EDP ANALYZER

Editor and Publisher: RICHARD G. CANNING

EDP ANALYZER

DECEMBER, 1981
VOL. 19, NO. 12

PORTABLE SOFTWARE FOR SMALL MACHINES

Due to the substantial investment required to create software, coupled with the rapidly changing capabilities of mini- and micro-computers, there is a growing belief that software portability for these machines should be given a lot more attention. Recent announcements by Xerox, Wang, and IBM supporting the CP/M operating system substantiates the importance of the portability of operating systems. But what about the languages? We investigated several of the major languages being used on small computers to find out just how portable the software is now, and is likely to be in the near future.

Texas Instruments Inc. (TI) is a major manufacturer and supplier of electronic components and equipment, semi-conductor chips, mini-computers, hand calculators, and digital watches. Last year, sales exceeded $4 billion and the company employs almost 90,000 people worldwide. Headquarters are in Dallas, Texas.

In 1977, TI conducted a most interesting experiment. It gave TI 59 hand-held programmable calculators (valued at $300) to more than 10,000 of its professional employees. The project included formal training, a software exchange program, programming contests, and a newsletter. Although the experiment was expensive, it re-couped its costs within two months, said a TI spokesman, through more productive employees.

Following this successful experiment, management became intrigued with increasing the productivity of even more of its employees, through the use of small general purpose computers. The group that ran the calculator experiment was given funding to study, recommend, and implement this new project. This group is the Personal Computing Internal Fanout (PCIF).

The project is independent of the corporate information systems divisions; however, one goal of the project was that the personal computers be able to interface with the existing computer centers. PCIF was given funding for the study, but they were not funded to purchase computers or provide training for the employees. So the success of the project depended on 'selling' their solutions to TI department heads.

ISSN 0012-7523.

The decision was made to see if the use of many of TI's small computers was viable. Applications deemed most appropriate for this approach were identified as: interactive, using local data, requiring fast turnaround, for local distribution only, and short lived (even one-time applications). The major requirement for this approach was appropriate software—in particular, software that could be used by TI employees who have little or no computer experience, who have widely varying problems to solve, and who would be using various types of machines. The ideal solution would be a 'system package' that would appear consistent across all of these TI machines, would be machine independent, would provide powerful end user facilities, and would not be obsoleted by new generations of hardware—in short, a portable software system.

In late 1979 the project team decided to design their portable software system around the UCSD p-System, from SofTech Microsystems in San Diego, California. The p-System is based on the UCSD PASCAL programming language—although it has compilers for FORTRAN and BASIC, as well as UCSD PASCAL. The UCSD p-System was created to allow programs written for one computer to be transferrable to another computer. With this system, *everything* gets transferred—application programs, operating system, utilities, and compilers.

The requirements for portability are that each computer have: (1) an appropriate peripherals-handling subsystem, and (2) a 'p-code' interpreter, which translates p-code (an intermediate code produced by the compilers) into the computer's individual instruction code. Therefore, the PCIF group wrote p-code interpreters for the various TI processors (about two work-months of effort per processor) plus they wrote peripheral handlers for the various configurations they planned to support. When new peripherals appear on the marketplace, such as the mini-Winchester disk drives, it takes less than a week to make the required changes. The group now has p-code interpreters and peripheral handlers for TI's 99/4, 771, DS-1, DS-2, DS-4, TM 990, OOF, and DX-10 systems. Programs that run on one will run on all of these systems.

Since the p-System is a popular commercial product, PCIF has been able to take advantage of a growing number of packages that work with it. In fact, some of the software they offer has been purchased from third-party suppliers—COBOL and FORTRAN compilers, sort utilities, PERT/CPM packages, and so on. To this, PCIF has added internally written programs for matrix and linear mathematics (along the lines of the popular VisiCalc electronic spreadsheet package), business graphics, database management, and communications (to permit communication with other computers in the company.)

As mentioned, when a user wishes to move to a new computer which has a p-code interpreter, all of the software is moved in p-code form—the applications, operating system, and compilers. No re-compilation or other modification by the user is needed. The p-code interpreter takes care of the differences between the two computers. If the physical media (such as floppy disks) are compatible, the media from the old system can be used directly, so no assistance from a programmer or systems expert is needed. If the peripherals are not compatible, a new peripherals handling routine needs to be acquired.

The PCIF product was announced in September 1980. By December 1980 they had installed 55 systems, costing from $9,000 to $15,000 each. They had also received numerous requests for custom software work to complement their basic system. By the end of this month they expect to have installed 200 systems within TI.

The people at TI have learned a number of lessons from this project. First, software standardization is much more feasible than hardware standardization—which they consider to be nearly impossible due to technological changes and users' diverse needs. Second, language standards are not really necessary with their approach. As long as the intermediate code (p-code) from the compiler worked once, it will continue to work properly, while language conventions are permitted to change.

Third, the portable software system needs to be accepted by the marketplace, so that outside packages can be acquired; developing all applications for small machines in-house is "not the way to go." For example, at TI the PCIF group

only spends 30% of its time on new software development and acquisition of packages. Another 60% is spent in customer support and marketing, and the remaining 10% is for the creation and maintenance of the p-code interpreters and peripheral handlers. The project would not have been economically feasible without the high amount of software portability that they have achieved, they told us.

Software portability issues

Micro- and mini-computers are being heavily touted for personal use, small business use, and even for specialized and stand-alone applications in lieu of conventional mainframe data processing. We see these small systems becoming increasingly important to data processing management.

The term micro-computer currently implies a computer with disk storage capacity of 100,000 to five million characters or more, and an internal memory of generally 32K to 64K bytes. Mini-computers are larger, ranging from (say) five million to 200 million characters of disk storage, with 64K to 500K of internal memory. The proliferation of these smaller machines in business is causing growing concern about software portability. In this report we will concentrate on the micro-computer field, since it is more likely to be unfamiliar to data processing management.

