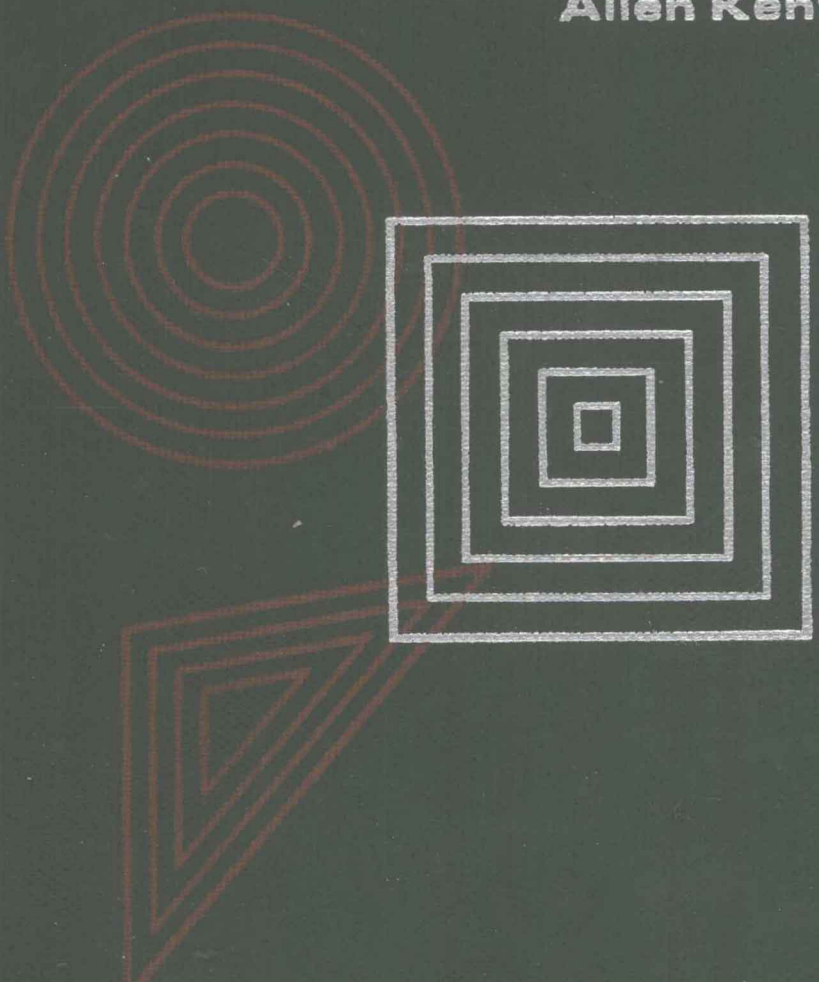
ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

11

EXECUTIVE EDITORS
Jack Belzer
Albert G. Holzman
Allen Kent



ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Jack Belzer *Albert G. Holzman* *Allen Kent*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

VOLUME 11

Minicomputers to Pascal

MARCEL DEKKER, INC. • NEW YORK and BASEL

MARCEL DEKKER, INC.
270 Madison Avenue, New York, New York 10016

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 74-29436
ISBN: 0-8247-2261-2

Current printing (last digit):
10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

VOLUME 11

INTERNATIONAL EDITORIAL ADVISORY BOARD

CONTRIBUTORS TO VOLUME 11

MORDECAI AVRIEL, Faculty of Industrial and Management Engineering, Technion University, Haifa, Israel: *Nonlinear Programming*

BRUCE G. BUCHANAN, Department of Computer Science, Stanford University, Stanford, California: *Models of Learning Systems*

J. CHRISTOPHER BURNS, Arthur D. Little, Inc., Cambridge, Massachusetts: *Newspaper Automation*

SAMUEL D. CONTE, Department of Computer Science, Purdue University, West Lafayette, Indiana: *Operating Budgets for Computer Centers*

JAMES W. COOPER, Department of Chemistry, Tufts University, Medford, Massachusetts: *Minicomputers*

