The minicomputer, a programming challenge

by ROBERT L. HOOPER

Compata, Incorporated
Tarzana, California

INTRODUCTION

Public attention, as well as the attention of groups such as this, has been focused on the design and application of large, expensive "super computers." Our national preoccupation with size and power makes this fact understandable. However, the minicomputer, which I define as a stored program computer selling for under twenty-five thousand dollars, is deserving of much more serious attention than it has heretofore been given.

While a 30% increase in units sold each year is possible over the next few years merely from increased use in the industrial areas, a much greater growth rate could come about if the minicomputer ever became an entry in the consumer goods market. At prices of two to four thousand dollars per main frame, a minicomputer with a dial-up service to specialized data bases could become attractive to the growing percentage of the population in the upper income bracket.

But such speculation is not the purpose of this paper. Rather, it is to examine the current status of programming support for the minicomputer and to suggest both software enhancements and hardware features to support them.

Ten years ago, an "inexpensive" computer typically sold for forty to fifty thousand dollars, was based on drum memory and discrete component technology, and could execute from a few hundred to a few thousand instructions per second. The numbers of all such computers delivered in any one year did not exceed a few hundred. Examples of such computers would include the Control Data LGP–30 (Librascope) and G–15 (Bendix).

Today, in 1968, the minicomputer is a proliferating breed, selling for anywhere from five to twenty-five thousand dollars, utilizing core memory and integrated circuits, and executing up to a million instructions per second. Over three thousand such systems will be delivered next year. It is my belief that during the next few years the percentage growth of this segment of the total computer market will increase three to four times faster than that of the overall computer market, which seems to be currently at 12% per year.

Speculations about the future in areas of technology usually suffer from extreme under- or overshoot. However, a continued sharp decrease in price (due to both manufacturing technology and to competition) and a modest increase in performance and reliability does not strain one's credulity. Changes in the software area are more uncertain. Price will no doubt continue to be the major determining factor in original equipment manufacturer orders, which are bread and butter in this price range.

Current minicomputer

Current minicomputers include the Digital Equipment Corp. PDP–8 series, the Hewlett-Packard HP2115A and 2114A, the Honeywell/Computer Control Division DDP–416, the Interdata models 2, 3 and 4, the Raytheon 703, the Scientific Control 650 and 655, the Systems Engineering Laboratories 810A and the Varian Data 620i and 520i. In addition, there are many other companies manufacturing or contemplating manufacture of minicomputers with a delivered price under $15,000. There has been a recent "outburst" of offerings from lesser known companies offering computers in the $5,000–$10,000 range: SPC 12, DT 1600, Elbit 100 and PDC 808, to name only a few. Table I lists some current minicomputers available for under $13,000. The more successful models have been delivered in quantities ranging

Manufacturer	Model	Price	Memory (bits)	Memory Cycle (usecs)
Digital Equipment Corp.	PDP–8/I	$12,800*	4096×12	1.5
Digital Equipment Corp.	PDP–8/L	8,500*	4096×12	1.6
Elbit Computers	100	4,900	1024×12	2.0
General Automation	SPC–12	6,400	4096×8	2.2
Hewlett-Packard	2114A	9,950	4096×16	2.0
Hewlett-Packard	2115A	14,500	4096×16	2.0
Interdata	2	4,700	1024×8	2.0
Interdata	3	7,000	4096×8	2.0
Interdata	4	10,000	4096×8	2.0
Varian	DATA/520i	7,500	4096×8	1.5
Varian	DATA/620i	12,500	4096×16	1.8

*Price includes ASR–33

TABLE I—Current minicomputers (under $13,000)

from hundreds to over a thousand. In general, these computers have word lengths of twelve or sixteen bits, and standard memories of 4,096 words, expandable to 32,768 words as an option An ASR–33 teletype is the usual I/O device, but optional equipment includes high speed paper tape reader and punches, card readers, line printers, magnetic tapes and disks.

To obtain some historical perspective concerning performance and cost of small control computers over the past eight to ten years, let us remember that prices have decreased by a factor of four to eight, while memory speeds have decreased by about the same factor. A trend has also developed away from the extremes of twelve and twenty-four bits toward sixteen bits as a standard word length for this type of computer, although the PDP–8 series, at twelve bits, has been and still is, the most popular system.

It is interesting to note that in this same time period there has been little change in the concept of what manufacturer-supplied software should be like. But more of this later.

