UO-LISP NEWSLETTER

| January 1984 | Vol. 1. No. 1 |

Premier Edition

The growing number of UO-LISP users has prompted Far West to publish this newsletter to acquaint users with new software being offered, provide tips on use of the system, and broadcast bugs and fixes. Each issue will provide news of upcoming events of interest to LISP programmers. A section will be devoted to answering questions from users and as a special feature, each issue will have a complete LISP program designed to illuminate to some aspect of the UO-LISP system.

We encourage you to submit programs, questions, articles, and news of coming events. Our close connection with Lisp means that we are not always in touch with needs of beginning users. Your problem may be one which has not occurred to us and an explanation of its solution will be of benefit to the entire community. Or perhaps you have a program or utility which is of use to everyone. You will be a much more productive LISP programmer if you can build on the tools provided by others and do not have to "reinvent the wheel" for each new program.

New Software

Many improvements have been made in UOLISP since the release of version 1.5 for CP/M and the TRS-80. The interpreter has been augmented, the number of support packages has doubled, and a large set of example programs in LISP and RLISP have been implemented. The following changes have occurred since version 1.5.

The Manual

Perhaps the biggest change has been to the UO-LISP User's Guide. The new manual is well over 300 pages long and is divided into 12 chapters. It includes new sections on the document formatter, big and fixed numbers, the revised structure editor, the utility packages, the macro packages, and the Little Meta Translator Writing system. In addition, nearly every function in the entire manual is now described with an example of its operation. Since the manual has gotten so large, we have also created a handbook which includes all of the functions in the system, a synopsis of commands for the editors, text processor,

and other systems, and a list of all packages in the system. To keep the manual size reasonable, we have also created a report series for the application and demonstration programs.

Version 1.14c Interpreter

1. COMPRESS, EXPLODE, and EXPLODE2 have been implemented. COMPRESS takes a list of single character identifiers and builds a lisp expression out of it. EXPLODE and EXPLODE2 create lists of characters from S-expressions.

2. The N2I function converts integers into single character identifiers. I2N converts single character identifiers into their corresponding numbers.

3. MACRO type functions have been implemented. MACRO functions permit implementing WHILE...DO..., FOR loops, data structuring in LISP without overhead (when compiled).

4. DIGIT and LITER return T when their arguments are single characters which are digits and letters of the alphabet.

5. Up to 4 disk files may be open at any time in any combination of input or output. Two other channels are connected to the CP/M print and read devices. The user can also install his own I/O drivers on any channel through the use of the INSTALL function.

6. The FLUID variable binding mechanism is implemented. FLUID variables are like GLOBAL variables, but can be used as function parameters or as PROG variables. They also permit variable name communication between interpreted and compiled code.

7. A backtrace on error mechanism is implemented. When an error occurs, the contents of the ALIST and frame stack are displayed.

8. The BPS!$ function displays the number of available identifiers, free code pointers, string space available, and the amount of remaining binary program space.

9. The loader and compiler check for overrunning the binary program space.

10. Packages which have been loaded by the fast loader can be unloaded and their binary program space and code pointers can be used for other packages.

11. A Coroutine mechanism has been implemented. This permits executing up to 5 processes concurrently.

*12. The disk I/O routines permit random access files to be used.

13. Many bugs have been fixed and many of the basic interpreter functions have been recoded for speed and size improvement.

14. A short call mechanism permits the compiler to generate two byte function calls for many functions with significant savings in the size of compiled code (10%).

15. The garbage collector removes unreferenced identifiers from the symbol table.

16. COND and LAMBDA now support an implied PROGN. The antecedent of a COND or the body of a LAMBDA can be a set of S-expressions without the necessity of adding a PROGN.

17. The PROG1 function has been implemented. It is like PROGN, but the value returned is the first of its statements rather than the last.

18. The support functions EQCAR (equivalent to (EQ (CAR ..) ..)), REVERSIP (like REVERSE but does so in place destroying the original list structure), LEQ, NEQ, and GEQ.

19. The COMMENT function has been implemented as a way of saving annotation in a LISP file.

20. The vector data type has been moved inside the interpreter.

21. Read macros and a read table are now supported.

The Compiler and Optimizer

1. The basic compiler optimizes calls to LIST that have 1, 2, and 3 arguments to the equivalent functions NCONS, LIST2, and LIST3.

2. Implied PROGN is supported in COND's and LAMBDA expressions.

3. The PROG1 function is open compiled.

4. Improvements have been made to the LAP pretty printer and LAP interface.

5. Lambda expresssions as functions are compiled. Thus: ((LAMBDA (X) (ADD1 X)) 12) is compiled correctly.

6. Improvements have been made in the code generated by the basic compiler.

7. The FLUID variable type is supported in compiled code.

8. The two byte call mechanism is supported by the compiler as an option.

9. The optimizer does argument reordering to take advantage of registers.

10. Functions are listed as they are compiled during the fast load file generation process.

11. The compiler and interpreter have been fixed to allow compilation of functions with more than 3 arguments (up to 63 are permissible).

RLISP

1. The terse printer is interfaced to RLISP.

2. Some improvements have resulted in a reduction in the size of the RLISP code.

3. Big numbers and fixed numbers are supported in source code.

The Trace Package

1. The basic package has been simplified by the removal of the BREAK function and the output improved. The Terse printer is interfaced to the output facility. The size of a traced function has been reduced.

