# The myth is dead—Long live the myth

*by* E. L. GLASER and F. WAY, III

*Case Western Reserve University*
Cleveland, Ohio

Although the field of computation has been of the most explosively expansive in the history of mankind, still we find certain basic ideas which were valid many years ago are still assumed to be valid today. On the other hand, some apparent truths that were discovered at the beginning of the computer era unfortunately have been lost in antiquity and must be rediscovered, not once but many times. In this paper we do not expect to be definitive but at least to examine some of these myths, the true and the false, and perhaps to find some that should be destroyed and to uncover others to prevent younger workers from having to rediscover what each of us has had to do several times ourselves. To compound our problem we have the fact that the electronic calculator of the programmable variety is now starting to encroach and has already well overlapped the capabilities of the minicomputer. "This new machine is a hybrid. It overlaps both the minicomputer and the programmable desk calculator. The standard machine containing 1000 locations is capable of containing a program of over 2000 steps. In addition, specialized tape units using modern cassettes are available. The basic machine comes equipped with a number sufficient to handle all of your temporary storage needs, will enable you to bring in new programming systems, and it will even have sufficient facilities to permit you to sort blocks of data. As a consequence this machine will meet the needs of office, laboratory, or accounting room! Its size is moderate and will fit on a reasonably large desk. Cost is under $18,000. As a special added feature, its basic instruction set is implemented by using the most modern techniques of microprogramming and, therefore, may be modified in the future as the manufacturer develops new classes of instructions that will be of use to the various customers of this excellent machine. Programming systems that take full advantage of the human oriented decimal arithmetic are already available with it and, furthermore, all registers can handle both numeric and alphanumeric quantities! For further information circle . . ."

Such an ad does not seem in any way unusual. It could have appeared in any one of the trade magazines such as "Datamation" or "Computer World" any time within the last year. It appears to be an interesting architectural structure, and a rather interesting compromise between a desk calculator and a general purpose computer. Obviously this is a made-up example. Further, it should be obvious to the reader that this is meant to lead you down some kind of a primrose path. For those who have not guessed or for those who are not old enough to have remembered, the machine described has a venerable history and it is known as Univac I. The only thing that we have done is to rescale it in terms of modern technology.

The purpose of the preceding bit of fairy story was strictly to bring out the fact that our concept of what is a small computer and what is a large computer, what is a mini and what is a desk calculator have undergone radical changes, not just in the last twenty years, but in the last five. For those that don't believe it they may go through the exercise that we have done. A more interesting one might be to examine some of the new minicomputer offerings that take advantage of electronic MOS memory. At our university we have recently installed such a machine that also has built-in floating operations and memory protection hardware. This particular machine is being used as a minicomputer to support a special laboratory. It happens to be satellited to a much larger machine. Imagine our shock and surprise when, at the time the programming system was being established, we found it has the throughput capability of what everybody in this audience was calling a large machine as recently as 1965. We have purposely not identified either the minicomputer or the machine we compared it to. We leave that as an exercise for the reader and the hearer since after making this rather shocking discovery we looked at other potential pairs in this little game and found that it almost doesn't make any difference which ones you look at.

The purpose of all this has been to ask the reader to re-examine some of the "truisms" that have been around. The problem with any unwritten law is that you don't know where to go to erase it. Within the field of computer systems we are in an unusual position of being still a very young field with unbelievable number of customs, old wives tales, etc., that bind us and many of us are not even aware of it. This particular example we feel is a good one since it starts to challenge the most common of all such myths. "There is economy in scale." This is true. There is. It is also true that as our understanding of computing grows our aspirations for more and more computing also grow. However, the above example raises the question of do we really need as much economy of scale, for example, for interactive time-sharing where the main purpose is simple programming and editing. Even in the case where we are developing programs for very large and demanding problems that do require the modern equivalent of the largest computers, the question can be asked "is it really necessary that humans interact to develop these programs with the largest machines or could we not by means of modern technology place a simpler machine in the hands of the user for this purpose and subsequently transmit the problem when debugged to the large machine if the large machine requires it?" Not that long ago most people would have felt that being able to afford a Univac 1107 or an IBM 7094 for each person that needed computing was an unbelievably optimistic dream. The question is, now that we can do it, is that the way we are going to go?

