. Box 409, Sunnyvale, CA 94086, October 1977.

"EDN System Design Project," J. Conway, *EDN* **22**, 133–233 (November 20, 1977).

"Microcomputer Overview" (eight chapters), C. A. Ogdin, *Mini-Micro Systems* **10**, 32–127 (November–December 1977).

"Is There a Tiger in Your Microcomputer?" *Electronic Business* **3** 24–27 (December 1977).

"Wrist Instrument Opens New Dimension in Personal Information," A. F. Marion, E. A. Heinsen, R. Chin, and B. E. Helmso, *Hewlett-Packard Journal*, pp. 2–10, 1820 Embarcadero Rd., Palo Alto, CA 94303 (December 1977).

"Microprocessor-Based Video Games," K. Li and A. Goldberger, *Electronic Design* **25**, 84–87 (December 6, 1977).

"Stripping the Mystery from Microcomputers," L. Teschler, *Machine Design* **49**, 161–170 (December 8, 1977).

1978

"A Guide to PL/M Programming for Microcomputer Applications," D. D. McCraken, Addison-Wesley Publishing Co., Reading, MA, 1978.

"A Microprocessor Course," M. E. Fohl, Petrocelli Books, Inc., New York/Princeton, 1978.

"Memory Design: Microcomputers to Mainframes," L. Altman, McGraw-Hill Book Co., Inc., New York, 1978.

"Microcomputer 3870/F8 Data Book," *Publication No. 79602*, Mostek Corporation, 1215 W. Crosby Rd., Carrollton, TX 75006, 1978.

"Practical Microcomputer Programming: The Z80," W. J. Weller, Northern Technology Books, P.O. Box 62, Evanston, IL 60204, 1978.

"Series 8000 Microprocessor Family Handbook," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1978.

"The Bugbook VI—Introductory Experiments in Digital Electronics, 8080A Microcomputer Programming and 8080A Microcomputer Interfacing," D. G. Larsen, P. R. Rony, and J. A. Titus, E & L Instruments Inc., 61 First St., Derby, CT 06418, 1978.

"The Complete Motorola Microcomputer Data Library," Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1978.

"The Motorola MC3870 User's Manual," *Manual No. M3870UM (AD)*, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1978.

"The Z-80 Microcomputer Handbook," W. Barden, Jr., Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1978.

"Understanding Digital Computers," F. M. Mims III, *Catalog No. 62-2027*, Radio Shack (div. of Tandy Corporation), Fort Worth, TX 76102, 1978.

"Z80 Instruction Handbook," N. Wadsworth, SCELBI Publications, Scelbi Computer Consulting Inc., P.O. Box 133, PP STN, Milford, CT 06460, 1978.

"Z80 Programming for Logic Design," A. Osborne, J. Kane, R. Rector, and S. Jacobson, Adam Osborne and Associates, Inc., P.O. Box 2036, Berkeley, CA 94702, 1978.

"8080/8085 Software Design with 190 Software Solutions," C. A. Titus, Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1978.

"9900 Family Systems Design and Data Book," W. D. Simpson, G. Luecke, D. L. Cannon, Ph.D., and D. H. Clemens, Texas Instruments Learning Center, Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, 1978.

"Handling Multilevel Subroutines and Interrupts in Microcomputers," J. F. Vittera, *Computer Design* **17,** 109–115 (January 1978).

"Using Microprogramming to Standardize Minicomputer Interfacing," F. Grothman and J. Washburn, *Digital Design* **8,** 27–36 (February 1978).

"Microcomputer Basics" (six-part series), *Electronic Design* **26, 27** (February 1, 1978, through January 18, 1979).

"Microprocessors Extend Scope of Automated Manufacturing," N. Sohrabji, *EDN* **23,** 101–106 (March 5, 1978).

"Bit-Slice Microprogramming Saves Software Compatibility," S. Y. Lau, *EDN* **23,** 42–46 (March 5, 1978).

"ICS Staff Goes Several Rounds with Microcomputers," J. Hall, J. Hickey, and R. Kuhfeld, *Instruments & Control Systems* **51,** 49–54 (April 1978).

"ICS Staff Goes Several Rounds with Microcomputers—Part II," J. Hall, R. Merritt, B. Reynolds, and L. Zerone, *Instruments & Control Systems* **51,** 73–80 (May 1978).