BENJAMIN S. DURAN, Department of Mathematics, Texas Tech University, Lubbock, Texas: *Normal Distribution*

JAMES S. DYER, University of California, Los Angeles, California: *Multicriteria Decision Making*

JAMES W. FLEMING, Materials Research Laboratory, Bell Telephone Labs, Murray Hill, New Jersey: *Optics, Fiber*

C. RICHARD JOHNSON, Jr., Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia: *Models of Learning Systems*

JACK P. C. KLEIJNEN, School of Economics, Social Sciences and Law, Tilburg University, Tilburg, The Netherlands: *Operations Research*

ERWIN KREYSZIG, Department of Mathematics, University of Windsor, Windsor, Ontario, Canada: *Nonlinear Vibrations*

G. JACK LIPOVSKI, Department of Electrical Engineering, University of Texas, Austin, Texas: *Nonnumeric Architecture*

HAROLD LORIN, IBM Systems Research Institute, New York, New York; Hofstra University, Hempstead, New York: *Operating Systems*

DAVIS B. McCARN, National Library of Medicine, Bethesda, Maryland: *National Library of Medicine–MEDLARS and MEDLINE*

EDWARD MINIEKA, Department of Quantitative Methods, University of Illinois, Chicago, Illinois: *Network and Graph Problems*

TOM M. MITCHELL, Department of Computer Science, Rutgers State University, New Brunswick, New Jersey: *Models of Learning Systems*

BRUCE RHOADS, Purdue University, West Lafayette, Indiana: *Operating Budgets for Computer Centers*

NAOMI SAGER, Linguistic String Project, New York University, New York, New York: *Natural Language Analysis and Processing*

RAKESH SARIN, Department of Management, Purdue University, West Lafayette, Indiana: *Multicriteria Decision Making*

G. MICHAEL SCHNEIDER, Department of Computer Science, University of Minnesota, Minneapolis, Minnesota: *Pascal*

REID G. SMITH, Defense Research Establishment Atlantic, Dartmouth, Nova Scotia, Canada: *Models of Learning Systems*

JACK A. TAYLOR, School of Music, Florida State University, Tallahassee, Florida: *Music*

B. L. TRIPPETT, NCR Corporation, Dayton, Ohio: *NCR Corporation*

CONTENTS OF VOLUME 11

MINICOMPUTERS

INTRODUCTION

In recent years the definition of a minicomputer has become somewhat difficult since the advent of powerful, expensive, shorter word-length machines and the introduction of low-cost microprocessors. It is therefore convenient to define the minicomputer by price range rather than by word length or capabilities since these latter two qualities vary so widely with each product. The typical microprocessor costs only a few tens to a few hundreds of dollars and may have a word length of 8 to 16 bits and a relatively complete instruction set. It usually does not include very much in the way of input or output devices, and may not even include a power supply. On the other hand, one can purchase at not inconsiderable cost a number of peripherals for the microprocessor including a hard or soft copy terminal, a low-cost storage device such as floppy disk, and even a floating point hardware processor. These peripheral devices can run the cost of what we originally called a microprocessor well up into the minicomputer range.

At the other extreme, the minicomputer of moderate capabilities can be embellished with extensive memory and memory management hardware, multiple terminals, large amounts of disk storage, and hardware floating point intructions, as well as expensive high speed or cache memory and special purpose peripheral controllers and business instruction sets. In this case the minicomputer has jumped to a new class of computer sometimes called a medium-scale or "midi"-computer and may cost well over $50,000.

For the purposes of this article then, we will define the limited microprocessor scale as having a price range of $5 to $3000 and the minicomputer as costing from $3000 to $50,000. We might define the cost range of midicomputers as $50,000 to $200,000 and class most systems costing over $200,000 as large-scale computers.

