The ACSS Newsletter

University of Minnesota
Twin Cities
April 1986

Computing Reflections

Seymour Cray's Machines (Part 1)

Lawrence Liddiard

Seymour Cray is known as the man who has been the chief architect of seven remarkable computers since 1958. The seven machines are the 1604, 160, 6600, and 7600 systems of Control Data Corporation (CDC) and the CRAY-1, -2, and -3 of Cray Research Incorporated (CRI), and all of them, except the 7600, were, are, or will be part of the computing resources used at the University of Minnesota.

The first director of computing at Minnesota, Dr. Marvin Stein, once told me about a conversation he had with a former Electrical Engineering graduate of the University in the late '50s. During that visit, the graduate, Seymour Cray, offered the University a chance to share in the development of a new 48-bit word computer made with transistors. Lacking the necessary funds (a common University problem), Dr. Stein turned down that offer, and Mr. Cray went on to be the architect of CDC computers.

Not withstanding this early rejection of Cray's work, the University of Minnesota acquired the CDC 1604 (serial number 50) and CDC 160 computers, new in 1962, the CDC 6600 (serial number 16), new in 1966, a slightly used CRAY-1 (serial number 12) acquired in 1981, and the CRAY-2 (serial number 3), new in 1985. The CDC 1604 retired in 1968, the CDC 6600 was replaced in 1974 by its twin sister the CYBER 74, which finally retired in 1983, and the CRAY-1 will be put out to pasture in 1986.

The CDC 1604 made major advances over previous machines, such as the 48-bit word length that gave floating point numbers a 10 to the 616 exponent range, compared to the 10 to the 76 range on competing systems, and the current IEEE standard single precision. In other respects the CDC 1604 combined elements of the earlier UNIVAC 1103, besides using memory banking and packing two instructions per word.

But the CDC 6600 was the first of Cray's major scientific computer advances: It included a simpified instruction set, register files, parallel operation both with I/O processors and arithmetic functional units, floating point arithmetic with infinite and indefinite values, a sequential instruction stack, and it only cost $7 million. In fact, the $7 million price tag seemed to be the right price during 1966-1976 for a leading-edge computer, since it also was the approximate price for the CDC 7600 and the CRAY-1. The CDC 7600 added the concepts of "pipelined parallel arithmetic units" (the precursor of vector units on the CRAY-1) and the associative instruction stack. A minor aberration was introduced on the CDC 7600 because it had both fast small and slower large core memories for single and eight-word loads/stores. Core memory was the speed bottleneck in the design of the 7600 and represented 80 percent of the component cost.

Cray corrected this deviation from the right path in the CRAY-1, which used chips for memory that other manufacturers had used only for high-speed register files. The CRAY-1 introduced "vector registers" with corresponding pipelined parallel vector functional units, returned to a uniformly fast

Continued on page 50

Continued from page 49

/4 cycle memory, and extended the exponent range to 10 to the 4960 while adding a post-operation floating point error detection and recovery.

Simplified Instruction Set

In the early 1980's the concept of reduced instruction set computers (RISC) as opposed to the then-current complex (CISC) machines began to be promoted within academia. The first VLSI chip using that concept was called the RISC I in a journal article by a group at the University of California, Berkeley. In that article,[1] the authors wrote:

Since architects of new machines build on the work of others, we believe it is important to trace the genealogy of RISC I.... The leading proponent of reduced instruction set computers for floating-point data is [Seymour] *Cray. For the last 15 years, he has combined simple instruction sets with sophisticated pipelined implementations to create the most powerful floating-point engines in the world. While Cray concentrates on impressive floating-point rates at impressive costs, RISC I concentrates on improved performance at lower cost for integer programs written in HLLs* [Higher Level Languages].

Not all agree that RISC is the wave of the future,[2] but some observers propose "the following elements as a working definition of a RISC":[3]

(1) "*Single-cycle operation* facilitates the rapid execution of simple functions that dominate a computer's instruction stream and promotes a low interpretive overhead."

(2) "*Load/Store design* follows from a desire for single-cycle operation."

(3) "*Hardwired control* provides for the fastest possible single-cycle operation. Microcode leads to slower control paths and adds to interpretive overhead."

(4) "*Relatively few instructions and addressing modes* facilitate a fast, simple interpretation by the control engine."

(5) "*Fixed instruction format* with consistent use eases the hard-wired decoding of instructions, which again speeds control paths."

(6) "*More compile-time effort* offers an opportunity to explicitly move static run-time complexity into the compiler."

From that proposed definition of a RISC, it does seem that Cray's machines from the 6600 onward could be considered RISC types. That same reference shows that the RISC I register file, organized as a series of overlapping registers that enabled fast procedure calls, was a large part of the performance improvement seen on that system. Perhaps the CRAY-2 should have organized its local memory to have similar procedure call speedups. At ACSS we have always considered the CDC 6600 and CYBER 74 as the world's largest and fastest micro engines.

Floating Point Arithmetic: Rounding

In the CDC 6600 and 7600, there were hidden 96-bit registers used in computing the full product, addition, or subtraction results of two 48-bit mantissas of floating point values. The instruction set had separate commands to obtain the upper or lower 48 bits of these results. Then double precision arithmetic could be done, taking approximately four times longer than single precision with short sequences of these upper and lower precision commands. The divide instruction, a harder process, did not have a corresponding lower command to produce the division remainder, and thus a fairly long sequence of commands was needed to produce the answer for a double precision divide.