Software portability can be described as vertical or horizontal. Vertical portability, familiar to all data processing managers, refers to compatibility of software within a manufacturer's line of equipment. IBM's System 360 and 370 families serve as typical examples, as do their smaller systems—System/3, System 32, 34, and 38. In these latter cases, programs written in RPG can function on any of the four small systems with few, if any, modifications.

Horizontal portability implies use of programs on different brands of hardware. Developers and distributors of software for micro-computers have had to address this area. And it behooves purchasers of small systems to become familiar with the few standards that are now available in this field.

Operating systems, programming languages, application programs, compilers, interpreters, peripheral interfaces, and storage media all need to be examined individually and collectively to determine portability from one computer to another. An application program may access data files in a manner that only selected operating systems can support. Or a BASIC language compiler may be compatible only with certain operating systems.

Micro-computers display the best example of horizontal portability through their operating systems. Many installed units use the CP/M operating system originated by Digital Research. And many compilers, interpreters, and application programs offered through computer stores and mail order houses are CP/M-compatible. Mini-computer users cannot say the same. A large number of mini-computer software systems, if they are portable at all, are only vertically portable. This approach by mini-computer manufacturers with large families of machines is easily understood: if a user wishes to upgrade to a more powerful machine, the supplier prefers it to be one of their machines. If the user wants to switch to a competitor, the non-portability of the operating system and utilities may be a deterrent.

Unfortunately, in the micro-computer field, a trend toward divergence of operating systems appears to be starting. With the advent of multiple-user micro-computer systems, networks for micros, and the new 16-bit micros, new, more powerful (and incompatible) operating systems are being pushed. It remains to be seen whether the software producers and distributors will be able to maintain the same level of operating system standardization as exists today.

Languages are variably portable. The proliferation of BASIC dialects causes many conversion problems. Although conceptually the dialects should be inter-changeable after only a few modifications, in fact, conversion is rather troublesome. COBOL is one of the most standardized languages and its programs migrate more easily. Assembly languages usually are machine dependent, but most micro- and mini-computer application programs are not coded in this level of language. However, even assembly language

programs are made easier to migrate if input/output is done through CP/M calls, we are told. FORTRAN programs theoretically should be very portable but may not be in actuality. Some dialects of PASCAL have been developed with both horizontal and vertical portability as a prime concern. We will look at several of these languages—and their portability—shortly.

Language portability can also be viewed by type of code. One type is source code portability. This means that the program's source code can be moved from one computer to another, and the compilers will translate the program into equivalent object code programs.

Intermediate code portability refers to code that has been 'semi-compiled,' and perhaps compressed, producing an intermediate code. To be usable, this code requires a run-time package. This type of portability is becoming increasingly important in the micro-computer world, because the code cannot easily be translated back into source code, so it discourages plagiarism and resale.

The extreme in software portability is object code portability. Here the compiled code will run directly on several machines. We have yet to see much object code portability—most so-called object code portability is actually intermediate code portability.

Most portable application programs depend upon the source code and its compiler. In addition they may be further restricted by input/output devices, file accessing procedures, operating systems, and utilities; here is where portability seems to encounter the most difficulty. If the program is a straight-forward calculation, as in a tax or interest subroutine written in COBOL, or if it was developed for micro-computers with limited input/output capabilities and file handling, it may be easily moved. But if extensive file or database management is involved, it may be difficult to modify the program for another computer. (The use of database management systems on micro-computers, although rudimentary, is growing.) System peripherals, in particular, are a common roadblock to portability of application programs, as we discuss later. Two other potential obstacles to portability are differences in media (floppy disk size, format, etc.) and input/output drivers.

The benefits of portability. Software portability is important to suppliers as well as to users. Programs are becoming increasingly expensive to develop. And developers of programs for small machines can only expect to obtain a profit if their programs can be run on numerous machines. Users benefit by paying less for such products. And off-the-shelf software is immediately available for the user—something not possible with in-house developed applications.

Software, particularly application programs and languages, outlives hardware. System expansions can be economically justified for new applications, but re-purchasing or re-programming existing applications negates such moves. So vertical as well as horizontal portability is desirable, to take advantage of future innovations in hardware.

Another advantage of portable software is flexibility for users. This type of software allows them to choose among several brands of computers based on price/performance, without being so dependent on the supplier's application software. To date, software distributors have encouraged portability in the micro-computer field; it remains to be seen whether the larger hardware manufacturers will hamper portability in the future.

Large data processing departments can reap benefits from small systems portability. If multiple micro- or mini-computers are installed for different applications, yet use the same operating system and language, training of personnel is simplified. Also, fewer variations of application programs are needed, with a consequent reduction in maintenance problems.

Software portability has always been an issue, but with the increasing use of small machines, it is becoming of paramount importance. The small machine software field is different from the mainframe field. To illustrate that point, we look at several languages used on small machines, to see just how portable they are.

Micro-computer Basics

The portability of software for micro-computers using BASIC is a complex issue. While the

products involved are well defined and understood—the application programs, language compilers, and interpreters—the factors that determine their portability encompass diverse areas. These range from the distribution and marketing channels to the actual syntax chosen for use in the BASIC language. We will look at these two areas that influence the portability of micro-computer BASICs. This discussion is based on numerous articles in two publications for micro-computer users—*Infoworld* and *Lifelines* (References 1 and 2).

BASIC began as a programming language for non-computer personnel. In 1964 two Dartmouth professors developed Beginners All-purpose Symbolic Instruction Code for use by their students on the university's General Electric 225 time-sharing system, and thus a new, interactive computing language was born.