"Designing a μC Test Unit," W. W. Moyer, *Digital Design* **8,** 112–120 (May 1978).

"Unified Buses Make the Peripheral IC/μC Connection," Dr. M. Smolin, D. Graves, K. Winter, and M. Schwartz, *Digital Design* **8,** 34–44 (May 1978).

"The Intel 8086 Microprocessor: A 16-Bit Evolution of the 8080," S. P. Morse, W. B. Pohlman, and B. W. Ravenel, *Computer*, pp. 18–27 (June 1978).

"Triple-Bus Architecture," J. Johnson, C. Kinnie, and M. Maerz, *Electronic Design* **26** (July 19, 1978).

"R6500 Microcomputer System Hardware Manual," *Document No. 29650 N31*, Rockwell International Microelectronic Devices, P.O. Box 3669, Anaheim, CA 92803, August 1978.

"Complex Systems Are Simple to Design with the MC68000 16-Bit μP," I. LeMair and R. Nobis, *Electronic Design* **26,** 100–107 (September 1, 1978).

"Get Minicomputer Features at Ten Times the 8080 Speed with the 8086," G. Alexy and H. Kop, *Electronic Design* **26,** 60–66 (September 27, 1978).

"Structured Programming Formulates μP Program Logic," Dr. L. A. Leventhal, *Digital Design* **8,** 30–40 (October 1978).

"EDN Software Systems Design Course," J. Hemenway, *EDN* **23,** 251–312 (November 20, 1978).

"Cars and Computers Come Together," C. E. Wise, *Machine Design* **50,** 24–30 (November 23, 1978).

"Second-Generation Microcontrollers Take On Dedicated-Function Tasks," J. Beaston and B. Schillhammer, *Electronics* **51,** 127–132 (November 23, 1978).

"Integrating Peripherals into Processing Systems," R. J. Eufinger, *Computer Design* **17,** 77–83 (December 1978).

"The Microprocessor Popularity Race," C. A. Ogdin, *Mini-Micro Systems* **11,** 58–66 (December 1978).

1979

"Basic Microprocessors and the 6800," R. Bishop, Hayden Book Co., Inc., Rochelle Park, NJ, 1979.

"Microprocessor Software" (design-discipline reprint of 13 articles from *Machine Design*), Penton/PC, Penton Plaza, Cleveland, OH 44114, 1979.

"Microprogramming Techniques with Sample Programs," S. J. Evans, Reston Publishing Co., Inc., Reston, VA, 1979.

"Comparing Microprocessor Architectures," K. Rothmuller, *Mini-Micro Systems* **12,** 74–79 (January 1979).

"Parallel Processor Architectures—Part 1: General Purpose Systems," K. J. Thurber, *Computer Design* **18,** 89–97 (January 1979).

"Are Single-Chip Microcomputers the Universal Logic of the 1980s?" R. H. Cushman, *EDN* **24,** 83–89 (January 5, 1979).

"As You Get to Know the 8086, Use Your 8-Bit Expertise," J. Hemenway and E. Teja, *EDN* **24,** 81–87 (January 20, 1979).

"Parallel Processor Architectures—Part 2: Special Purpose Systems," K. J. Thurber, *Computer Design* **18,** 103–114 (February 1979).

"Comparison of Selected Array Processor Architectures," S. P. Hufnagel, *Computer Design* **18,** 151–158 (March 1979).

"New Options from Big Chips," J. Bayliss and J. McKevitt, *IEEE Spectrum* **16,** 28–34 (March 1979).

"Troubleshooting Microprocessors with a Logic Analyzer System," R. Lorentzen, *Computer Design* **18,** 160–164 (March 1979).

"Single-Chip N-MOS Microcomputer Processes Signals in Real Time," M. E. Hoff and M. Townsend, *Electronics* **52,** 105–110 (March 1, 1979).

"CPU Brings 16-Bit Performance to 8-Bit Systems," J. Bartlett and R. Retter, *Electronic Design* **27,** 76–80 (March 15, 1979).

"Coming: New Generations of Microcomputers," L. Teschler, *Machine Design* **51,** 108–114 (March 22, 1979).

"A Microcomputer Industrial Control Interface Using I/O Modules," *Manual No. MP716*, Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, April 1979.

"Designing a Microprocessor Driven Multipurpose Peripheral Controller," R. F. Binder, *Computer Design* **18,** 83–91 (April 1979).