## SOME SACRED COWS FOR THE SLAUGHTER

The preceding sacred cow which was exposed is not the only one. In fact, it is the most commonly referred to myth of our field. At the time we were originally preparing this paper we identified four additional such canards that had to be examined and either accepted or rejected on their worth but not on faith. We find, fortunately, that at least one of these has been partially rejected. It applies primarily to the user of small machines. For years it was felt that it was possible to cut corners on small machines. One can have good debugging features and all kinds of aids for the programs in the large machine but since the small machine is so small one can cut corners and not give the user all of this "help". We are glad to say that this feeling no longer is rampant in the industry as it once was. Large machines can support large staffs. Small computers, minis, and programmable electronic calculators cannot support staffs at all and therefore must supply the help for the user. We regret to say, however, that the remaining sacred cows are still alive, healthy, and doing ecologically un-nice things to the programmatic landscape. The first of these is "well, I know the machine does have some problems in its architecture but we'll fix that with software". How many times have we heard that, either in those words or words similar to it? We would even suggest that perhaps some new terms be offered, that henceforth the software be called hardware, and the hardware should be called easyware. All of us know of systems where this particular concept was applied and the pieces were never picked up totally. There are even two or three machines that we know of, made incidentally by different manufacturers, where the pieces never were picked up. No manufacturer is immune from this particular set of comments.

The second of our three mythical animals for the slaughter is "of course computers operate in binary. That's how God meant them to, that's why he gave you two thumbs!" to paraphrase Tom Lehrer, who once pointed out that counting in octal was just like counting in decimal if you didn't have any thumbs. We might add that counting in binary is just like counting in decimal if you are all thumbs. We have always assumed in recent years that the machine must operate in binary, and we can store other representations such as alphanumeric and decimal in it if we so need because they can always be displayed on small indicator lights for the serviceman and after all the software will make sure it gets printed out on the printer. What can we say? Machines are finished and operating systems are finished but computing systems are never done. The programmable calculator people have learned this and are now delivering machines that are ready to use as soon as you uncrate them. Unfortunately, however, they are having to relearn what the computer industry learned some 15 years ago. They are having to deal with the problems of specialized functions versus generality and in some cases they are having to relearn the whole technique of algebraic scanning and interpretation. However, they have learned many of their lessons well. For those of you who don't believe it, go look at some of the modern programmable desk calculators that operate from an algebraic language. The fact remains that people still think in decimal despite the fact that many programmers pride themselves in being able to read in octal almost as well. This is not to say that binary manipulation is not required inside the machine. The question however, must be raised and re-examined as to whether it is better to compute in decimal when decimal answers are required. For those who think that this is a dead issue, when was the last time you tried to explain to a good hardnosed accountant why it was by putting his problem on a modern machine you could only come out with inexact answers?

The third sacred cow that should be examined carefully is that of secondary storage. The cache memory has come into quite wide acceptance over the last several years. Yet, most machine designers insist that secondary storage disk memories and drums, are really input-output. We might humbly suggest that the core or MOS memory is really a cache for the larger online store. The implications of this are broad, however, they should be examined carefully since ultimately if you want online storage, you've got to be able to get at it in some reasonable way and an integrated approach rather than the massive peripheral attack that has been mounted upon the problem of memory, memory organization, and name space, just might be fruitful.

One of our most widely regarded cows has apparently been strangled by a twisted pair of wires—the cow was "closed shop is better" and the demise is being caused by remote terminals running in either batch or interactive mode. It is indeed marvelous to view the influence of state of the art technology (a pair of wires) making such a pronounced change in the management of computation centers. Another beastie which is long overdue for extinction is the practice of letting circuit designers dictate the arithmetic properties of a machine. This has produced such anomalies as a machine which does not have a floating point multiplicative identity—try and explain to an irate user just why one times X is not X. and yet another marvel wherein raising a real number to an integer power is more accurately done by exponentiation and logarithms than by successive machine multiplication. To add further fat to the sacrificial fires we now have shirt pocket calculators with nine or ten DECIMAL digit floating point (+99 to −99) numbers, trig functions and inverses, etc. It clearly is high time for some specialists in numerical mathematics (note—not numerical analysts) to have a say in things.

As a concluding blast in the large scale animal division, let us merely note some atrocious characteristics of some of the higher level languages such as letting machine "features" to percolate thru to the point where the user *must* know about them, or languages which have made the compiler writer's job simple at the expense of the target user—again try to explain just why the user should not write

```
DO 403 X = -9.35,0.45,0.33
```

Of course we can get around the problem, but some users seem to think that the machine should help them solve problems rather than the other way around.