The convenience of entering a program one line at a time, with on-line editing in a conversational environment, made BASIC fun and easy to use. It's acceptance was immediate. Many manufacturers, especially Hewlett-Packard and Digital Equipment Corporation, added extensions to the language, and offered their own versions. What followed was a sprint to see who could produce the most powerful BASIC, with little regard for compatibility with other versions. This resulted in BASIC becoming one of the most highly used langauges in the computer industry, and at the same time, the most non-standard. A class of languages evolved, instead of one expanded version. But its popularity never suffered. Time-sharing services became prolific in the early 1970s, and the language that they most often supported was BASIC.

In 1975 an event occurred which catapulted BASIC from the environment of low-cost time-sharing to home computers: the first micro-computer was introduced. As a result, personal computers pushed BASIC's position from immensely popular to the most widely used language in the world (certainly in terms of the number of people programming in it).

Distribution channels. Quick acceptance and the growth of micro-computers caused the need to insert standards which previously had been disregarded. BASIC versions had proliferated at an alarming rate, but primarily under the auspices of time-sharing companies or large computer manufacturers. The new micro-computer users did not usually work in data processing departments, with established departmental standards, nor were they buying time from a large company with an army of support personnel. They needed programs to run on their tiny-but-powerful systems. Thus a new type of distribution network evolved to provide this missing link; it contains software package producers, software distributors, and computer stores. All three have influenced the portability of the software on the market.

Portability of the numerous BASIC dialects stems less from similarities of different versions than from this distribution scheme that emerged to service the micro-computer users. Software producers developed packages, and by doing so, they standardized the versions of BASIC. Two of the largest software producers in the micro industry, Microsoft and Compiler Systems, offered MBASIC and CBASIC, respectively. These enhanced portability, as we discuss shortly.

The software programs developed by these and other producers are marketed by both mail order software distributors and computer stores. These outlets, too, promote portability of both hardware and software, for they must offer products in useable and compatible forms to end users. For instance, they offer software via many floppy disk formats.

The use of the S-100 bus, and the development of the CP/M operating system, in popular brands of micro-computers, also proved to be early influences on the success of these distribution channels.

The *S-100 bus* is a multi-wire cable that is used to transmit information between components in the micro-computers that use it. It was developed for the first personal computer, the MITS Altair 8800, which used the Intel 8080 micro-processor (the first highly successful 'processor on a chip'). The Altair 8800 became so popular that other manufacturers produced plug-compatible components to work with the system. Thus the S-100 bus allowed standardized hardware boards—for memory, input/output, controllers, and processors—produced by differ-

ent companies to be used in one computer. The Institute of Electrical and Electronic Engineers (IEEE) has proposed industry standards for the S-100 bus, and numerous manufacturers conform to the anticipated requirements, but it still does not totally command the marketplace. Some of the more popular micros that do *not* use the S-100 bus are Apple, TRS-80, PET, Altos, Lanier, Wang, Xerox, IBM, and most LSI-11 computers.

Paralleling the portability of hardware through the S-100 bus was a software counterpart, the *CP/M operating system.* It was developed by Gary Kildall, who set up Digital Research of Pacific Grove, California, in 1976, to market it, again for use with computers based on the Intel 8080 micro-processor. Its intent was to offer a machine-independent operating system. It is now the most widely used operating system in the micro-computer industry. It can only be used with the 8080, 8085, and Z-80 families of microprocessors, but Microsoft of Bellevue, Washington, has cleverly extended its use to Apple computers (which use 6502 micro-processor chips) by producing a Z-80 printed circuit board that can be inserted into an Apple computer, thereby allowing it to run CP/M-based software. Similar products are appearing for other micros. But this does not mean that the term 'CP/M-compatible' refers to total portability. The application packages using the newer versions of CP/M for 8-bit and 16-bit micros are not necessarily compatible with the more limited older versions, and vice versa.

The original CP/M was intended for single-user 8-bit micro-computers. But the new 16-bit micros, micro networks, and multiple-user systems appear to be diluting the effect of this early standard, without necessarily replacing it with new standards. Versions of Digital Research's multiple-user operating system MP/M and Bell Laboratories' UNIX are touted as future standards. But it is too early to tell. These are times of dramatic change in the micro-computer field.

Software distributors and developers are most concerned with preserving a few standards. They will only develop and offer their products in conjunction with widely used operating systems or versions of BASIC. Generally if a user purchases a package using, say, CBASIC and CP/M and wishes later to move it to another computer with these two components, the distributor can offer the same package for the second system. However, the packages are not usually user-portable, because distributors and developers worry about piracy. So often the user must purchase the different version of the package in order to migrate.

Distributors have found that supplying portability is increasingly difficult. The lack of standards in system peripherals is especially a problem. When floppy disks all used single-sided, single-density, and 8-inch diskettes, the distributors could use the IBM 3740 format as a standard format for program distribution. Now double density, dual-sided recording, and the new 5-1/4 inch diskette size, are being used by more and more micro-computers. This diversity has forced distributors and computer stores to provide either conversion services or a large number of recording formats for their packages. In fact, this may be the most important service that the software distributors perform for users. One new company, TeleSoft, in San Diego, will distribute software by data communications because of this problem.

Other system peripherals have also caused portability problems for distributors, but some software package producers now provide help. They imbed a menu of options for terminal and printer selections in their programs. The user selects the appropriate input/output devices, which does make these products more portable.

Language portability refers to the ease with which source programs may be made to run in another dialect of BASIC, as in a change from CBASIC to BASIC-80. The need to convert from one version of BASIC to another will most likely be caused by hardware expansions, or upgrades to different operating systems. The hardware or system selection will often narrow your choice of BASIC format. A discussion of two major versions—CBASIC from Compiler Systems of Sierra Madre, California, (recently acquired by Digital Research) and BASIC-80 from Microsoft of Bellevue, Washington—will illustrate some of the problems that may be encountered.