HISTORICAL BACKGROUND

It is only necessary to look back in the literature of 25 years ago or so to realize the enormous power of today's computers both in size, power requirements, and cost compared to the original designs. In 1952 Everett [1] reported the design and capabilities of the Whirlwind I computer. This computer and a number of its later descendants developed at M.I.T.'s Lincoln Labs served as the predecessors of the modern minicomputer. The Whirlwind I was a 16-bit word computer having a 5-bit instruction field and an 11-bit address field. It was capable of performing 20,000 operations/sec. Because of the high cost of memory in those days, it had originally only 256 words of storage but was later expanded to 1024 words and was one of the first computers to utilize magnetic core memory. Additional storage was gained by use of a high-speed magnetic drum. The 11-bit address field led to a possibility of direct addressing of 2048 words, which was the eventual goal for this machine.

The Whirlwind I had over 5,000 tubes in it, mostly pentodes, as well as 11,000 crystals. Of the 32 possible instructions, 27 were assigned operations. The architecture of the computer was basically that of a single accumulator machine with

additional registers for multiplication and division. Arithmetic operations were carried out in 9's complement arithmetic. The size of this early machine was staggering: a single flip-flop required a volume of about 8 ft^3 and the power requirements were over 150,000 W.

At this period in computer development, all programs were written in machine language, and because of the high demand for computer time, no on-line debugging time was ever allowed. Programmers were expected to interpret dumps of relevant memory segments to discover their errors. The computer's uptime during a $2\frac{1}{2}$ month period was found to be 85%, a reasonable record even compared to some of today's high technology computer systems.

Perhaps the most important statement in this article, though, was a philosophical one regarding the machine's design with a limited word length and addressing capability:

> It is much better to waste computing time occasionally . . . than to waste continuously a large part of the storage and computing equipment of the machine by providing an unnecessarily large [word length].

It is really this philosophy that led to development of other machines of lower cost and limited capability at M.I.T. such as the TX0 and the LINC computer. Basically, if the computer could be built cheaply enough that it could be dedicated to fewer tasks, then the limited computing capacity could be circumvented when necessary by additional somewhat slower programming techniques. In today's world of dedicated computer systems and microprocessors in many appliances and vehicles, this philosophy has been carried to its logical conclusion.

DEVELOPMENT OF THE MODERN MINICOMPUTER

One of the earliest successful minicomputers was the PDP-5 and its closely following relative, the PDP-8. These machines represented the second generation of computers in price, compactness, and design innovation. The PDP-8 and at least two other competitors featured a 12-bit word and a 3-bit instruction field, leading to eight principal instruction classes. Two of these classes were further divided, so that the I/O instruction could produce three different pulses to 64 different possible devices. The other subdivided class contained the Operate instructions, which had no address field but decoded the remaining 9 bits into various shift and test instructions.

The instruction set of the PDP-8, then, featured six memory reference instructions:

AND - logical AND between a memory location and the AC
TAD - two's complement addition of a memory location to the AC
ISZ - increment a memory location and skip if the result is zero
DCA - deposit the AC in a memory location and zero the AC
JMS - Jump to a subroutine, leaving the PC in the first location
JMP - jump to a specified memory location

Several things should be noted about the PDP-8 instruction set. For instance: there was no type of OR instruction, either inclusive or exclusive, and the operate

instructions provided only single and double shifts. The PDP-8 was a single accumulator machine with all arithmetic operations occurring in that register as far as the programmer was concerned. The 12-bit accumulator (AC) and a 1-bit carry-out register could not simply be loaded using this instruction set. Instead, numbers were always added to the AC, and when the AC was stored in some memory location, it was always zeroed. Thus the AC had to be restored after every deposit if further operations were to be done on that number and cleared before every load if addition was not desired.

In the earliest PDP-8's multiplication was accomplished only by software, although hardware multiply-divide instructions were added in later models. Even in these, however, multiplication was unsigned and sign-keeping was the responsibility of the user. Since the PDP-8 was a single accumulator machine, the facility for handling double precision numbers was extremely limited. Only one of the two numbers could be in the accumulator at a time so that the addition could consist of quite a few steps. This, however, was in line with the general philosophy expressed above: if something is not done often it is better to do it more slowly by software than to purchase hardware that is seldom used.