"Designing the LSI-11/23," G. Dulaney, *Mini-Micro Systems* **12,** 55–60 (April 1979).

"Processor Architecture Anticipates Future Performance Requirements," R. E. Birney, *Computer Design* **18,** 71–79 (April 1979).

"Software: Micros vs. Minis," K. Schroeder, *Digital Design* **9,** 20–26 (April 1979).

"μP Selection—Some Do's and Don'ts," P. Snigier, Part 1, *Digital Design* **9,** 28–32 (April 1979); Part 2, *Digital Design* **9,** 28–34 (May 1979).

"How to Design Single Chip Microcomputers into Control Systems," W. Bottari, *Control Engineering* **26,** 69–72 (May 1979).

"Micro Power Microelectronic Devices Data Catalog," Rockwell International Microelectronic Devices, P.O. Box 3669, Anaheim, CA 92803, May 1979.

"Setting Up a Microprocessor Development Lab," B. Schweber, *Digital Design* **9,** 42–49 (May 1979).

"Standard Modules Offer Flexible Microprocessor System Design," A. D. Hirschman, G. Ali, and R. Swan, *Computer Design* **18,** 181–189 (May 1979).

"Comparing Architectures of Three 16-Bit Microprocessors," H. A. Davis, *Computer Design* **18,** 91–100 (July 1979).

"How to Select the Optimum Microprocessor for Consumer and Industrial Controls," R. T. Barck, *Control Engineering* **26,** 106–110 (July 1979).

"Microprocessor User's Guide," Pro-Log Corporation, 2411 Garden Rd., Monterey, CA 93940, July 1979.

"A Preview of the Motorola 68000," A. I. Halsema, *BYTE* **4,** 170–174 (August 1979).

"MC68000 Design Module User's Guide," *Manual No. MEX68KDM (D2)*, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, August 1979.

"Wizards of Silicon Valley," G. Bylinsky and Z. Lane, *OMNI* (August 1979).

"Zilog Z8000 Family Technical Overview," Zilog, Inc., 1315 Dell Ave., Campbell, CA 95008, August 1979.

"16-Bit Microprocessors," S. Davis, *EDN* **24,** 70–85 (August 5, 1979).

"MC6809 Course Notes," Motorola Technical Training, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, September 1979.

"Single-Chip 6801 Offers Versatility," J. J. Farrell III, Part 1, *Digital Design* **9,** 62–71 (September 1979); Part 2, *Digital Design* **9,** 42–52 (October 1979).

"70-Series Microprocessor Users Manual," *Manual No. μPG-000001*, National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, September 1979.

"Microprocessor Lab Teaches Operation and Troubleshooting," B. Bronson and M. Slater, *Hewlett-Packard Journal*, pp. 3–8, 1820 Embarcadero Rd., Palo Alto, CA 94303 (October 1979).

"The 8086 Family User's Manual," *Manual No. 9800722-03*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, October 1979.

"EDN Advanced Software Systems Design Course," J. Hemenway and E. Teja, *EDN* **24,** 293–336 (October 20, 1979).

"Learn to Apply the Power of the Z8002 by Studying a Small 16-Bit Computer," R. Korody and P. Alfke, *Electronic Design* **27,** 90–96 (October 25, 1979).

"The Intel 8086," S. Ciarcia, *BYTE* **4,** 14–24 (November 1979).

"Use a Systematic Procedure to Evaluate New μPs," J. Hemenway, *EDN* **24,** 185–193 (November 20, 1979).

1980

"Crash Course in Microcomputers," L. E. Frenzel, Jr., Howard W. Sams & Co., Inc., 4300 W. 62nd St., Indianapolis, IN 46268, 1980.

"MC68000 Cassette Training Tape," Motorola Technical Training, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, undated.

"MC68000 Course Notes," Motorola Technical Training, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, January 1980.

"MC68000 16-Bit Microprocessor User's Manual," *Manual No. MC68000UM (AD2)*, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1980.

"Microelectronics Product Guide," General Instrument Corporation Microelectronics, 600 W. John St., Hicksville, NY 11802, 1980.

"Microprocessor Basics" (design-discipline reprint of 11 articles from *Machine Design*), Penton/PC, Penton Plaza, Cleveland, OH 44114, 1980.

"The NS16000 Family of 16-Bit Microprocessors," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1980.