Microsoft BASIC (MBASIC) was first available through the use of interpreters. An interpreter

processes a source language on a line-by-line basis. Few offerings of business applications written in MBASIC were made by software distributors in the early days of micro-computers. For one thing, the risk of piracy was high—source code had to be supplied to the users for use with the interpreters, and it could simply be copied and re-used on other, compatible machines. (This understates the importance of MBASIC, since it has been offered on many popular micros and is often preferred by programmers for developing their own programs. However, in response to this problem, Microsoft later began offering a compiled version, BASIC-80.)

Compiler Systems then introduced CBASIC, calling it a psuedo-compiler. CBASIC originally produced an intermediate file (INT) which required a small run-time package in order to execute. Users needed to purchase CBASIC, with its run-time program, in order to execute the INT file. Programmers happily sold their applications partially compiled into the INT format with little fear of unauthorized (and unpaid for) re-use. In addition, CBASIC is now offered in a native code (full) compiler version.

CBASIC became, and still is, a very widely used BASIC dialect, and it is now available for the 16-bit 8086 processor and for use with the UNIX operating system. Interestingly, Microsoft's licensing policy enhanced CBASIC's popularity, because developers must pay royalties to Microsoft for application programs that use their BASIC-80 compiler. For this reason, many authors have chosen to stay with CBASIC.

CBASIC is the least transportable of any CP/M BASIC, due to its syntax and conventions. Unlike other BASICs, line numbers are not required, except as targets for GOTOs and GOSUBs, and even then they need not be in any particular order. They can even be expressed in exponential notation or floating point. Also, certain keywords are unique to CBASIC, with their functions expressed in different terms in other BASICs.

Microsoft now offers both a BASIC-80 interpreter and compiler. The interpreter is best used when creating and testing a program. Once it seems to be working properly, it is wise to compile it in order to produce code which runs faster. The interpreter and the compiler must be purchased separately. The portability of BASIC-80 is good, although it only operates under CP/M. Users planning to upgrade to a larger machine for expanded application work should be aware of size limitations with BASIC-80. It has not been suitable for use with large source programs. In the past, the COMMON statement has not been supported, although this may be changing; this limits program size to the amount of available memory, or causes the need for intermediate results to be written to disk. Speed is degraded and programming is cumbersome under these circumstances. Yet it is a popular BASIC and is found in many application packages for micro-computers.

Up to now, using one of these more popular version of BASIC along with CP/M has provided a good deal of portability. But the future is cloudy.

Micro-computer Cobol

As might be expected, due to its extensive use in the mainframe and mini-computer environments, COBOL is now emerging as a major business language for micro-computers. A number of COBOL compilers for micros, based on the U.S. government's ANS-74 standard, are now available.

In the mainframe environment, this COBOL standard has led to source code portability. Yet the standard does not insure portability on small machines, as Howkins and Harandi (Reference 3) point out. They explain that the 1974 standard consists of a nucleus and eleven functional modules. Each module has two or three levels. Full COBOL contains all of the features. Minimum COBOL contains Level One of the nucleus, plus table handling and sequential input/output modules. Subset COBOL contains any combination of the levels of the nucleus and other modules. For both mini- and micro-computers, the authors report that there has been no minimum standard defined. The COBOL compilers for micro-computers advertise their products as handling 'a subset' of the standard (the most undisciplined type, we gather). Howkins and Harandi state that, in total, COBOL can have 104,976 official variants—a significant barrier to easy portability.

Hogan (Reference 4) states that, since COBOL is not well suited to interactive applications, the

compilers for micro-computers often have terminal drivers embedded in them—and these are not standard from one compiler to another. In other cases, a separate driver is provided. In some cases the compilers create intermediate code (not true object code) which is executed at run time by a run-time program.

While this discussion illustrates a few barriers to portable COBOL application programs on small machines, the people at David R. Black and Associates, who offer a COBOL program generator, believe that portability of COBOL programs can be increased in the future. They see the possibility of including an operating system, a compiler, and a program generator on one or a few chips. This entire ensemble could then be moved (along with the application programs) among different hardware configurations. We assume this approach would provide intermediate code portability; in this case, a machine-dependent run-time package would be needed to execute the programs.

So why is the use of COBOL on small machines growing? The rationale described to us at Tandy Corporation pretty well tells the story.

Tandy Corporation

Tandy Corporation is a leading manufacturer and retailer of consumer electronics products with headquarters in Fort Worth, Texas. In fiscal year 1980, Tandy had sales of $1.4 billion, of which 12.7% came from computer sales through their subsidiary Radio Shack. Radio Shack markets the TRS-80 family of micro-computers.

The TRS-80 family consists of six computers. The Model I is based on the Z-80 micro-processor chip. It has from 4K to 48K bytes of memory, and it can be interfaced with a printer, communication facilities, a cassette recorder, and up to four 5-1/4 inch single or double density floppy disks (for a total of over 600K bytes of storage). Prices start at $500, and Radio Shack says they have sold over 200,000 Model Is since 1977. Production of the Model I in the U.S. was discontinued in December, 1980.

In 1980 Radio Shack announced the Model III. It is an upgraded version of the Model I with two built-in 5-1/4 inch floppy disk drives and a smaller-than-standard CRT screen. It is aimed at personal and small business users.

Radio Shack also recently announced: (1) a color computer which provides color graphics capabilities; (2) a pocket computer, slightly larger than a hand-held calculator, which has 1.9K bytes of memory and is battery operated; and (3) a videotex terminal for use with any telephone and television set for two-way information retrieval from viewdata-like services.