## SOME SACRED CALVES

In this section we would like to put forth some sacred calves that have not had the chance to grow up yet. It is our fervent hope that they too someday may also be cluttering up the technological landscape and a latter day programmatic Don Quixote may sally forth to do battle with them. Our first candidate is already becoming well accepted. Don't put it in software if you know it isn't going to change. Put it in the hardware, it's easier to debug. If it is in the software, it's there because it needs to be parameterized and will change, either within one user's environment or between users, and change rather drastically. A parenthetical comment might well be inserted here that often is put in software for a somewhat different reason, namely "we don't really understand it so we'll give it to the programmers."

A second small critter has to do with microprogramming. Microprogramming really isn't an answer, it's a question. The question is, what is the place of interpreters? There is nothing basic about a microprogramming system. It is just that the old concept of interpreting in real time has been rediscovered. What is its future? Will we start seeing languages again that are interpreted rather than compiled and executed? Obviously we already have. A very well-known one is APL. Excellent work is going on in several centers including Harvard on languages and language systems in which both interpretation and compilation can be exploited quite interchangeably and continuously.

Our third candidate for immortality as bovinus mythicalus is in many ways the most important. A number of networks have sprung up in recent years. The work currently going on between a number of centers on the ARPA network may well be one of the most important information handling experiments that have been conducted in the last decade. Obviously it is of interest to demonstrate what can be done with long range, broadband communication that ties together a number of computing centers for load-sharing, for gaining access from one center to many other centers, and even more importantly, for making available computer resources at a distance so that each center need not have their own machine, but buys the service as they do "electric power", to cite a common cliche. The importance of this experiment may well transcend any of these reasons. As speed of devices increases the problems of the interconnection increase also, but at this point they are increasing at a much higher rate. We are rapidly approaching the time when it will no longer be possible to increase the speedy machine merely by increasing the speed of its elements. For those of you who are interested in circuit and hardware design, pray consider the dilemma if we were to offer you an unlimited supply of absolutely free components that were capable of operating in the one to two pico second range. These components, however, are sized about the same as our present IC dual-inline packages. What difference would they make? The answer is not much! We are already at the very edge of what present day packaging can do with present day organizations. Still, if we can network machines together across the country and have them work on some kind of a cooperative task, then obviously we can do it across a cabinet or even a PC board. This is not quite the same as array processors such as ILLIAC IV, rather it is looking to the time of learning how to use cooperative independent processes for the solution of large problems.

## CONCLUSIONS

It is hard to say what will happen if we either follow the new dictums or keep the old. Prognostication in this field has been woefully poor. The majority of it has either been overly pessimistic or just totally off the beam. There is one conclusion that can be drawn. Whether these new dictums are the ones to be adopted or not, the fact remains that computer systems design is becoming much more complex than it ever was before. Systems design must do quite a bit more than merely putting software on existing hardware. What is needed is an integrated set of design tools aimed at solving users problems and meeting users needs. Where possible, the users needs should be anticipated since user behavior will change with the advent of new tools and the availability of new techniques. It is our firm belief that it is this specific problem of being able to handle the complexity of modern systems that has caused the rather noticeable slowdown in the change and design of large computers today and it is the lack of this inhibition that has promoted the burgeoning minicomputer industry. Computers have been designed as tools to handle large and complex problems. They have been applied by many different industries, not to just the implementation of technological solutions but to the investigation of complex technological design problems. Unless and until the computer industry adapts its own tools and finds workable mathematical models that will permit the handling of the inordinate complexity of modern day computing systems, the stagnation that has appeared in the medium to large scale computer area will continue and these medium and large scale computers could ultimately be swept away by new, small, high-powered minicomputers aimed at individual use or for the use of a few people together with cooperative networking techniques. It is our fervent hope that this does not happen. However, hope is insufficient and the change can only come with identification of the valid problems to be attacked and a conviction as to where the real design aims are coming from. Are they coming from the designer or are they a part of a set of myths? Based partially on reality but primarily having their roots in lack of understanding, lack of awareness of the user, and lack of that most essential of all ingredients, the willingness to look at a problem in a new way and understand that the world has changed, the real design aims must be examined carefully and before the system is unleashed on the innocent users.