The PDP-8 also provided a single level interrupt facility which could be either on or off. Later PDP-8's allowed the user to select which devices were capable of causing an interrupt at any given time, but the earlier ones required that the programmer poll all devices to find out which one caused the interrupt. This usually was not as onerous as it seemed since, in a dedicated computer environment, the number of possible devices was quite limited.

ADDRESSING MODES OF THE PDP-8

Since the PDP-8's 12-bit word severely limited the number of bits available for addressing, one might believe that it could only address a few memory locations much as the Whirlwind I could. However, the PDP-8 allocated 8 bits to the address field, seven for a page-relative address and one for the selection of page-zero or current page. This addressing scheme allowed the computer to address directly 128 words on the current 128-word block (or page) and 128 additional words on page zero, or the first 128 words of memory. While this still allowed addressing of only 256 words directly, the computer could also use the contents of any word as a pointer address to some other location on any page. Thus the resourceful programmer put program code on all pages but page zero and put a list of pointer addresses on page zero which allowed him to address constants and routines on any page by indirect addressing.

The PDP-8 also allowed indirect addressing of the sort useful in processing lists: it provided eight locations on page zero which when addressed indirectly, that is, used as pointer addresses, automatically incremented after use. Thus one could easily step through a table of addresses without having to specifically increment the pointer each time.

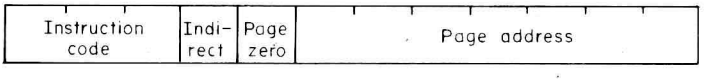


FIG. 1. Memory reference instruction decoding for the PDP-8 minicomputer.

The above description deals with three common addressing modes: direct, indirect, and indirect with autoincrement of the pointer. The direct addressing mode utilized 7 bits of the instruction as an address to which the address of the start of the page was added if the current page bit is set and to which no address was added if the page zero condition is set: the current page bit off. This address was addressed directly if the indirect bit is zero, and is used as a pointer address to the actual data address if the indirect bit is set. This is illustrated in Fig. 1.

ADDITIONAL ADDRESSING MODES

While the above three modes, direct, indirect, and indirect autoincrement, are the only addressing modes in the PDP-8, many other minicomputers use a number of other addressing modes which we will summarize here. Since, in all cases, the minicomputer's word length is rather short, say 12 to 24 bits, the number of bits allocated to addressing is rather small and various tricks must be used to allow the computer to address any substantial number of memory words.

Two-Word Addressing

The simplest possible method of extending the addressing capabilities of a computer is to always use two words: one for the instruction and one for the address. In a 16-bit word computer, this can amount to a total addressing capability of 65,536 elements, either words or bytes as the designer prefers. On the other hand, such capabilities, which have been included in a number of minicomputers, require twice as much memory for programming, since every instruction requires two words. The obvious alternative is to use a second word for addressing only when more distant addresses are required and use a few bits of the single word instruction for accessing nearby addresses. This method is utilized in a number of minicomputers such as the PDP-8, where indirect addressing always uses a second word containing the address but direct addressing uses only one word. It should be noted that in very short word-length computers such as the PDP-8, even a full word address can define only 4096 different words. Consequently, larger PDP-8's required that additional address bits be set by separate special instructions to change the instruction field and the data field.

Optional 2-word addressing is used in a number of minicomputers such as the Nova and the Varian 620 and V74 series, and the Nicolet 1080, 1180, and Bruker Aspect series, where a single instruction word can address only some memory locations, but when the indirect bit of the instruction is set, the second word is addressed as a pointer location for indirect addressing.

RELATIVE ADDRESSING

The bits of the instruction which specify the word addressed can contain the address of the location relative to the start of some arbitrary page of, say 128 words, so that addressing location 6 from an instruction in decimal address 150 would actually refer to address 128+6 or address 134. This is called page-relative addressing. Alternatively, the address can be relative to some other constant such

as the address of the current instruction. This is called program counter (PC) relative addressing, since the program counter always contains the address of the next instruction. In this latter case, addressing location 6 from decimal location 150 would actually address location 156. In addition, it could be relative to some constant address stored in an index register.