The most powerful member of the TRS-80 family is the Model II. Based on a Z-80A chip, it has 32K to 64K bytes of memory, one built-in 8-inch double density floppy disk drive, and a standard-size CRT screen. It can interface to three more eight-inch floppy disk drives (for a total of 2.4 megabytes of storage), a printer, and communication facilities. Prices start at $3450.

Up until one year ago, Radio Shack offered only Assembler, FORTRAN, and BASIC programming languages. But last year they began offering COBOL for their Model II and Model III machines.

The company chose to offer COBOL for several reasons. First, COBOL is an accepted business language and Radio Shack is increasingly aiming at the business marketplace. Second, ANS-74 COBOL is very portable, due to the U.S. government's validation program for the compilers. Third, COBOL is more 'under control,' due to these ANS-74 COBOL standards. Radio Shack sees BASIC standards as inconsistent and therefore lacking portability.

Fourth, micros have now become large enough to support respectable COBOL compilers. And fifth, as micros become even larger and more powerful, Radio Shack expects COBOL to become *the* standard business language, just as it is on larger machines. One additional major reason is that there are so many COBOL programmers in the world.

The Radio Shack COBOL compiler contains a subset of the standard ANS-74 compiler functions for mini-computers. It supplies syntax checking, interactive debugging, screen input/output, and keyboard control compatible with the larger compilers. Just about the only thing missing is the SORT capability, we were told, and this is

partially offset by the multi-key index-sequential capability.

Currently, Radio Shack writes about 60% of their own software, and obtains the other 40% from outside sources. For in-house development, they use both TRS-80s and a Tandem computer.

In order to enhance portability among their current and future offerings, Radio Shack has standardized not only on ANS COBOL for business applications, but also on other system aspects. For example, they are instituting a screen interface package, which all Radio Shack programs must use. This will cause all screen functions to appear the same, not only to programmers but also to users.

Radio Shack will soon have a COBOL program generator both for sale and for development use. The program generator is also intended to enhance software portability, because programs created using it will have the same structure. Radio Shack expects it to speed up development and ease maintenance. And they note that use of the generator by third party software houses will help enforce the Radio Shack standards. For example, the generator will employ the screen interface procedures mentioned above.

Radio Shack sees COBOL, along with a standard operating system, database management system, and program generator, as enhancing the portability of their application programs, at least within their own line of micro-computers.

Pascal

One relatively new language that has generated a lot of interest, especially in the microcomputer world, is PASCAL. It has rigorous structured programming conventions, and one dialect in particular—UCSD PASCAL—has been used to promote software portability. Since this language is not well known to users of mainframe computers, we will discuss its approach to portability in some detail.

UCSD Pascal

The 'conventional' approach to portable software is to write the programs in a common, standard language—such as COBOL or FORTRAN—for which compilers are available on numerous host computers. In addition, with COBOL, the data definitions are explicit, making it more likely that data files can be moved to a different computer without undue effort.

There are some difficulties with this conventional approach. The compiler on the second computer may develop code that does different things from the code on the first computer. This is one of the main reasons that the U.S. Navy has been conducting tests and audits of the COBOL compilers that are offered to the U.S. government. The second computer generally will have a different operating system from the first one, making the user interface different. And if the data is stored under a database management system on the two computers, there is a good chance that the two DBMSs are sufficiently different that program changes will have to be made.

UCSD PASCAL has taken a quite different approach to this question of software portability. This approach is to move *the whole environment*—operating system, utilities, compiler, assembler, text editor, file handlers, and application programs—from one host computer to another.

The UCSD software system has itself been written in PASCAL. These programs are then compiled to produce code in an intermediate language, called 'p-code.' (An assembly language and an assembler, plus FORTRAN and BASIC compilers, have also been developed; these, too, create p-code.) To run this p-code on a host computer, an interpreter is needed to translate the p-code into the host's native code. Thus, for each type of host computer, a p-code interpreter is needed. Once the interpreter is available for a host, the whole environment—from operating system to application programs—can be moved to that host.

Proof of portability is furnished by the UCSD PASCAL system itself, because it is written in PASCAL. Since it runs on the various hosts, one would expect that application programs written in UCSD PASCAL would also run on each of the hosts. And this is, in fact, what occurs.

The system has an excellent screen-oriented text editor; we have been using it for our publications for over two years and think very highly of it. An important point to note is that the user

interface for this text editor is *identical* for all UCSD PASCAL host computers. As Dr. Kenneth Bowles, the architect of the UCSD PASCAL system, has said, "Try accomplishing this same thing with a text editor written in any other higher level language (such as COBOL, FORTRAN, or BASIC) that must run under different operating systems."

The UCSD PASCAL system is now available for some 20 host computers, most of them micro-computers. The micro-processors used by these hosts include the 8080/8085, Z-80, LSI-11, TI9900, and 6502 (used by Apple II).

There is a price that is paid for this approach to portability, of course. The use of an interpreter for the p-code slows down the net speed of the processor by a factor of perhaps 20 or so. Most users of UCSD PASCAL have used it in a single user mode on a micro-computer. In this mode of operation, frequently no delay is noticeable by the user. But because some users have noticed delays, and because concurrent (multi-user) processes have been requested, other approaches have been developed to speed up the system, as will be discussed.

History of UCSD Pascal

The UCSD PASCAL system was developed at the Institute for Information Systems, on the campus of the University of California at San Diego, under the leadership of Dr. Bowles. It was first run on a PDP-11 system in the fall of 1977, and by December of that year the text editor was running on an 8080 processor. The 8080 system was fully running by the following March.