Seymour Cray was always looking for the fastest execution rates, and normal arithmetic rounding requires that the round bit be added after any normalization in an additive or multiplicative operation to correctly produce the rounded result. Since this can add several cycles to a command, the 6600 and 7600 were designed to have the round bits added before the operation took place in the additive (1/2 bit) and divide (1/3 bit) operations. Since a post-normalization down-shift of 1 bit can occur, the rounding for those cases is sometimes only by 1/4 bit or 1/6 bit respectively. CDC initially delivered 6600s with the rounded multiply adding a 1/2 bit before the post-normalization, which caused a bit to be added in the lowest position for product results that shifted up the one-bit maximum. This meant that integer products such as 1*1 received an answer 1/2**47 too large. All of the 6600s were corrected to round only by a 1/4 bit before the post-normalization, giving a resulting round of 1/2 or 1/4. Numerical analysts have included these Cray arithmetic rounding anomalies in programming puzzles and articles on how not to design rounded arithmetic.

In our FORTRAN compilers (MNF and M77), running a program with rounded arithmetic usually produces more accurate answers and, combined with a run with unrounded arithmetic, quickly shows if loss of significant digits is occurring at a cost penalty of only two (i.e., often cheaper than converting to DOUBLE PRECISION or INTERVAL arithmetic).

Next month I will continue to examine Seymour Cray's machines, discussing some of the other innovative ways Cray solved floating point arithmetic problems and also examining some of the interesting features of his machine designs.

Notes

[1] David A. Patterson and Carlo Sequin, "A VLSI RISC," *IEEE Computer*, Vol. 15, No. 9 (September 1982).

[2] William C. Hopkins, "HLLDA defines RISC: Thoughts on RISCs, CISCs, and HLLDAs," *ACM SIGMICRO Newsletter*, Vol. 14, No. 4 (December 1983).

[3] Robert P. Colwell, Charles Y. Hitchcock III, E. Douglas Jensen, H. M. Brinkley Sprunt, and Charles P. Kollar, "Computers, Complexity, and Controversy," *IEEE Computer*, Vol. 18, No. 9 (September 1985).

Programming Languages on the VAX 8600: An Overview

Jim Miner and Janet Eberhart

With the advent of VAX 8600 service, we are introducing some new language processors and upgrading the existing ones as well. FORTRAN and Pascal are the perennial favorite languages on our systems, and we have a new version of each. There is also a new version of COBOL and new Ada, Common LISP, and OPS5 processors. Each of these processors is described below.

In addition, we are installing several other new programming and data management tools that augment the facilities provided by the language processors. New tools include DATATRIEVE (a query and report system), the Common Data Dictionary (CDD), Code Management System (CMS), Module Management System (MMS), and the Language Sensitive Editor (LSE).

Because of the careful design of the VAX/VMS system, the various programming languages share a Common Run-time Library and most can access the VAX Record Management System (RMS) and the complete range of VMS system services. This all adds up to a versatile and powerful environment for developing and maintaining sophisticated applications.

Ada®

Ada is a powerful, general-purpose language with integrated facilities supporting many modern programming practices. Ada Version 1.0 will be available for educational use only.

The U.S. Department of Defense sponsored the design of Ada. Among the requirements the DOD established for the language were features that would reduce software costs by making software more easily maintainable and evolvable and increasing reliability and portability. Ada provides language features for multi-tasking such as tasks, rendezvous, priorities, and entry calls. It also allows modularization and separate compilation. Other features of Ada include strong typing, data abstraction, concurrent processing, generic definitions, and exception handling.

VAX Ada is a validated implementation of the ANSI- and DOD-standard language. It is integrated into the VMS Common Language Environment. All VMS system services and utilities are thus available to programs written in VAX Ada. VAX Ada supports VAX Record Management Services (RMS). VAX Ada programs can invoke modules written in other VMS languages, and programs in other languages can invoke VAX Ada modules.

COBOL

VAX COBOL will be upgraded from Version 3.0 to Version 3.2. The versions are upwardly compatible.

VAX COBOL is based on the 1974 ANSI COBOL standard X3.23-1974 and includes many of the features planned for the next COBOL standard. It is designed to meet the low-to-intermediate level requirements of the U.S. government. Additionally, some modules are designed to meet Level 2 requirements: Nucleus,

Continued on page 52

Continued from page 51

Table Handling, Sequential I/O, Relative I/O, Segmentation, Library, Interprogram Communication, SORT/MERGE.

VAX COBOL is supported by both the VAX Common Run-Time Library and the VAX Symbolic Debugger. VAX COBOL supports access to VAX DBMS data bases and also to common record definitions stored in the VAX Common Data Dictionary. Object modules produced by VAX COBOL can be linked with object modules produced by other languages, including FORTRAN and Pascal. All VMS system services and utilities are available to programs written in VAX COBOL. VAX COBOL programs can invoke modules written in other VMS languages, and programs in other languages can invoke VAX COBOL modules.

FORTRAN

VAX FORTRAN will be upgraded from Version 3.5 to Version 4.3.

VAX FORTRAN is an implementation of the full FORTRAN-77 language. It includes optional support for programs conforming to the previous standard (ANSI X3.9-1966).

VAX FORTRAN provides many extensions to FORTRAN-77, including composite data declarations, additional data types, explicit specification of storage allocation, and language elements for keyed and sequential access to VAX RMS multikey ISAM files. VAX FORTRAN can also make use of VAX Common Data Dictionary information, VAX Symbolic Debugger, and it has a multi-phase optimizer.