"The 8086 Book—Includes the 8088," R. Rector and G. Alexy, OSBORNE/McGraw-Hill, 630 Bancroft Way, Berkeley, CA 94710, 1980.

"The 8086 Primer—An Introduction to Its Architecture, System Design, and Programming," S. P. Morse, Hayden Book Co., Inc., Rochelle Park, NJ, 1980.

"Primer on Microprocessor Development Systems," G. Nadler, *Electronic Products* **23,** 54–57 (January 1980).

"Realtime Analyzer Aids Hardware/Software Integration," R. Francis and R. Teitzel, *Computer Design* **19,** 140–150 (January 1980).

"Logic-State Analyzers Seek Out μP-System Faults," G. Brock, *EDN* **25,** 137–140 (January 5, 1980).

"The Promise of Analog μPs: Low-Cost Digital Signal Handling," R. H. Cushman, *EDN* **25,** 127–132 (January 5, 1980).

"Forum on Testing Microprocessors," A. Mendelsohn, *Electronic Products* **23,** 35–40 (February 1980).

"Microprocessor Troubleshooting Techniques," D. Wiseman, *Electronics Test* **3,** 42–48 (February 1980).

"Communications in Distributed Systems—Part 1: Interfacing Techniques," M. G. Gable, *Computer Design* **19,** 30–34 (February 1980); "Part 2: Common Bus and Shared Resource Access Schemes," *Computer Design* **19,** 14–27 (March 1980); "Part 3: Communication Protocols and System Design Considerations," *Computer Design* **19,** 14–22 (April 1980).

"To Get to Know Analog μPs, Simulate Simple Examples," R. H. Cushman, *EDN* **25,** 137–146 (February 5, 1980).

"The MC68000—A 32-Bit μP Masquerading as a 16-Bit Device," R. Grappel and J. Hemenway, *EDN* **25,** 127–134 (February 20, 1980).

"Meeting EPROM Requirements of Advanced Microprocessors," T. Coffman, *Computer Design* **19,** 212–220 (March 1980).

"Designer's Guide to: Testing and Troubleshooting μP-Based Products," M. J. Weisberg, *EDN* **25,** 175–214 (four parts) (March 20, 1980).

"Bit-Slice Design Approaches," H. Brineen, *Computer Design* **19,** 184–191 (April 1980).

"Fiber Optics Successfully Links Microcomputers," S. Evans and J. Herman, *Digital Design* **10,** 36–37 (April 1980).

"Memory-Management Units Help 16-Bit μPs to Handle Large Memory Systems," J. Hu, H. Yonezawa, and B. Pueto, *Electronic Design* **28,** 128–135 (April 26, 1980).

"Microcontroller Doubles as Boolean Processor," B. Koehler, *Electronic Design* **28,** 57–62 (May 24, 1980).

"μP-Controlled 'House of the Future' Serves as a Product-Development Lab," W. Twaddell, *EDN* **25,** 65–74 (June 20, 1980).

"Pick a Computer Language That Fits the Job," M. Schindler, *Electronic Design* **28,** 62–78 (July 19, 1980).

"Indexed Mapping Extends Microprocessor Addressing Range," I. LeMair, *Computer Design* **19,** 111–118 (August 1980).

"Word Processing System Design for High Throughput," P. D. Cherry, *Computer Design* **19,** 95–99 (August 1980).

"μP-Based Product Design Starts with μP Selection," M. Mihalik and H. Johnson, *Electronic Design* **28,** 119–125 (September 1, 1980).

"Compare the Newest 16-Bit μPs to Evaluate Their Potential," R. Grappel and J. Hemenway, *EDN* **25,** 197–201 (September 5, 1980).

"Microcomputer Development Systems," A. Santoni, *EDN* **25,** 141–151 (September 5, 1980).

"Development of Microprocessor Software," M. Rooney, *Design News* **36,** 81–89 (November 17, 1980).

1981

"An Introduction to ASM86," *Order No. 121689-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1981.

"iAPX 432 General Data Processor Architecture Reference Manual," P. Tyner, *Manual No. 171860-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1981.

"Microprocessors Data Manual," *Manual No. DL-120*, Motorola Semiconductor Products Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721, 1981.

"NSC800 Microprocessor Family Handbook," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1981.

"STD Methods for Modularity," R. Born (presented at Electro 1981), Pro-Log Corporation, 2411 Garden Rd., Monterey, CA 93940, 1981.