Applications characteristics

As the price per unit dropped and reliability and performance improved, a predictable result occurred. More and more applications became feasible for computer-based systems. The hard-wired "black box" was increasingly replaced by a stored program computer. Applications of minicomputers are frequently of a "real-time, on-line" nature and include closed loop control of processes, data acquisition and recording, analytical instrumentation, automatic test, and communication systems. Users of the minicomputers are unlikely to be professional programmers but rather to be scientists, engineers and technicians without extensive programming experience. For this reason, the minicomputer should be supported by software suitable both to the application and to the user. Unfortunately, most present day minicomputer software does not appear to meet this requirement.

Standard software

The typical software configuration for a minicomputer can be described as follows: FORTRAN is available for systems with 8K words of memory. A basic one-for-one assembler is available for 4K systems and a macro assembler is available for 8K systems. Loaders which link, desectorize and relocate are usual. A tape editor is quite common and every computer seems to have math packages of single precision and possibly double precision fixed point routines. Hardware fault detection and on-line debug routines are also standard. Basic I/O drivers are normally included, as are mathematical routines for elementary functions.

Some of the software problems are caused by the minimum hardware configurations sold:

- small memory (4,096 words)
- slow I/O (ASR 33)

- lack of mass storage
- limited ability to address memory within an instruction.

Other problems, previously mentioned, result from an attempt to make available the standard tools of the professional programmer to the novice or occasional user. This seems inappropriate.

An economic point is worthy of some discussion. The desire to keep the unit price low is certainly an overriding requirement. If ever there were a case for separate pricing of software, this would seem to be it. This would be especially appreciated by the quantity buyer. My suggestion would be that a minimum set of software (to be defined) be included with each computer sold at a small additional cost and that one copy of associated documentation be furnished. Additional documentation sets and/or additional programs would be available for added payment. The customer must realize that printing costs and tape reproduction and verification costs are not negligible. They might run as high as half of the amortized non-recurring development costs. For one manufacturerer, these costs ran from a few hundred to several hundred dollars for documentation production costs associated with the delivery of a single minicomputer.

A basic software set might be:

ASSEMBLER (BASIC)
LOADER (RELOCATING AND DESECTORIZING)
TAPE EDITOR
DEBUG PACKAGE (BASIC)
I/O CONTROL SYSTEM
HARDWARE FAULT DETECTION ROUTINES

This basic software set might be developed for under $50,000. Amortized over 250 computers, the non-recurring cost would add only $200 per system, with perhaps another $100 added for reproduction costs, resulting in a $300 additional charge for the customer who needs only this minimum software package.

An augumented set available at additional cost might be:

ASSEMBLER (MACRO)
COMPILER—SPECIALIZED FOR APPLICATION
LOADER (LINKING, RELOCATING AND DESECTORIZING)
OPERATING SYSTEM
MATH ROUTINES—SINGLE AND MULTIPLE PRECISION ARITHEMETIC
HARDWARE FAULT LOCATION
SYMBOLIC DEBUG PACKAGE
SUPER CALCULATOR PACKAGE
ASSEMBLER
COMPILER { PACKAGE TO RUN ON LARGE COMPUTER

This package might run over a quarter million dollars to develop. Amortized over only 50 computers, a price of under $6,000 would cover both development and reproduction costs. Since the person who wants this package probably bought an expanded hardware system, this cost increment may be insignificant (approximately the cost of a 4K memory).

However, the point of the discussion at this point is not to insist on specific software items but rather to suggest that there are alternative methods of configuring software which may be more attractive both to manufacturers and to users.

Typical applications

It may be desirable at this point to step back and examine a few "typical" applications of minicomputers. Some of the characteristics of these applications may lead us to certain conclusions about desirable characteristics of the supporting software.

In the case of data acquisition and recording, the computer sits between a multiplexed A → D converter and a magnetic tape. The computer will typically send channel selection information to the multiplexer and supply a start convert signal. After the conversion complete signal goes on, the computer will read in the sampled data and store it in memory. At this point, the data may be limit checked, converted to engineering units, etc., but the processing per sample will likely be low. Fixed point, single precision operations are very adequate. After a memory buffer has been filled, this buffer will be written on magnetic tape while an alternative buffer is being filled with sampled data. The throughput rate will be from a few hundred to a few tens of thousands of samples per second. Channels must be sampled at a very precise rate, to avoid skewing of data. This requires either an interrupt timer or precisely written loops.

The amount of computation is relatively small

but the I/O rate is relatively high. (1,000–10,000 word/second composite rate). Standard I/O control routines probably are too "fat" and would not be used. The program itself would be written in assembly language. The user of such a system would need only an assembler, loader and debug package from the manufacturer.