VAX FORTRAN is integrated into the VMS common language environment. All VMS system services and utilities are thus available to programs written in VAX FORTRAN. VAX FORTRAN supports VAX Record Management Services (RMS). VAX FORTRAN programs can invoke modules written in other VMS languages, and programs in other languages can invoke VAX FORTRAN modules.

With the exceptions noted below, the old and new versions provide upward compatibility at the source, object, and image levels.

The new optimatizations implemented for Version 4.0 introduce the small chance that the run-time behavior of some FORTRAN programs may change.

Boolean Evaluation Order. The order of evaluation of Boolean expressions might have changed in some cases in the new version. The order of evaluation of logical expressions in IF statements may change from release to release.

Subtle Bounds Errors. Programs that address beyond arrays or string bounds are incorrect, and if not carefully written may work differently in the new version. Such programs may appear to work properly under the old version, but may fail at run time under the new version. The /CHECK=BOUNDS option can be used to find such out-of-bounds references.

DMOD More Accurate. The accuracy of the DMOD function is improved. In particular, in some cases where 0.0D0 was returned in the old version, a number very close to the first argument is returned in the new version.

Use of VAX/VMS Real-Time Features. Variables and arrays that are used to communicate with the operating system, such as Asynchronous Systems Trap (AST) service routines, should be declared VOLATILE. Otherwise an optimization might cause an old value, held in a register, to be used rather than the current value of the variable or array.

*Unformatted WRITEs of NTEGER*2 Parameter Constants.* In the old version, if an INTEGER*2 constant declared in a PARAMETER statement was used in an unformatted WRITE statements in a routine compiled with the /I4 qualifier, a long word (4 bytes) would be written. This problem has been corrected.

LISP

VAX Common LISP, Version 1.0, will be available.

LISP has been used extensively in artificial intelligence programming. It is characterized by symbolic expressions, simple syntax, representation of data by multi-level lists, and the representation of LISP programs as LISP data.

VAX LISP is an extended implementation of the COMMON LISP language defined in *COMMON LISP: The Language* by Guy Steele, Jr. (1984). COMMON LISP is an attempt to define a portable, powerful, efficient, and expressive dialect of LISP that is reasonably compatible with several older implementations, specifically ZETALISP, MACLISP, and INTERLISP.

VAX LISP supports many extensions to COMMON LISP including:

- Access to DCL (DIGITAL Control Language).
- Debugging facilities.
- Ability to call routines written in other languages.
- An error handler.
- An extendable editor.

In addition, some of the functions, macros, and facilities defined by COMMON LISP are modified for

the VAX LISP implementation. VAX LISP does not support complex numbers.

OPS5

OPS5 will also be available on the 8600.

OPS5 is the most widely used general-purpose expert-systems language for production-system programming. (Production-system programming is designed to simplify rule-based programming, and is one of several techniques for solving difficult industrial problems.)

VAX OPS5 is highly efficient and has an easy-to-program interface to all other languages available on VMS.

Pascal

VAX Pascal will be upgraded from Version 2.5 to Version 3.2. The versions are upwardly compatible.

Pascal is a small, general-purpose language that supports systematic programming with structured statements and data types. It is simple and easy to learn, yet powerful, supporting user-defined data types, recursion, and strong type checking. Initially designed for teaching, Pascal has found widespread use for implementing portable software.

VAX Pascal supports both the ISO (international) and the ANSI/IEEE (American) Pascal standards. It provides numerous extensions including variable-length character strings, access to VAX RMS relative and multikey indexed files, a module facility for separate compilation, and a value initialization section. VAX Pascal is integrated into the VAX Common Language Environment, giving it support for interlanguage calls, access to all VMS system services and the VAX Common Run-time Library, and use of the VAX Symbolic Debugger, Language Sensitive Editor, and Common Data Dictionary.

® Ada is a registered trademark of the United States Government (Ada Joint Program Office).

Choosing a FORTRAN Compiler for your Job

Janet Eberhart

ACSS has made five FORTRAN compilers available to our users. They are:

- University of Minnesota's M77 and MNF
- Control Data's FTN5 and FTN
- Digital Equipment's FORTRAN

In choosing one of these compilers, you should consider execution speed, compilation speed, compatibility with existing routines, and/or portability. The cost and effort of transferring data between machines may also be important considerations. And familiarity or ease of access may be of prime importance if you intend to use a program only once or are selecting a compiler for classroom projects.

M77 is well suited for student use and debugging. It provides excellent error messages and compiles very quickly, so errors are less costly. FTN5 may be the best choice for production runs on the CYBERs. Most programs will execute more quickly under FTN5(OPT=2) than under M77. Since M77 compiles more quickly than FTN5, it may be to your advantage to write and debug production programs using M77, than recompile under FTN5 to generate efficient object code for production runs.

There are several reasons to use the 1977 standard. First, FORTRAN-77 increases the portability of FORTRAN programs. (Note that the VAX no longer maintains a 66-standard compiler.) Second, writing programs that conform to the 1977 standard will avoid a conversion effort in the near future. Third, FORTRAN-77's control structures, such as the block-IF statement, make programs more understandable and easier to modify. If possible, try to stick to standard features when writing new programs.

Benchmarks

As shown in the accompanying graphs, we ran several benchmarks on the FORTRAN compilers we currently have available at ACSS.