The PC-relative addressing mode is not limited to single word addressing, but can also be utilized where the second word is always to be added to the PC to determine the actual address. Using signed arithmetic, the address word can then lie either before or after the location of the instruction. This addressing mode has the advantage along with the straight indirect addressing mode that any word in the computer's memory whose address can be represented in one full word can be addressed. However, in minicomputers such as the PDP-11, where the relative address is always stored in the second word of the instruction, there is the definite disadvantage that two words must be used whenever this same distant address is accessed, while in the indirect method the same nearby pointer can be utilized repeatedly.

MULTIACCUMULATOR MACHINES

With the development of longer word minicomputers and more sophisticated instruction sets, the single accumulator architecture of the PDP-8 type machine was expanded to several accumulators so that several calculations could be done at one time. In the Data General Nova series, for example, there are four accumulators which can be added to each other, used for double precision work and as multiply-divide registers. On the other hand, since it takes 2 bits to specify which accumulator is being used, the number of instructions is somewhat reduced so that additions and similar instructions can take place only between accumulators rather than between memory and accumulators.

In other machines, such as the Nicolet 1180 series, the power of the single accumulator has been enhanced by a more active multiplier-quotient register and accumulator extension which can be utilized directly within instructions for data storage, multiple precision shifts, and additions. Thus, while the accumulators are not separate, the overall power is increased by the ability to keep several numbers in active registers at once.

MULTIPLE REGISTER MACHINES

Later minicomputers have taken the register approach to the instruction set, where all of the memory reference instructions refer only to a set of special registers. These registers can be used as accumulators, but they also can be used as address pointers which can be incremented or decremented automatically, or, in conjunction with a memory location, as PC-relative address pointers. This approach gives a great deal of versatility to the handling of tables of data, since several registers can be used as pointers to the tables, but when the registers are used in conjunction with a memory location to generate PC-relative addressing, the programs can use larger amounts of core for even very simple operations. Further, when data must be in the registers or referred to by the registers, the operation of getting it there may require several two-word instructions, where the second word defines the offset to be added to the PC to retrieve the constant needed.

LARGER MINICOMPUTERS

As the costs of core memory and solid-state memory continue to decrease, the number of minicomputers having longer word lengths has increased. The pioneering 20-bit Nicolet 1080 series featured a single accumulator architecture which became extremely powerful because of the 10-bit address fields and the number of powerful memory reference instructions that could be generated in 20-bits. The later 1180 computer system carried this power to an even greater extreme, featuring a full complement of interrupts, more addressing modes and page-0/current page addressing, as well as more memory reference instructions. The 24-bit Bruker Aspect computer continued this architecture to very long word lengths and implemented 13 different addressing modes, utilizing four index registers, page-0/current page addressing, and PC-relative addressing as well, making it one of the most powerful minicomputers available. The 24-bit Datacraft computer is another extremely powerful machine, but its sophistication of hardware and peripherals takes it out of the minicomputer price class. In addition there are some 32-bit computers whose instruction sets grew out of lower-priced 16-bit cousins which allow full 32-bit instructions and addressing which can no longer be considered minicomputers in either price or capabilities.

SUBROUTINES

There are two basic methods for allowing computer programs to call subroutines from several points in the program. One method allocates a location at the beginning of the subroutine to storage of the program counter so that the subroutine can return to the point from which it was called. The other places the program counter in a special area of memory reserved for a table of such return addresses called a stack. There are advantages to both approaches and both types of machines are quite popular with their users.

The memory-pointer method always allows the programmer to know easily where the routine was last called from during debugging and most easily allows abnormal exit from one subroutine to some other location. On the other hand, it is not possible to write subroutines that call themselves or call each other as easily as when the stack method is used. The stack method is most useful in writing subroutines that in evaluating some expression must store intermediate results and call themselves over again, since after each call a new entry will be made on the stack and when exit from the various calls begins it will eventually end up at the location following the original call, no matter how many times the routine has been recalled since. The principal disadvantage to this stack method is that abnormal exit from one subroutine through another is much more complex. Further, for such recursive subroutine calling a fairly sizeable memory area may be needed for all the intermediate stored PC's.