Other applications involve communications, which is largely a case of character processing and control. Often the minicomputer controls peripheral devices on one side and looks across a phone line to a large computer on the other side. Some logical ability is required but almost no computational capability is needed. Again, an assembler, loader and debug package will suffice.

Instrumentation and process control applications will typically make use of larger memories and the computational capabilities furnished by a language like FORTRAN. Interrupt processing may become quite important. Some bit shuffling must be done. Monitor programs may be of considerable value. This type of user will generally wish much, much more than the preceding two users, although all may be using the same basic computer. Since the user purchasing minimum hardware is not forced to buy the additional memory required for the more complex application, why should he pay for software he doesn't need either?

Proposed standard software

A question which commonly arises is whether program development should be carried out in assembly language or compiler language. The compiler language commonly furnished for minicomputers is some version of FORTRAN, a computationally oriented language. Problems in implementation of these languages have led to complaints about object program consumption of memory space and of execution time. Furthermore, the nature of some applications, which deal with bits and bytes as operands, does not always make FORTRAN an attractive language, regardless of the quality of the implementation. Algol, Jovial and PL/I all have their supporters, and if these compilers were to run on a different, larger system, and produce code for the minicomputer, some of the implementation problems might be relaxed. As of now, only one minicomputer manufacturer is known to have provided any language other than a FORTRAN dialect for a minicomputer. Hewlett-Packard has made both ALGOL and BASIC available. It is not the purpose of this paper to make language suitability comparisons but rather to suggest that a language more suitable than FORTRAN for typical "bit-shuffling" applications can be found, and that the associated language translator does not have to run on a minicomputer with 8K memory and no mass storage. No manufacturer has taken this approach, although several DEC users are reported to be generating object code on larger systems.

Nonprocedural approaches such as report generators and decision tables might be expected to stir considerable enthusiasm among inexperienced users. Again, implementation in a common language on a larger system seems an attractive approach.

Another frequently used, but esthetically less attractive solution, is to drop from the universal to the particular to provide a series of compilers each accepting an application dependent language which presumably would appeal to a specialized class of users. A common syntax but a varied semantics might make this approach feasible.

Nevertheless, assemblers for 4K memories and FORTRAN compilers for 8K memories are well nigh universal. A desirable approach, mentioned above, is a translator running on a large- or medium-sized computer system and producing code for a minicomputer. This translator program itself could be written in a machine independent language and thus would be capable (theoretically) of running on several different systems. The point here is not which language, but rather, the concept of operating this translator on a larger computer system with facilities more appropriate to the task of program translation. This method of program translation has been validated by the manufacturers of airborne computers, who commonly provide translators which do not operate on the target computer.

The second point to be made is that debugging should be conducted at the symbolic level: octal or hexadecimal dumps of memory are not an adequate tool for the typical user. Source language debugging seems to be the appropriate approach even though it is space consuming. Furthermore, the real-time/on-line nature of many applications needs to be recognized and appropriate tools developed, perhaps in connection with interrupt service routines.

Desirable debugging techniques can often be achieved more easily when program operation is

interpretive: that is, the computer hardwired order code is used to simulate instructions which are desirable from a programming point of view. At the present time, fixed point multiply and divide, double precision add, subtract, multiply and divide and floating point operations are commonly furnished as subroutines. The programmed operators of the SDS 900 series went a long way toward relieving the user from writing calling sequences to subroutines which provide functions not available in the hardware. This is a typical example of how software can be supported by appropriate hardware features. The high internal speeds of computers today relieve the problem of speed reduction (20:1 to 50:1) connected with the use of interpreters. Five to ten thousand instructions per second is an adequate rate in many cases. It is my prediction that over the next few years, a microprogrammed approach to this problem will remain too costly for minicomputers.

Inasmuch as programmer time spent in program debugging is generally accepted as equal to time spent in coding, the present obvious emphasis on translators at the expense of debugging packages seems somewhat out of keeping with things as they are. This will be more true with inexperienced users.

The mnemonic and symbolic debugging package furnished by DEC for the PDP-8 series seems a step in the proper direction. The fact that many of us have survived with octal dumps for the past ten years does *not* mean that this is the appropriate approach to take. The concept of the small computer as a black box in a system should include not only the words "inexpensive," "reliable," "fast," but also "easy to use." In this sense, "easy to use" should encompass the effort to check out the program which enables the minicomputer to be a part of the larger system.