Figures 1 and 2 show the results of these tests, as run on the CYBER CA, the MERITSS ME and MD, and the VX (the new VAX 8600). Keep in mind that the choice of a

Continued on page 54

Continued from page 53

benchmark test is always subjective; a different set of tests might well have generated significantly different results. The benchmarks we chose for this comparison are heavily slanted toward numerical analysis; they do little I/O, and, in order to run the same programs on all compilers, they do not use any of the 1977 standard features.

Figure 1 compares the sum of the compilation times in CP seconds. Figure 2 compares average execution times in CP seconds. These give an approximation of the cost involved in using the compilers. Since the billing structures are different on the different machines, the CP seconds only estimate the actual differences. For most jobs, the actual job cost is a weighted sum of CP seconds, mass storage used, file transfers, and off-line costs. Other hidden costs include file storage and data transfer between machines. Costs of running on a VAX will vary depending on how heavily loaded the system is and how much workspace is acquired.

CYBER FORTRANs

M77: M77 was written at the University of Minnesota. It is based on MNF, the University-produced 1966 FORTRAN compiler. M77 is actively maintained at ACSS. Three versions of M77 are available on our CYBERs: past (2.6), current (2.7), and future (2.7). All three use the Michigan State Record Manager to process files, so binaries can be intermixed. Version 2.7 uses dynamic file opening. (See the January *Newsletter.*)

We normally recommend the use of the current version; if, however, you suspect a compiler error, try the future version, because the future version is upgraded as new bugs are discovered and fixed. M77TS, the interactive subsystem, uses the same compiler as the batch version. M77 conforms to the 1977 ANSI FORTRAN standard and contains extensions to maintain compatibility with MNF. For a detailed look at the differences between M77, MNF, and FTN5, see the on-line document WRITEUP(M77) on the CYBERs.

Figure 1: Average Compilation Times

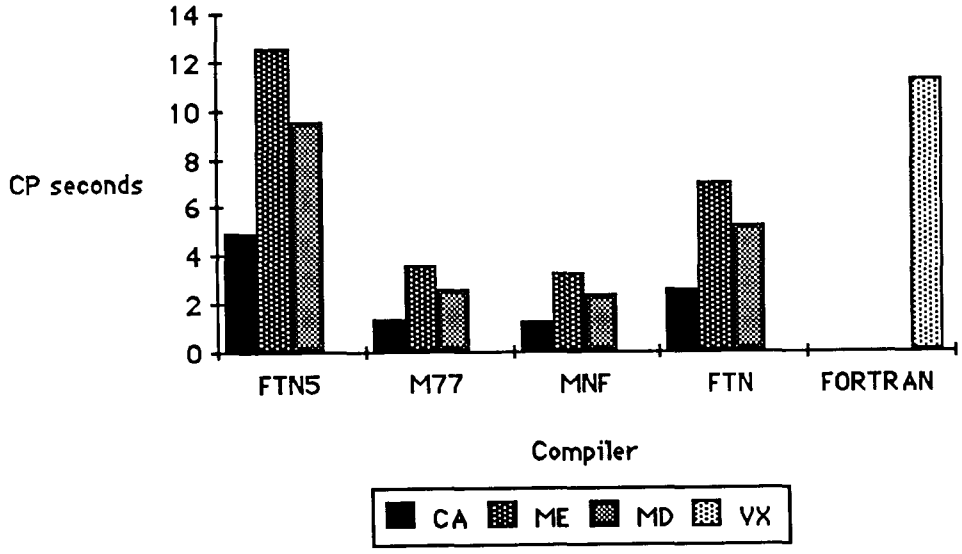
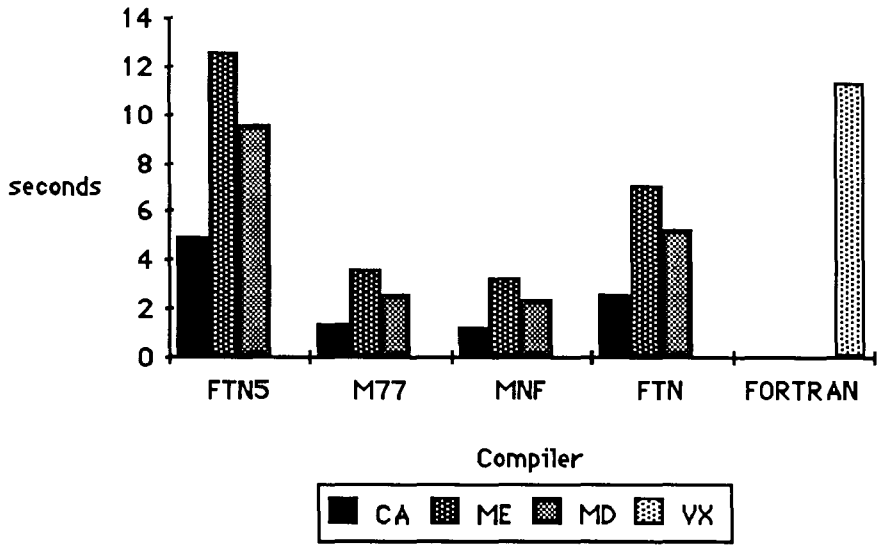


Figure 2: Average Execution Times



MNF: MNF was written at the University of Minnesota. It conforms to the 1966 FORTRAN standard. Currently four versions are available at ACSS: fetch, past, current, and future. The fetch version uses its own I/O routines, so it is not compatible with any other version. The past and current MNFs both use the Michigan State Record Record Manager (MSUIO). The future version uses the Cyber Record Manager (CRM). The differences between the past and current MNF result from bug fixes, so current MNF should be used. We are retaining MNF to provide users an easier transition path and for the convenience of its many users. New programs should generally **not** be written in MNF.