INPUT AND OUTPUT DEVICES FOR THE MINICOMPUTER

The usual communication medium for use with the minicomputer is some sort of keyboard terminal, sometimes with the ability to read paper tape at low speeds. These terminals, exemplified by the low-cost Teletype, are in wide use, but are

limited to slow printing and tape reading speeds of, say, 10 characters per second. The binary tape format used by the tape reading programs is such that two or three 8-bit lines of punched paper tape are read for each computer word to be stored. Paper tape reading is not an error-free process, however, since there can be tape-reading errors as the tapes get older and the tapes are therefore punched with checksums at regular intervals which the tape reading program can compare with the sum of the data read in to that point. There is no easy recovery from this type of error, however, and the entire tape must often be read again to rectify the error.

Most advanced printing terminals have 30 character per second printing speeds and no tape-reading facilities at all. In these systems programs are read in by a high-speed tape reader (of 50 to 300 characters/sec) or from a cassette of audio-grade tape. More advanced systems feature floppy disks or "diskettes" having storage capacities of about 100,000 words for storage of programs, or in the better systems cartridge disks having million word capacities or more.

FLOATING POINT ARITHMETIC

While the predominant trend in minicomputer calculations is to do most of them as single-precision signed integers, there are a number of cases in which it is desirable to manipulate numbers outside the range possible in a single word. For example, the 16-bit word can only represent numbers from 0 to 65535 and if signed numbers are used from -32768 to +32767. This range is so small that it cannot be used in most scientific calculations. While numbers up to 2^{31} or so could be represented in two words, this would leave no room for the representation of numbers less than 1. Therefore a system of representation much like that used in scientific notation is used in the computer.

In scientific notation, we simply arrange all numbers to lie between 1.00 and 9.999 . . . and adjust the power of 10 accordingly, so that 567,234 becomes 5.67234×10^5. In floating point notation we likewise adjust all numbers to lie in a specified range, say between 0.5 and 1.0, and adjust an exponent of two accordingly. Thus a number like 5 becomes in binary:

101 or
10.1×2^1 or
1.01×2^2 or
$.101 \times 2^3$

In the last line above, we have shifted the bits in the number to lie between .5 and 1 since the place to the right of the binary point represents $2^{-1} = 0.5$. The exponent portion of the number is then 3, meaning that the fraction must be multiplied by 2^3 to be restored to its true value. The bits representing the exponent are stored in one part of the floating point number, say in 8 to 10 bits, and the remainder become the fractional part, sometimes called the mantissa, and are stored in the remaining bits of the 2-word number. These numbers are most commonly manipulated by software packages provided by the manufacturers, but on occasion may be produced by optional hardware floating instructions in the higher level minicomputers.

INTERRUPTS

When a minicomputer is used in a dedicated environment, it may be extremely busy with some computational task much of the time, but on occasion may have to process some data caused by some external event. This is most commonly a key being stuck on a keyboard of a terminal or some peripheral device such as a disk or analog-to-digital converter requiring attention. When these latter two devices require attention it must be granted immediately or the data may be lost. This can also be true with some teleprinter keyboards. It is not convenient, however, nor practical to write programs which, in the midst of many calculations, continually examine the state of various peripheral devices. This function is most efficiently accomplished by allowing the peripheral devices to interrupt whatever program is executing when, and only when, they need a response. The interrupt system usually calls for the programmer to set some register or memory location with an address where code for servicing that device is located and set some status bit associated with the device which allows it to cause interrupts. Then, when the device needs service, the interrupt hardware takes over, saving some or all of the values associated with the processor's current activity and jumping to the specified location where a program routine exists for sending or receiving information for that device.