Bowles wanted this system developed in order to meet several goals: (1) He wanted to use micro-computers in the campus' programming training lab, in order to reduce the cost of providing students with access to a computer. (2) At the same time, he wanted the students to use a 'big machine' language, as well as a language that was compatible with the concepts of structured programming (as PASCAL is). And (3) he wanted the system to be portable, so that as new, more powerful micro-computers became available, they could easily replace the existing micros in the lab.

As the existence of UCSD PASCAL became known, widespread interest developed. Micro-computer users in many countries wanted to use it. Micro-computer manufacturers wanted to offer it. So the university began licensing users—both individuals and companies—to use it. And fairly quickly, the Institute found itself in the business of licensing, supporting, maintaining, and enhancing the system.

University administration became concerned that this activity would bring the university into conflict with the taxing authorities. So they decided that the whole 'commercial' aspect of UCSD PASCAL would have to be contracted to a private company. In mid-1979, SofTech Inc., of Waltham, Massachussets, was licensed by the university to take over all further commercial licensing, support, maintenance, etc. of the system. SofTech set up a new subsidiary, SofTech Microsystems, located in San Diego, to conduct this business.

Improvement of UCSD Pascal

In the summer of 1978, Bowles convened a workshop on the campus to which all implementors of PASCAL compilers, and others involved with standardization efforts for PASCAL, were invited. Thirty implementors did, in fact, attend. The purpose of the workshop was to identify needed improvements and extensions to the language and to propose solutions that could lead to a draft standard for PASCAL. At the time, both ANSI and the IEEE had committees working on PASCAL standardization.

There was fairly general agreement among the participants on the desirable standarization efforts. This consensus of ideas has been used—first by UCSD and then by SofTech—in the continued enhancement of the UCSD PASCAL system.

With the signing of the SofTech contract, the Institute was directed by University administration to complete current development work then in progress, turn over the results to SofTech, and then wind down most of the Institute's work on the system. So there was a transferral of the development effort from the university to SofTech, during the last half of 1979 and much of 1980.

SofTech set for itself a number of goals, for the enhancement of the system.

Consolidated system. UCSD PASCAL, like any such system, has gone through a number of refinements and releases. The 'regular' system was given release numbers 1.0, 1.1, and so on, with version 2.0 released at about the time when SofTech took over. Also, Apple Computer had been licensed by the university to offer the system to Apple purchasers, and that company made some changes to the system. The Apple system was considered to be release 2.1, and was somewhat incompatible with release 2.0. Finally, Western Digitial Corporation decided to build a 'PASCAL Micro-engine' computer, in which the p-code interpreter was built into the hardware, to increase the speed. To lengthen the technological life of this approach, they sought and received permission to incorporate features not yet in releases 2.0 or 2.1. So the Western Digital system was considered to be release 3.0.

SofTech set the goal of coming out with release 4.0 just as soon as they could—and 4.0 was to pull together all of the features found in 2.0, 2.1, and 3.0. The 4.0 release of the system was made available in February 1981.

Additional languages. Even though PASCAL has a number of advantages as a programming language, the system could have much wider usefulness if other programming languages were also supported. So, in mid-1980, a compiler for FORTRAN-77 was released. Like the PASCAL compiler and the assembly language assembler, the FORTRAN compiler creates p-code. Then, in early 1981, a BASIC compiler was released. Additional languages are being considered.

New system name. Because of the additional languages, the name 'UCSD PASCAL' was no longer appropriate for the system. So SofTech has chosen the name 'UCSD p-System' as the name of the system.

Concurrent processes. As it was originally developed, the UCSD p-System could handle only one process at a time. For a single user at a micro-computer, this was not too much of an inconvenience, although users at times wished that they could 'spool' output at the same time that they were using the text editor on another file.

The handling of concurrent processes in portable software is difficult because each host processor handles interrupts differently. But the concurrent process feature was incorporated in release 4.0—although to use it, programmers must follow some rather strict rules.

Increased speed. As previously indicated, the interpreter approach proved to be too slow for some users, and would become more of a problem with concurrent processes. So SofTech considered several ways of speeding up the system (based on work that had originally begun at UCSD).

One approach was to put the interpreter in the hardware, in the form of read-only microcode. This was the approach used by Western Digital. It increases the speed of the system by a factor of about five, said a SofTech executive.

Another approach is to translate the p-code into the host's native code, which is then executed directly. A 'simple' translation would increase the speed by a factor of ten over the original interpretive speed, said the executive. And if the native code were optimized, the speed advantage would be about twenty, or double the simple translation. So this is the direction that SofTech has been working.

The UCSD p-System represents an encouraging example of language portability. Unfortunately we do not see other efforts in this area. In fact, we see more effort going toward non-standarization, with hardware manufacturers gaining influence in the micro industry.

We recently heard of a large government military agency that is looking to PASCAL, specifically the UCSD p-System, for office automation. They expect to spend under $5000 (a one-time cost) for a micro-based work-station for each of their office workers. To keep the cost this low, they first are writing applications to run under the UNIX operating system, because they see UNIX as being the prime operating system for micro-computers in the near future. The applications are being written in PASCAL, with all UNIX system calls highlighted.

Second, they are designing a PASCAL-based operating system. When it is completed, they will be able to change the UNIX system calls to PASCAL system calls and then move those applications to their PASCAL-based system. They expect this operating system to be portable across many types of future work-stations. So they are con-

centrating on getting their office automation software to outlive not only the hardware, but also the current operating systems.

There are several additional languages that we have not discussed; for instance, one is FORTH and another is 'C.' 'C' is closely tied to the use of the UNIX operating system; UNIX is written in C and users of UNIX often prefer to use C over other available languages. UNIX is too large an operating system to fit on the 8-bit micros; however, we understand it will be offered in a wide number of variations on both 16-bit and 32-bit micros. So although C is not now widely used, it may become important in the future. Also, FORTH has developed a fairly wide following, but we were unable to locate any business application users.