FTN5: CDC actively maintains FTN5, Control Data FORTRAN Version 5. It conforms to the 1977 standard and uses the standard Cyber Record Manager (CRM). FTN5 has several extensions, including optimization levels. These different optimization levels perform almost as different compilers. OPT=0 must be used for debugging runs, but it runs very slowly and it is usually better to use M77 for debugging. OPT=2 produces very good production code.

FTN: FTN, Control Data FORTRAN Extended Version 4, is no longer maintained by Control Data. Its use is generally not recommended. Control Data provides a conversion aid, F45, to assist with the conversion of FTN programs to FTN5. FTN conforms to the 1966 FORTRAN standard and uses the Cyber Record Manager.

VAX FORTRAN

Digital Equipment fully supports VAX FORTRAN, Version 4.3. See the article elsewhere in this issue on VAX languages for more information.

AI Notes

Tom Rindflesch and Ron Zacharski

If you would like to find out more about artificial intelligence (AI), you will be interested in an organization called TCSIGART and a symposium it is co-sponsoring.

TCSIGART

TCSIGART (Twin Cities Special Interest Group for Artificial Intelligence) holds monthly meetings, usually on campus, which feature lectures by people in academia or industry on a topic in AI. The lectures describe current research in AI or commercial products being developed. For information on the May meeting contact Tom Rindflesch or Ron Zacharski at the address or number below.

AI Symposium

IEEE and TCSIGART are sponsoring a day-long symposium on AI to be held on May 10 at the University Radisson Hotel in Minneapolis. The topic of the symposium is "Natural Languages and Man-Machine Interfaces." Professor Bonnie Weber of the University of Pennsylvania is the featured speaker. Registration fees, which include lunch and materials, are $60 for IEEE or TCSIGART members, $40 for students, and $80 for others. For more information and registration, call Dr. Alicja Ellis (541-2063) or Michael Kramer (631-3175).

Golden Common LISP: $130

Golden Common LISP runs on IBM-PC compatible machines with a minimum of 512K memory. This implementation of LISP has received very good reviews (see *BYTE*, December 1985, pp. 317-21) and is used by a number of universities and research laboratories. The package includes the interpreter, an on-line help system, and a GMACS editor. If we can find University purchasers for 12 copies, the price would be $100 per copy, plus $30 for each set of documentation. The usual university price is $350. If you would like to know more about GC LISP or are interested in purchasing a copy, please feel free to call us.

Clearinghouse for AI Software Information

If you have had experience with any AI software, we are interested in knowing your opinion of the product. In this way we can serve as a clearinghouse of information for people who are interested in purchasing software. We are interested in the expert-system shells people are using and the experiences people have had with various implementations of LISP and PROLOG (and FP and HOPE). So far we have received positive comments on Texas Instruments' version of SCHEME (a dialect of LISP) and negative comments about Expertelligence's version of LISP for the Macintosh.

Any information you provide may be of service to others. Please feel free to call or write us.

Tom Rindflesch
Ron Zacharski
124 Shepherd Laboratories
376-2944

CYBER Full Screen Editing

Steven Brehe

FOn SE, Control Data's Full Screen Editor, has recently been installed on the CYBER CA, ME, and MD. To learn the basics of using FSE, see the short on-line document WRITEUP, FSE on all CYBERs or our free printed document, the *FSE Brief*, available in the ACSS Reference Room, 128A Lind Hall.

The May issue of this *Newsletter* will also include an article on using FSE.

On the CA, some FSE commands work but produce error messages anyway. This will end when the CA is on CDCNET.

For a complete description of FSE, refer to the Control Data user's guide, *NOS Full Screen Editor*, available at the Electronics Desk in the Minnesota Book Center, Williamson Hall. You may also examine a copy in the ACSS Reference Room.

The HELP-Line: Past and Present

Tom Kovarik and Jill McAllister

All users of ACSS systems–faculty, staff, student, or corporate users–may call the ACSS HELP-Line at (612) 376-5592 any work day between the hours of 8:00 a.m. and 5:00 p.m. and ask any question, however simple or complex, in reference to any software or any hardware in use at ACSS. If the HELP-Line consultant on duty cannot answer the question, solve the problem, or give suitable advice immediately, the consultant will call the user back with the required solution as soon as possible, usually within a day.

Beginnings

The HELP-Line began eleven years ago, on February 10, 1975, when former staff member Richard Franta instituted the telephone consulting service in much the same form as we know it today. Rich even created the mnemonic for our HELP-Line phone number, 6-5592, which, when dialed, spells out **MKJWB**, an abbreviation for *My Kronos Job Went Bananas.* (At that time, the CYBERs were running the KRONOS operating system.)

Range of Questions

Since then, the HELP-Line has been open every work day and serves thousands of users each year. It is impossible to quote "typical" questions, because we receive such a wide range, from the simplest (such as the telephone numbers for dialing into the CYBERs interactively) to difficult, complex questions (such as techniques for efficient file structuring and retrieval in COBOL or subtle and unusual bugs in FORTRAN or COMPASS programs).