"TMS9900 16-Bit Microprocessor Family," *Manual No. CL-483A*, Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, 1981.

"Requirements for High-Performance Microcomputers," J. Gorin and L. Stern, *Mini-Micro Systems* **14,** 127–136 (March 1981).

"Multitasking Executive Speeds 16-Bit Micros," J. M. Irwin, *Electronic Design* **29,** 131–135 (March 5, 1981).

"Optimizing Microprocessor Input/Output Techniques," L. E. Costlow, *Computer Design* **20,** 151–160 (April 1981).

"A Tale of Four μPs: Benchmarks Quantify Performance," R. D. Grappel and J. E. Hemenway, *EDN* **26,** 179–265 (April 1, 1981).

"Built-In Test Capabilities Could Cure μP-Based System Ills," D. Jones, *EDN* **26,** 105–109 (April 15, 1981).

"Distributed Intelligence vs. Centralized Logic," P. L. Alker, *Mini-Micro Systems* **14,** 103–115 (May 1981).

"iAPX 186 Microprocessor Architecture Overview," J. Klovstad and S. Kopel, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, May 1981.

"The Impact of 16-Bit Microprocessors on Software Development Tools," M. J. Elmore, D. Miller, and J. Schwabe, *Computer Design* **20,** 111–115 (June 1981).

"Understand Emulator Use to Increase Prototyping Skills," M. Mihalik and B. Francis, *EDN* **26,** 121–128 (June 10, 1981).

"Forum on 8 and 16-Bit Microprocessors," A. Mendelsohn, *Electronic Products* **24,** 47–53 (June 15, 1981).

"Microprocessor Applications Reference Book," Zilog, Inc., 1315 Dell Ave., Campbell, CA 95008, July 1981.

"Memory-Management Chip Masters Large Data Bases," D. L. Collins and C. M. Collins, *Electronic Design* **29,** 115–121 (August 20, 1981).

"SDLC Interface Mates M6800 Peripheral to 8086," S. Yakobovitch, *Computer Design* **20,** 169–180 (September 1981).

"Silicon Operating System Standardizes Software," C. McMinn, R. Markowitz, J. Wharton, and W. Grundmann, *Electronics* **54,** 135–139 (September 8, 1981).

"16-Bit-μP Benchmarks—An Update with Explanations," W. Patstone, *EDN* **26,** 169–203 (September 16, 1981).

"Microprocessor Data Manual," D. Bursky, *Electronic Design* **29,** 77–175 (November 26, 1981).

1982

"CMOS—LSI Microprocessors—Memories—Peripherals," RCA Solid State, 1998 Springdale Rd., Cherry Hill, NJ 07066, 1982.

"COPS Microcontrollers Databook," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1982.

"Peripherals Technical Overview," Zilog, Inc., 1315 Dell Ave., Campbell, CA 95008, 1982.

"Texas Instruments Semiconductor Products Master Selection Guide," *Manual No. SCG682*, Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, 1982.

"8086/8088 16-Bit Microprocessor Primer," C. L. Morgan and M. Waite, *BYTE*/McGraw-Hill, 70 Main St., Peterborough, NH 03458, 1982.

"As μP/μC Chips Mature, Support Chips Proliferate," R. H. Cushman, *EDN* **27,** 155–202 (January 6, 1982).

"Introduction to the iAPX 286," *Order No. 210308-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, February 1982.

"16-Bit-μP Peripheral ICs Provide Datacomm Support," D. R. Snyder, *EDN* **27,** 181–190 (February 17, 1982).

"Memory Protection Moves onto 16-Bit Microprocessor Chip," P. Heller, R. Childs, and J. Slager, *Electronics* **55,** 133–137 (February 24, 1982).

"Microprocessor Peripherals UPI® User's Manual," *Order No. 210317-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, April 1982.

"Software in Silicon: More Than a Program in ROM," A. Verhalen, *Electronic Design* **30,** SS33–SS36 (May 13, 1982).

"16-Bit μP Crams Peripheral Support on Chip," J. Klovstad, G. M. Gatlin, and T. Zingale, *Electronic Design* **30,** 191–196 (June 10, 1982).

"Microprocessors" (special issue on microprocessor systems development), *IBM Journal of Research and Development* **26** (July 1982).

"Technology Profile," D. Bursky, *Electronic Design* **30,** 83–94 (October 14, 1982).