Challenges to portability

The UCSD p-System has brought into focus a number of challenges to portable software for small computers.

Unique features. Computer manufacturers typically do not support true portability; they tend to give more lip-service than adherence to it. What they generally would like to do is to add their own unique features to anything 'standard;' these unique features are designed to give them sales advantages.

What is more, if attempts are made to prevent them from adding these unique features, they may make charges of 'restraint of trade.'

To the extent that users make use of such unique features, the software loses portability.

Can one organization control? Can one organization satisfactorily control a widely used software system? To exert control, the organization must provide continued technical leadership. With many innovative users, how can one organization keep up with all the ideas that are generated, much less stay ahead of users?

How to handle the volume of work? With the many types of host computers on the market, who is going to write the 'interpreters' (or equivalent) for each such host? If the suppliers write this type of customizing software, can the controlling organization demand the right to audit that software for compliance with all standards?

How interested are users in portability?. When a supplier offers some unique features in conjunction with a 'standard' system, users are faced with a choice. They can get immediate advantages by using those unique features—but in so doing, they jeopardize portability. How will the population of users decide, when faced with this dilemma?

With companies just now trying to decide which small computer(s) they should standardize upon, we think portable software considerations should be high on the criteria list.

REFERENCES

1. *Infoworld* is a bi-weekly newspaper published by Popular Computing, Inc. (375 Cochituate, Box 880, Framingham, Massachusetts 01701); subscription price: $25 per year.
2. *Lifelines* is a monthly newsletter from Lifelines Publishing Corp. (1651 Third Avenue, New York, New York 10028); subscription price: $18 per year.
3. Howkins, T. J. and M. T. Harandi, "Towards more portable COBOL," *The Computer Journal*, British Computer Society, November 1979, pp. 290-294. Back issues can be obtained from William Dawson and Sons Ltd., Cannon House, Folkestone, Kent, Great Britain.
4. Hogan, Thom, "COBOL is coming," *Infoworld*, (address above), January 19, 1981, p. 11; price: $1.25 per issue.
5. For more information on the UCSD p-System, contact SofTech Microsystems, 9494 Black Mountain Road, San Diego, California 92126.

Next month, we will address the question of 'practical office automation'—which means "making the best use of what you already have." Some organizations have been doing a good job of moving toward the automated office in a planned, step-by-step manner. We will describe how they have been accomplishing this.

Then in February, we will discuss the use of computer graphics in business. Computer graphics are finally here, in an economical, practical manner. Useful graphics are even available on personal computers, and some large organizations are making use of these small machines to get a quick start in computer graphics.

COMMENTARY

SOME CRITERIA FOR CHOOSING A MICRO-COMPUTER

by Larry Press, Small Systems Group, Santa Monica, California

The portability of software has been the most important single factor in the rapid acceptance of personal computers for business and professional applications. For this reason, if you are thinking of acquiring a personal computer, you should take a close look at CP/M-based systems and the software which is available for them.

CP/M was developed by Gary Kildall to support a compiler he had written for the Intel Corporation for their 8080 processor chip. When Intel decided that they were not interested in a floppy disk operating system, Digital Research Corporation was formed to market the package. Currently there are 380 suppliers using CP/M, and it has been included on the Datapro Honor Roll. Many of the companies offering CP/M-based computers—for instance Vector Graphics, Altos, and North Star—were founded to market personal computers. Older companies, such as Xerox, Wang, Datapoint, and most recently, IBM, are now offering CP/M-based machines. A conservative estimate of the installed base is 300,000 machines.

The success of CP/M has attracted many independent, third party software vendors. My company publishes an index of this software and the current edition lists 740 programs offered by 248 vendors. The programs are organized into 76 categories—for example, word processors (27 programs), integrated accounting packages (29 programs), language processors (55 programs), and medical office packages (14 programs). There is a lot of software available under CP/M and the quality is, in general, quite high. Furthermore, many of the vendors do a good job of supporting their products.

That is the good news, but there are a few problems with CP/M software portability. Horizontal portability of both CP/M and programs running under it has been made relatively easy because all of the hardware manufacturers use either the 8080 or the upward-compatible Z-80 processor chip. On the other hand, they do not all use the same video displays or the same disk formats. This means that either you or your dealer may be in for a bit of conversion work. In the case of video displays, it is necessary to let the software know the characteristics of your system. If you use a popular terminal, it will probably suffice to select it from a menu provided in the CP/M package. In more difficult cases, it may be necessary to supply the program with escape codes from your terminal manual. And some memory-mapped displays may not work with a given package at all.

Disk formats present another, generally surmountable, problem. The only standard is the eight-inch IBM single density format, and virtually all CP/M software is available in that format. Even if you do not need the capacity of eight-inch disk drives, this standardization is a good reason to consider them. There are many different 5-1/4 inch disk drives on the market and no stan-

dard, so you may have problems getting a package onto your system using that size disk. Fortunately, there are a number of large distributors, such as Lifeboat Associates, who offer many vendors' software in a wide variety of disk formats. If a given package is not available on your disk format, you may have to transfer it to your system through a serial communications port or buy it from a dealer who will be responsible for the conversion. In either case, be sure to get a machine with at least one standard RS-232 communications port.

CP/M itself has also gone through two major versions, and version three will be out soon. A number of newer programs will run under CP/M 2, but not CP/M 1. Furthermore, there are now MP/M and CP/NET for the 8080 and Z-80. Again there are incompatibilities in some cases. And I would be careful in dealing with companies that advertise 'CP/M compatible' operating systems—be sure your software runs properly first.