Because of this, our callers are also not easily typified. For example, a user may call who is attempting his or her first logging onto the VAX system and who has no mainframe experience. This call may be followed by a call from a user who has fifteen years of fulltime programming experience and who is dealing with a delicate issue in linkage of libraries and subroutines. Other FORTRAN users call us with problems with arrays, overlays, using external files, and files not being rewound. Pascal programmers call especially for problems with I/O (READ vs. READLN) and on how to do exponentiation.

Tape problems also get a good representation. There are questions on utilities such as EXAMINE, TBLOCK and UNBLOCK, and COPYCH, plus questions on character set conversion and what error messages mean. Likewise, people call us when they've lost files, when they need to know how to kill a job, and when they need permission changes. Users also ask about our graphics packages and statistical packages frequently.

Perhaps it is because of this variety of questions from our diverse community of users that the HELP-Line staff finds consulting so challenging. The popularity and success of the HELP-Line is best measured by its volume of usage and the recurrance of calls from users who have come to rely upon and trust the timeliness and accuracy of problem resolution from our consulting staff.

Consider these statistics: During the month of January 1986, an average of 86 people called the HELP-Line each work day, and a caller could expect a wait of 73 seconds (while listening to music) before speaking to a consultant. Because HELP-Line staff have access to a complete library of manuals and microfiche for all ACSS systems and a terminal logged into all ACSS systems simultaneously as the master user number over all other user numbers, most of these calls were satisfactorily resolved immediately, whereas on the average 8 calls per day required a call back to the user for an unusual or very difficult problem.

Special HELP-Lines

ACSS also sponsors specialized HELP-Lines, such as the Data Base HELP-Line, the Statistical Packages HELP-Line, the Micro HELP-Line, and the HELP-Line for text processing and analysis and artificial intelligence. These HELP-Lines are staffed by specialists in these fields, at reduced hours in proportion to their usage. Their telephone numbers and hours are listed inside the back cover of this *Newsletter.*

We have contact with many computer users and vendors in government and industry who are impressed with the concept and

Continued on page 58

Continued from page 57

successful implementation of a generalized HELP-Line that attempts to answer all questions from all users on any ACSS system. We see this facility as the principal way ACSS reaches out to all our users during their more critical and sometimes frustrating computing endeavors. We urge you to use this resource: Call us with your next problem, or just call in and tell us who you are and what you plan to do on our systems.

Using the Lind Hall Laser Printer

Parker Johnson

ACSS has recently installed a new laser printer at the 128B Lind Hall Input/Output station, and we are removing the Lind Hall impact printers. The laser printer is now the default printer for all jobs. To get files printed at Lind, use the **ROUTE** or **PRINT** commands. For details about the **PRINT** command and laser printer capabilities, see the article on **PRINT** in the March *Newsletter* or WRITEUP, PRINT.

Files printed at Lind Hall will be printed by default in landscape mode with 66 lines per page. You may also get 88 lines per page, portrait, or reduced output printed at Lind Hall by means of the **PRINT** command. You may not, however, specify duplex, three-hole punch, or shifted output for files printed at Lind. If you want these printing options, you should designate Lauderdale as your printing site (use **UN=BC** as a parameter in your **PRINT** command), and have your output delivered to Lind (using the **UJN** parameter). The turnaround time is about an hour.

Compared with the Xerox 8700, the Lind Hall printer offers few special features. The Lind Hall printer can only do underlining, while the Lauderdale printer can do other kinds of special effects (such as boldface, italics, or overprinting). Users requiring these features should use the **PRINT** command to send their files to the Lauderdale printer (**UN=BC**). The Lind Hall printer will print 136 characters per line in landscape mode. You should assume that all lines longer than 136 characters will be truncated.

Examples

To send a file to the Lind printer, you may use either the **PRINT** or **ROUTE** commands with essentially the same parameters. For example, the following command will print the file ***myfile*** in standard landscape mode to be delivered to bin 450 in 128B Lind Hall.

```
print,myfile,un=ea,ujn=ea*450
```

If you add parameters such as **SH** or **D** to this command, your file will be diverted to the printer at Lauderdale where shifted or duplex printing is taken care of, and the output will be returned to bin 450 in Lind. The following **ROUTE** command will also get your file printed at Lind Hall:

```
route,myfile,dc=pr,un=ea,ujn=ea*450
```

For further information and examples, especially on using the PRINT and ROUTE commands and the Lauderdale laser printer, see WRITEUP, PRINT.

BMDP 83

Bruce Center

At the end of Spring Quarter, we will make BMDP 83 the current version of BMDP on the CYBERs and BMDP 82 the past version. BMDP 81 will no longer be available.

You can get more information on BMDP 83 by referring to the on-line CYBER document **WRITEUP,BMDP.**

New Features in BMDP 83

Many new transformation functions have been added. These include a crude RECODE function, date and time functions, ability to replace missing values by linear interpolation, and numerous summary statistics within a case, including sum, mean, median, standard deviation, interquartile range, linear interpolation, trend, t-value, and the correlation coefficient. These transformations and others are documented on p. 52 of the *BMDP Users Manual.*

BMDP Save Files may optionally be written as a formatted file (rather than a binary file). A formatted file can be visually inspected and is easier to transfer to other non-BMDP programs. See p. 69 of the *BMDP Manual.*

Many bug fixes have been supplied, especially for BMDP1T and BMDP2T.