"Integrating Memory Management into the CPU," G. Alexy, B. Childs, and J. Crawford, *Electronic Products* **25,** 55–62 (October 25, 1982).

"Multi-User Systems from Advanced Processor Chips," R. M. Schell, *Computer Design* **21,** 149–158 (November 1982).

"Multiprocessing Improves Robotic Accuracy and Control," P. Gupta, *Computer Design* **21,** 169–176 (November 1982).

"Fundamentals of Microprocessors," E. Lee, *Telephone Engineer & Management* magazine (November 1, 1982).

1983

"A Programmer's View of the Intel 432 System," E. I. Organick, McGraw-Hill Book Co., Inc., New York, 1983.

"Bipolar Microprocessor Logic and Interface Data Book," Advanced Micro Devices, Inc., 901 Thompson Pl., P.O. Box 453, Sunnyvale, CA 94086, 1983.

"iAPX 286 Hardware Reference Manual," *Order No. 210760-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"iAPX 286 Operating Systems Writer's Guide," *Order No. 121960-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"iAPX 286 Programmer's Reference Manual," *Order No. 210498-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"iAPX 86/88, 186/188 User's Manual," *Order No. 210911-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"Introduction to the 80186 Microprocessor," K. Shoemaker, *Order No. 210973-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"Microprocessor and Peripheral Handbook," *Order No. 210844-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1983.

"MOS Microprocessors and Peripherals Data Book," Advanced Micro Devices, Inc., 901 Thompson Pl., P.O. Box 453, Sunnyvale, CA 94086, 1983.

"MOS Microprocessor Data Manual 1983," Signetics Corporation, 811 E. Arques Ave., P.O. Box 409, Sunnyvale, CA 94086, 1983.

"NCR/32-000-32-Bit Microprogrammable Microprocessor," NCR Microelectronics Division, Colorado Springs, CO, 1983.

"NS16000 Databook," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1983.

"TMS32010 User's Guide—16/32-Bit Digital Signal Processor," Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, 1983.

"An Architectural Comparison of 32-Bit Microprocessors," A. Gupta and H. D. Toong, *IEEE MICRO*, pp. 9–50 (February 1983).

"Mostek 1983 Computer Products Data Book," Mostek Corporation, 1215 W. Crosby Rd., Carrollton, TX 75006, March 1983.

"Designing with Microprocessors: The Modular Approach," M. Biewer, *Telephone Engineer & Management* **87,** 67–69 (March 1, 1983).

"Computerizing the Car," W. Brown, *SKY* magazine, pp. 24–33 (October 1983).

"Microprocessors: Speed Up, Price Down, and CMOS Everywhere," S. Bassett, *Computer Design* **22,** 177–187 (October 1983).

"Advanced Features Squeeze onto Processor Chip," J. Slager, *Computer Design* **22,** 189–193 (October 1983).

"The M68000 Educational Computer Board," R. W. Floyd, *BYTE* **8,** 324–336 (October 1983).

"Examine Architectures When Evaluating μPs," K. Christian, *EDN* **28,** 193–202 (October 13, 1983).

"EDN's Tenth Annual μP/μC Chip Directory," R. H. Cushman, *EDN* **28,** 111–256 (November 10, 1983).

"Educational Board Computer Teaches Engineers the Micro-Processor Game," M. Gallagher, *Electronic Products* **26,** 115–121 (November 17, 1983).

"Bipolar Arithmetic Chip Speeds 68000's Math Throughput," V. J. Coli, C. Hastings, S. Rajpal, and R. W. Blasco, *EDN* **28,** 179–193 (November 24, 1983).

1984

"Distributed Control Modules Databook," *Manual No. 230973-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1984.

"Guide to Using the Distributed Control Modules," *Manual No. 146312-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1984.

"How to Use Surface Mount Technology," J. Mullen, *Manual No. SSYZ001*, Texas Instruments Inc., P.O. Box 225012, MS-54, Dallas, TX 75265, 1984.

"MC68020 32-Bit Microprocessor User's Manual," *Motorola Manual No. MC68020UM (ADI)*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

"Microcontroller Handbook," *Order No. 210918-002*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1984.

"Microsystem Components Handbook" (two-volume set), *Order No. 230843-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1984.

"Series 32000 Instruction Set Reference Manual," *Publication No. 420010099-001B*, National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, 1984.