The 16-bit processor chips are also introducing problems with software portability. The 16-bit Intel 8086 and 8088 chips are now turning up in a number of machines (most importantly the new IBM personal computer), and the same will be true of the Z8000 and M68000. CP/M is already available for the 8086 and 8088, and MP/M will be soon. Digital Research has a program which converts 8080 assembly language (but not Z-80) to 8086 automatically. And currently, only two of the widely used higher level languages, CBASIC and PL/1 subset G, both products of Digital Research (which recently acquired Compiler Systems), are running under CP/M-86. However, fourteen other companies, including Microsoft, are writing upward compatible language processors to run under CP/M-86.

In looking to the future, two operating systems will clearly become important. First is IBM's personal computer DOS. While I have not yet seen more than a demonstration of the system, IBM promises that conversion of CP/M-based programs to DOS will be very simple. Furthermore, IBM DOS will doubtless become available on other machines. With the resources of Microsoft and IBM behind it, it will be an important factor in the near future.

The other operating system which will certainly play an important role in the more distant future is UNIX. UNIX, a multi-tasking system, is pervasive in the academic community. It is also written in a higher level language (C) and is therefore relatively portable. I know of fourteen companies that have, or are working on, UNIX or UNIX-like operating systems for the new 16-bit micro-processors. At the present time, there is virtually no installed base, relatively few vendors, and no channels of distribution. However, in a few years there will doubtless be a good deal of software which runs on many manufacturers' systems under UNIX.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1978 (Volume 16)

Number	*Coverage*
1. Installing a Data Dictionary	G
2. Progess in Software Engineering: Part 1	H
3. Progress in Software Engineering: Part 2	H
4. The Debate on Trans-border Data Flows	L
5. Planning for DBMS Conversions	G
6. "Personal" Computers in Business	B
7. Planning to Use Public Packet Networks	F
8. The Challenges of Distributed Systems	E,B
9. The Automated Office: Part 1	A
10. The Automated Office: Part 2	A,D
11. Get Ready for Major Changes	K
12. Data Encryption: Is It for You?	L

1979 (Volume 17)

Number	*Coverage*
1. The Analysis of User Needs	H
2. The Production of Better Software	H
3. Program Design Techniques	H
4. How to Prepare for the Coming Changes	K
5. Computer Support for Managers	C,A,D
6. What Information Do Managers Need?	C,H
7. The Security of Managers' Information	L,C,A
8. Tools for Building an EIS	C
9. How to Use Advanced Technology	K,B,D
10. Programming Work-Stations	H,B
11. Stand-alone Programming Work-Stations	H,B
12. Progress Toward System Integrity	L,H

1980 (Volume 18)

Number	*Coverage*
1. Managing the Computer Workload	I
2. How Companies are Preparing for Change	K
3. Introducing Advanced Technology	K
4. Risk Assessment for Distributed Systems	L,E,A
5. An Update on Corporate EFT	M
6. In Your Future: Local Computer Networks	F,B
7. Quantitative Methods for Capacity Planning	I
8. Finding Qualified EDP Personnel	J
9. Various Paths to Electronic Mail	D,M
10. Tools for Building Distributed Systems	E,B,F
11. Educating Executives on New Technology	K
12. Get Ready for Managerial Work-Stations	C,A,B

1981 (Volume 19)

Number	*Coverage*
1. The Coming Impact of New Technology	K,A,B
2. Energy Management Systems	M
3. DBMS for Mini-Computers	G,B
4. The Challenge of "Increased Productivity"	J,K,A
5. "Programming" by End Users	C,H,B,G
6. Supporting End User Programming	C,H,B,K
7. A New View of Data Dictionaries	G,B,
8. Easing the Software Maintenance Burden	H,B,G,
9. Developing Systems by Prototyping	H,B,G
10. Application System Design Aids	H
11. A New Approach to Local Networks	F,K
12. Portable Software for Small Machines	B,H

Coverage code:

A Office automation
B Using minis & micros
C Managerial uses of computers
D Computer message systems
E Distributed systems
F Data communications
G Data management and database
H Analysis, design, programming
I Computer operations
J Personnel
K Introducing new technology
L Security, privacy, integrity
M New application areas

(List of subjects prior to 1978 sent upon request)

Prices: For a one-year subscription, the U.S. price is $60. For Canada and Mexico, the price is $60 *in U.S. dollars,* for surface delivery, and $67 for air mail delivery. For all other countries, the price is $72, including AIR MAIL delivery.

Back issue prices: $7 per copy for the U.S., Canada, and Mexico; $8 per copy for all other countries. Back issues are sent via AIR MAIL. Because of the continuing demand, most back issues are available.

Reduced prices are in effect for multiple copy subscriptions, multiple year subscriptions, and for larger quantities of a back issue. Write for details.

Please include payment with order. For payments from outside the U.S., in order to obtain the above prices, *use only an international money order or pay in U.S. dollars drawn on a bank in the U.S.* For checks drawn on banks outside of the U.S., please use the current rate of exchange and add $5 for bank charges.

Editorial: Richard G. Canning, Editor and Publisher; Barbara McNurlin, Associate Editor. While the contents of this report are based on the best information available to us, we cannot guarantee them.

Missing Issues: Please report the non-receipt of an issue within one month of normal receiving date; missing issues requested after this time will be supplied at the regular back-issue price.

Address: Canning Publications, Inc., 925 Anza Avenue, Vista, California 92083. Phone: (714) 724-3233, 724-5900.

Microfilm: EDP Analyzer is available in microform, from University Microfilms International, Dept. P.R., (1) 300 North Zeeb Road, Ann Arbor, Mich. 48106, or (2) 30-32 Mortimer Street, London WIN 7RA, U.K.