Changes and Incompatibilities

The program known as BMDQ3M (Block Clustering) in BMDP 82 has been replaced by BMDP3M in the 1983 release. The old program BMDP3M is no longer available. Users may still access it by typing **PAST,BMDP3M.**

The command parameters `D`, `S`, `AI`, `MF`, and `AF` are no longer permitted in BMDP 83. Attempting to use them will result in a CPU Abort with the dayfile message:

```
ILLEGAL PARAMETER
```

Instead, these file names are communicated to the BMDP programs by means of control language directives described below. This usage is consistent with the *BMDP 83 Manual.*

The broadly applicable parameters:

`I`	[INPUT]	Input file
`L`	[OUTPUT]	Output file
`B`	[none]	User-supplied relocatable binary
`W`	[10,000]	Decimal workspace available

are still permitted. Default values are in brackets.

Continued on page 60

Continued from page 59

File Use in BMDP 83

Local file names are specified directly within the BMDP control language. The standard input file is requested by specifying:

```
/INPUT UNIT = 5.
```

All other files must be referred to by a file name in the **FILE = *lfn*.** sentence. The **UNIT = *n*.** sentence, used in previous releases, is not permitted in BMDP 83, except **UNIT = 5.** to refer to the standard input file. Note that this usage differs from that described in the *BMDP 83 Users Manual.*

This new way of communicating file names allows greater flexibility in file use, since more than one file of any given type can be used per run. Consider the following BMDP job consisting of two problems, each of which reads data from its own raw data file:

```
GET,DATA1.
GET,DATA2.
BMDP3D.
.EOR
/PROBLEM
/INPUT VARIABLES=5.
FILE=DATA1.
.
.
.
/END
/PROBLEM
/INPUT VARIABLES=3.
FILE=DATA2.
.
.
.
/END
```

In this job, the **FILE=DATA1.** sentence in the first problem specifies that raw data is to be read from the file DATA1. Similarly, the **FILE=DATA2.** sentence in the second problem indicates that data is to be read from DATA2.

See WRITEUP,BMDP for further conditions and examples of file usage in BMDP 83.

Library Changes and Additions

Mike Frisch

On Thursday, February 13, we corrected the problem with MATLAB that prevented it from running when it was called for the first time. (People had worked around the problem by calling MATLAB twice in a row.)

Microcomputer Security Devices

Dan Whealdon

ACSS's Engineering Services group is now providing security devices for microcomputers and their associated peripherals. The standard installation includes five equipment plates, two cables, and a lock. This configuration is intended to secure the system unit, monitor, keyboard, and printer. (The fifth plate attaches to a desk, table, or some other suitable surface to act as the anchor point.) A large cable connects the units to the anchor point. A smaller, more flexible cable provides security for the keyboard while allowing a reasonable degree of movement.

The standard security device configuration is $75.00, installed. Additional parts may be purchased according to the following schedule:

Security plates	$ 5.00 each
Standard (10') large cable	$10.00 each
Standard (10') small cable	$ 5.00 each
Custom length large cable	$ 1.25 per foot
Custom length small cable	$ 1.00 per foot

All prices include installation.

If you wish to have a security device installed on your system, call Engineering Services at 376-1313, indicating the type of equipment you want to secure. University departments should also provide a budget number for the installation.

As with many other theft protection strategies, these security devices deter theft, but cannot provide complete security. ACSS assumes no responsibility for the theft of microcomputer equipment equipped with one of these security devices.

Expanded Microcomputer HELP-Line Hours

The Microcomputer HELP-Line hours have been extended. Effective *now* you can call (376-4276) or visit (125 Shepherd Labs) during the following hours:

Wednesday & Thursday	9:00 a.m.-4:00 p.m.
Monday, Tuesday, Friday	9:00 a.m.-noon and 1:30-4:00 p.m.

These extended hours should reduce the waiting time when you call or stop by.

DI-3000 Available on MD

Michele Lewis

The Precision Visuals Inc. graphics packages, DI-3000 and the Metafile Translator, are now available on the MD machine. These packages are also available on the VAX 780 and CA machines and will be on the new VAX 8600. DI-3000 is a library of FORTRAN-callable routines in which two- and three-dimensional graphics are supported in both batch and interactive environments. The package is device-independent and contains several advanced features such as retained segments, segment priority control, and full 3D modeling. The Metafile Translator allows an application program to create metafiles, store them, and then post-process them on a device.

Documentation for the packages is available through a number of sources. The *DI-3000 User's Guide,* provided by the vendor, is available at the Minnesota Book Center in Williamson Hall on the East Bank and at the H. D. Smith Bookstore on the West Bank. This manual describes both the underlying principles and functional capabilities of DI-3000. The *DI-3000 Quick Reference Guide* and the *Metafile System User's Guide* are also available. Copies of the DI-3000 documentation are located in the ACSS Reference Room in 128A Lind Hall. In addition, WRITEUP, DI-3000 on the CA and MD and MOREHELP WRITEUP GRAPHICS on the VAX 780 provide the system-specific information needed to run an application program on the MD machine and the other ACSS machines. For help in using DI-3000, call the HELP-Line, 376-5592.