"Software Handbook," *Order No. 230786-001*, Intel Corporation, 3065 Bowers Ave., Santa Clara, CA 95051, 1984.

"8088 Family on the STD Bus," *System Designer's Guide, Manual No. Ziatechniques 5*, Ziatech Corporation, 3433 Roberto Ct., San Luis Obispo, CA 93401 (undated).

"Introduction to the NS16000 Architecture," National Semiconductor Corporation, 2900 Semiconductor Dr., Santa Clara, CA 95051, February 1984.

"STD Bus Technical Manual and Product Catalog," Pro-Log Corporation, 2411 Garden Rd., Monterey, CA 93940, February 1984.

"8X300 Family Capability Manual," Signetics Corporation, 811 E. Arques Ave., P.O. Box 409, Sunnyvale, CA 94086, March 1984.

"Abundant Board-Level μCs Pack Increased Processing Power," D. Powers, *EDN* **29,** 213–220 (April 19, 1984).

"Versatile Serial Protocol for a Microcomputer-Peripheral Interface," D. C. Stanley (presented at Mini/Micro Northeast, May 1984), RCA Solid State, Rte. 202, Somerville, NJ 08876.

"VLSI-Based LAN-Controller Chip Eases μP-to-Network Interface," R. H. Cushman, *EDN* **29,** 207–220 (May 3, 1984).

"Synchronous 32-Bit Backplane Buses Open Up Distributed-System Design," B. Nicholson, *EDN* **29,** 75–86 (June 14, 1984).

"Thirty-Two Bit Micros Power Workstations," N. Mokhoff, *Computer Design* **23,** 97–112 (June 15, 1984).

"Serial Backplane Suits Multiprocessor Architectures," M. Webb, *Computer Design* **23,** 85–96 (July 1984).

"32-Bit Extension to the 68000 Family Addresses 4G Bytes, Runs at 3 MIPS," W. Twaddell, *EDN* **29,** 75–77 (July 12, 1984).

"32-Bit Processors Pack Mainframe Muscle," J. Javetski, *Electronic Products* **27,** 49–55 (July 16, 1984).

"Semiconductor Memories: Density and Diversity," T. Williams, *Computer Design* **23,** 105–116 (August 1984).

"Circuit Density and Speed Boost Tomorrow's Hardware," J. Bond, *Computer Design* **23,** 210–225 (September 1984).

"Hardware/Software Problems Yield to Today's Logic Analyzers," K. Lowe and M. Van Hook, *EDN* **29,** 203–208 (September 6, 1984).

"Memory-Management Varieties Suit Different Application Areas," D. Phillips, *EDN* **29,** 135–143 (September 6, 1984).

Received June 18, 1984; revised October 10, 1984

Robert C. Stanley *IBM System Products Division, P.O. Box 1328, Boca Raton, Florida 33432.* Mr. Stanley is an advisory engineer in the process automation group at Boca Raton. He joined IBM in 1959 at Poughkeepsie, New York, where he held various assignments in test equipment and numerical control. In 1965, he was transferred to the East Fishkill, New York, plant, where he designed controls for automated assembly machines, process tools, and test equipment. He was the lead engineer for the controls on a computer-controlled sensor-based process line and the systems engineer for a fully automatic computer-controlled alignment system. In 1977 he joined corporate components procurement engineering in Poughkeepsie and was active in the corporate qualification of the IBM 6800 and IBM 8086 families of microprocessors and peripherals. In 1978, Mr. Stanley became a member of the IBM Mid-Hudson Education Microprocessor Education Advisory Committee and a member of the Electrical Technology Advisory Committee for Dutchess Community College. Since moving to Boca Raton in 1980, he has been active in designing microprocessor-based control systems for robotics and machine control. Mr. Stanley has been very active in engineering education and has given over 100 seminars on microprocessors and taught over 80 logic design and microprocessor system design courses ranging from digital logic, numerical control, and basic microprocessors to detailed design engineering on seven different microprocessors. He has been a guest instructor and lecturer at many IBM plants and laboratories as well as at high schools and colleges. He has received a number of awards for his work in machine control design and teaching and in 1980 was honored by the IEEE for outstanding educational services and superior technical contributions in the field of microprocessor engineering education. In 1984, Mr. Stanley received an IBM Outstanding Innovation Award for the system architecture of an electric drive robot.