Other areas can be mentioned but the most important needs of manufacturers seems to be a psychological one: the awareness that the minicomputer will, within a very few years, be contributing a much greater part of the total main frame market than it is today. And a major market deserves major support. The importance of adequately supporting this type of system can be assessed by thinking of the background of the individual attempting to use it. Scientifically trained, perhaps, but certainly not likely to be a programmer, especially in view of the present and projected shortage in this field. The outcries of early System 360 users will no doubt seem infrequent compared to the outraged voices of those novices who don't realize that early vintage software isn't supposed to work!

Hardware enhancements

Some suggestions concerning hardware development should be advanced at this point, if only to satisfy the hardware-oriented person who has stayed with it this far.

I would like first to advance the heretical idea that 18 bits are far more desirable than 16 bits of a minicomputer word size. Those two additional bits are exceedingly valuable in the instruction word, as anyone would know who has participated in minicomputer design. Since instruction words are generally more frequent than data words it would seem reasonable to produce a computer with an appropriate word size for instructions, rather than USASCII data. Perhaps the first step toward an 18 bit instruction word would be to discard the generally superfluous parity bit. In any event, I would divide the 18 bits in the instruction word somewhat as shown below:

5 bits for op code
1 bit for indirect addressing (paging is still necessary)
1 bit for "interpretive mode" (op code used as an address in a transfer table)
1 bit for use of current page/base page
10 bits for address, yielding an adequate 1024 word page size

My concept of indirect addressing would permit the direct addressing of 65K words via the 16 low order bits of the indirect word referenced. However, the leading 2 bits of the indirect word also would play a part in generating the true address. These two bits would be used as follows:

00 this word contains the effective address
01 use contents of hardware index register 1 added to the low order 16 bits of this word
10 use contents of hardware index register 2 added to the low order 16 bits of this word
11 applying one more level of indirect addressing

This somewhat restricted use of hardware index registers is compatible with current concepts of paging in minicomputers, where the program is stored in one group of pages and the tabular data which is to be indexed is in a second group of

pages, which must be indirectly addressed. Non-memory instructions could be used for index stepping and testing.

The base page, of course, should be controlled by a programmable base page register. This is in keeping with a multilevel program structure, with each level of priority associated with a different base page.

Another point to be made is that most applications would be better off with 8K words of memory with a 4-8 μsec cycle rate than 4K words with 1 μsec cycle rate. Only rare applications demand the sub microsecond memory cycle found with some of the newer minicomputers. However, many applications require more than the standard 4K memory. Of course, present cost factors do not seem to permit this kind of tradeoff to be made. In this connection, I would like to insert a plea for a bulk read/write memory of modest dimensions (50,000 to 500,000 words) with access times ranging from 10 to 100 milliseconds. This device should be sufficiently low in price (ten to fifteen thousand dollars) to encourage wide usage. NCR appears to have made a major step forward on software development for their Century series by stating that the twin disk drives are an inseparable part of even the least expensive system.

Many minicomputers operate on numeric data which requires two or more words to express. Double precision fixed or floating operations are fairly common in some applications (process control, instrumentation). These can more easily be programmed, if when the "interpretive mode" bit in the instruction word is set, the program counter and other status information is stored in location A and the next instruction is taken from A+1. A is determined by adding some constant to the 5 bit operation code. This approach also permits "upward compatibility," if the minicomputer is the smallest number of a computer family and can provide instruction compatibility through interpretation.

A final word concerning hardware should not overlook what I consider the most important single area in minicomputer design: I/O. The minicomputer *must* be able to communicate with a great variety of I/O devices. A reasonable approach to I/O includes the availability of a priority interrupt structure with sufficient capability such that each device can, if necessary, cause an interrupt to a different starting location. This will reduce the response time of the computer to the interrupt by removing the need for programmed "sorting out" of the interrupt signal from array several alternatives.

Such hardware must be supported by a comprehensive I/O control routine which encourages the potential user by its simplicity. Hewlett-Packard appears to have more in the right direction with their BCS package.

I/O data itself should be capable of being transferred either to memory or to the arithmetic registers when operating under a programmed I/O scheme.

CONCLUSION

The advent of medium and large scale integration and of batch memory systems, as well as improved intra-computer communication facilities will all have great impact on the future size of the small computer market. However, the greatest determiner will be the ease of use, as determined by the interaction of hardware and software. Low price and reliability alone will not satisfy the user who expects a job to be done, a task to be accomplished. Frustration after delivery may determine whether there is a five thousand unit market or a twenty - fifty thousand unit market for minicomputers in the early seventies.