Spring Quarter Short Courses

1986

INTRODUCTORY COURSES

Introduction to Computers	McAllister	April 7-18	(MWF)	2:15-4 pm	$15,$25,$35
Beginning NOS 2 (Cyber OS)	Krmpotich	April 21-May 2	(MWF)	3:15-5 pm	$15,$25,$35
Introduction to VAX/VMS	Stearns	April 22-May 8	(TTh)	2:15-4 pm	$15,$25,$35
Graphics Packages at ACSS	McAllister	May 5-14	(MW)	2:15-4 pm	$15,$25,$35

ELECTIVE COURSES

SPSS-X (Statistics Package)	Alberg	May 19-23	(MWThF)	2:15-4 pm	$25,$35,$60
Magnetic Tapes in NOS 2	Oberg	May 20-29	(TTh)	2:15-4 pm	$25,$35,$60
Beginning FORTRAN Programming	Kovarik	May 23-June 6	(MWF)	2:15-4 pm	$25,$35,$60
SIR DBMS Seminar	Oberg	June 11-13	(WThF)	9 am-4 pm	$100,$100,$150

If you need more information on short courses, call Jerry Stearns at 376-8806.

PHONE NUMBERS

Access:
- CYBER(CA)—10, 30 cps 376-5730
- —120 cps 376-5706
- MERITSS(ME)—10, 30 cps 376-7730
- —120 cps..... 376-7120
- VAX/VMS(VA)—(autobaud) 376-8070

Accounts:
- MERITSS 373-7745
- User Names 373-4548

Computer-Aided Instruction 376-2975
Computer Hours (recorded message) 373-4927
Consulting:
- HELP-Line 376-5592
  8 a.m.–5 p.m., Monday-Friday
- Statistics Packages 376-1761
  1–2 p.m., Monday-Friday
- Data Bases 376-1761
  10–11 a.m., Monday-Friday
- Microcomputers 376-4276
  9:30 a.m.–noon and 1:30–4 p.m.,
  Monday, Tuesday, Friday
  9:30 a.m.–4 p.m., Wednesay, Thursday
- Text Processing & Analysis,
  Artificial Intelligence 376-2944
  3-4 p.m., Monday-Friday

Contract Programming 376-1764
Data Base Applications 376-1764
Engineering Services 376-1023, 376-8153
Equipment Purchase/Information 376-8153
Lind Hall I/O 373-4596
Graphics Software 376-5592
HELP-Line 376-5592
8 a.m.–5 p.m., Monday-Friday
HOURS-line (recorded message) 373-4927
Information, Wulling Hall 373-4360
Information, Lauderdale 373-4912
Instructional Labs 376-2703
Instructional Services 373-7745
Lauderdale Computer Room 373-4940
Lauderdale Services 638-0523
Newsletter Subscription 376-1491
Permanent File Restoration 376-5605
Professional Services Division 376-1764
Project Assistance 376-1764
Reference Room 373-7744
Remote Batch (RJE) Services 376-2703
Short Courses 376-8806
Shuttle Bus Service 376-3068
System Status (recorded message) 373-4927
Tape Librarian: see Lauderdale Services

OPERATING HOURS

	CYBER (CA)	Low Rate	MERITSS (ME)	MERITSS (MD)	VAX (VA)
M-F	7 a.m. - 4 a.m.	8 p.m - 4 a.m.	7:45 a.m. - 3:30 a.m.	8:00 a.m. - 1:30 a.m.	7 a.m. - 4 a.m.
Sat	4 a.m. - 5:15 p.m	4 a.m. - 5:15 p.m.	7:45 a.m. - 3:30 a.m.	8:00 a.m. - 1:30 a.m.	4 a.m. - 5:15 p.m.
Sun	4 p.m. - 1 a.m.	4 p.m. - 1 a.m.	4 p.m. - 3:30 a.m.	4:00 p.m. - midnight	4 p.m. - 1 a.m.

PUBLIC LABS – TWIN CITIES CAMPUS

Location	Batch	Interactive	Micro
East Bank			
Arch 160			X
CentH		X	
ComH		X	
DiehlH 207		X	
EltH 121, 124		X	
EltH N640	X		
FolH 14, 14a	X	X*	X
FronH		X	
LindH 26		X	
LindH 128B	X	*	
LindH 306B			X
MechE 308		X	
Physics 69		*	
PlH		X	
SafH		X	
Terr			
TerrH		X	
VincH 4		X	
Walib 9		X	
West Bank			
AndH 170			X
BlegH 25		*	
BlegH 90		X	
BlegH 140		X	
MdbH		X	
OMWL 2		X	
St. Paul			
BaH		X	
CentLib B50			X
ClaOff 125	X	X	

* Research cluster; access to CYBER CA and VAX/VMS
X in interactive column indicates access to MERITSS

For more information see WRITEUP(LABS)

Contents

***The* ACSS Newsletter**
April 1986
Volume 20, Number 4

Acting Director: *Michael M. Skow*
Editors: *Steven Brehe, Paula Goblirsch*

The *ACSS Newsletter* is published monthly by Academic Computing Services and Systems (formerly the University Computing Center) of the University of Minnesota, Twin Cities. Deadline for articles is the 10th of the month preceding publication; deadline for short announcements is the 15th. The *Newsletter* is produced with an Apple Macintosh running Microsoft Word, MacPaint, MacDraw, and Aldus Pagemaker software, with camera-ready copy produced on the Apple LaserWriter.

Direct comments, suggestions, articles, announcements, and subscriptions to the editors at the address below, or call (612) 376-1491. On-campus address changes *must* include your department's name and your *departmental* address.

***The* ACSS**
Newsletter

Academic Computing Services and Systems

Technical Publications
5 Wulling Hall
University of Minnesota
86 Pleasant Street SE
Minneapolis, Minnesota 55455