Interrupts in minicomputers have ranged in complexity from none at all to multipriority vectored schemes. The early PDP-8 had simple interrupt capabilities: a single level of interrupt, such that all enabled devices caused a jump to the same place, where the current PC was saved and the user had to write fairly complex code to decide which device has caused the interrupt. In this simple interrupt structure there was no way of differentiating highly time-critical devices such as clocks and high-speed analog-to-digital converters from lower speed devices such as teleprinters. More complex interrupt structures in later computers allocated different classes of devices different priorities, so that those with the most time critical functions could not only interrupt whatever main program was being processed, but could also interrupt servicing of lower speed devices. Some of the most advanced interrupt capabilities come in computers such as the PDP-11 where not only are several priority levels provided, but each device has its own address in memory where a pointer for its specific service routine can be placed, thus completely eliminating any need to poll the various devices on that level to find out which one caused the interrupt.

In all interrupt systems, the program counter value is saved when the interrupt occurs so that the service routine can easily return to the place in the main program where the interrupt occurred. However, almost none of the minicomputers currently in production automatically save any of the other important active registers, such as accumulators or multiplier-quotient registers. One exception is the Bruker Aspect computer system which has a special stack on which the three accumulators are automatically saved as each priority interrupt occurs.

INTERRUPT LATENCY

All interrupts require some time to occur in minicomputers. At any given instant the computer can be anywhere in the processing of an instruction and the computer can clearly be interrupted only at certain times, usually at the end of the processing of an instruction. Since some of the more complex instructions in modern

minicomputers can take quite some time to execute, in particular multiply, divide, and floating point operations, some minicomputers have tried to allow more interrupt flexibility by allowing interrupts to occur during these longer instructions, with subsequent resetting of the PC and reexecuting of the instruction when the interrupt servicing is over. This method, of course, adds to the cost and complexity of the design but may be required for certain time-critical applications.

OTHER DATA CHANNELS

It is clear that in any computer there must be a direct and rapid pathway between the central processor and the computer's memory for the rapid processing of instructions and accessing of data. However, most minicomputers also have at least one other pathway for access to memory. High-speed devices such as disk drives and some magnetic tape units generate data quite rapidly and can require the constant attention of the processor to load or store data in memory from the disk drive. However, this transferring of data to or from this disk is such a simple, linear task that it needn't require the attention of a complex central processor. Instead, a second data path is provided to memory and a simple peripheral processor is built as part of the disk drive controller which simply moves data to or from disk and memory as directed. The central processor meanwhile need only be instructed to leave that area of memory alone until the transfer is complete. This process requires an additional data path sometimes called direct memory access or a high-speed data channel.

While most minicomputers have had the capability to transfer data using these data channels for some time, there is currently a trend away from use of these capabilities in some of the newer lower priced models and in the microprocessors, since, following the philosophy of the Whirlwind computer, a little extra elapsed time is less important than the excess unnecessary capability.

SPECIAL HARDWARE FOR DATA ACQUISITION

The minicomputer used in a dedicated system very often is concerned with data logging or acquisition of data from scientific or industrial instruments. These data may or may not be in digital form as they leave the instrument in question, but, if as is usual in older instruments, they are an analog voltage, they must be converted to a digital number by an analog-to-digital converter (ADC). ADC's convert voltages in a specified range to digital numbers in a specified range. For example, a 6-bit unipolar ADC might convert voltages from 0 to 1 to be represented as binary numbers from 000000_2 to 111111_2.

ADC's today are available in ranges from 6 to 16 bits with input voltages that can be adjusted to any convenient range with an input attenuator or preamplifier. In addition, the computer must be told how often to acquire a data point from the instrument or device. This timing can be accomplished by counting memory cycles in a wait loop or by the use of a programmable real-time clock. These clocks can usually be set using a countdown buffer and a frequency selector so that a fairly wide range of clock output frequencies is possible. In acquiring data from the ADC, most sophisticated data-gathering systems allow the clock to be set to virtually any desired counting period and allow the clock to start the ADC directly. The only