2. An extended trace package implements tracing of FEXPR's as well as assigments to variables. A program BREAK facility permits selective tracing and environment examination at run time.

3. An execution profiling package has been implemented. This package lists the number of times a function gets called during the execution of a program.

The Document Formatting Program LISPTEX

The document formatting package implements LISP based word processing. The formatter does text justification, centering, page numbering, index and table of contents maintenance, table formatting, and switching between different fonts. This document was formatted and printed by LISPTEX.

The Structure Editor

The structure editor has been completely rewritten. A file package constructs and maintains multiple copies of the source file by attaching a version number to the file name. More than one file can be edited at any time. The structure editor contains many new commands including the ability to perform editing on more than one function or structure at a time. There is a limited ability to maintain comments in the source file. The editor also contains a character editor, a facility for editing the individual characters of an atom or structure.

New Packages

Several new support packages have been implemented and many old ones have been updated.

1. CP/M operating system support. This package includes a number of routines which perform calls on the CP/M disk and basic I/O routines. A second package implements more complex calls and permits interaction with the CP/M filing system for deletion and renaming of files as well as directory search.

2. Fast arithmetic, bit-logical operations, and random number generator. This package contains routines for doing fast addition, and subtraction as well as shifting and logical operations on integers. A pseudo-random number generator is implemented.

3. History saving read loop. This package maintains a list of the commands that have been previously entered. These may be examined, edited, and rexecuted.

4. Terse printer. This package complements the pretty printer and can be used with the structure editor and the trace package. It limits the amount of output from PRINT by displaying only the top few levels of a tree structure and the first few elements of an array or list.

5. Internal GLOBAL variables. This package implements named access to many of the internal global variables used in the system.

6. Macro packages. These packages contain a number of compiled macros for advanced control and data structures in LISP. This includes CASE, IF, FOR, REPEAT, and WHILE macros in the control package, structure definition and assignment functions in the data structures package, and some useful I/O macros in a third package. The fourth package implements the backquote facility for easy construction of macros.

7. Terminal drivers. This is a collection of routines to be loaded with programs that require CRT screen operations. Each terminal type has their own driver program. The source code for the Televideo and ADM22 terminal drivers are included.

8. Sort packages. These two packages implement a list insertion sort and a disk based merge sort so that large numbers of items can be sorted into a disk file.

9. The LSED Screen Editor. This program edits LISP source program files on a character rather than structure basis.

10. Z80 assembler package. This package is an extension to LAP to permit all the Z80 instructions to be used.

*11. File Transfer Program (FTP). This package permits you to communicate with other UOLISP installations and with Far West. The program includes an electronic mail facility, talk

facility, and the file transfer mechanism.

*12. Distributed LISP. This package implements synchronization and communication of process between two or more LISP systems.

*13. Franz LISP compatibility package. This package permits Franz LISP programs to be written and tested. These can then be moved to UNIX version of Franz LISP. Not all of Franz LISP is supported.

14. Auto loading. This package implements automatic loading of functions within the lisp system. A second package implements package swapping.

15. Low level debugging. This package is a byte and address level DDT which permits examination of compiled LISP functions and contents of buffers.

16. The Little Meta Translator Writing System is now available for CP/M systems.

*17. PROLOG. The programming language PROLOG implemented in LISP (contributed by Rabbe Fogelholm of the Royal Institute of Technology, Stockholm, Sweden). This package permits the user to create and debug simple PROLOG programs.

18. A package has been written to support the construction of read macros.

*19. Cross reference program. This program takes a UO-LISP source file and displays information about which functions are called from where, and what GLOBAL and FLUID variables are used.

20. The 'hunk' data structure is a fast byte vector. It is useful for storing single bytes in a fixed length vector and retrieving the value without a list search as in the UO-LISP vector structure. This will speed up many applications by an order of magnitude.

Educational Software

Far West is now offering two packages for those learning LISP. These make UO-LISP look like the LISP presented in various text books. Currently packages exist for 'LISP' by Winston and Horn, and the 'LISP 1.5 Primer' by Clark Weissman.

* item is to be released in the near future.

Demonstration Programs

A number of demonstration programs have been implemented. The

source code for these are distributed together with a document describing each one and examples of its use (where applicable).

1. The SNAKE game. A game for CRT's: the snake gobbles the random numbers that appear on the screen and gets longer. The operator controls movement with characters from the terminal. The game ends when the snake runs into itself or the wall.

2. Othello. A game program which demonstrates some uses of vectors. The program is not very smart though it has been known to win. The documentation describes the rules and the strategy used by the program.

*3. The NLARGE computer algebra system. This program accepts equations and performs operations on them. For example, (X+1)^2 is expanded into (X^2 + 2*X + 1). Bignums permit (X+1)^20 to be expanded (we don't know what the limit is). Polynomials can be added, subtracted divided, multiplied, and differentiated with respect to any variable. The package also includes some matrix manipulations to do computation of determinants, matrix inversion and multiplication. The program treats the algebraic operators as "objects" and is a simple example of "object oriented" programming in LISP. A demonstration program which runs about 10 minutes is included.

4. Fruit world. A simple intelligent system shows how the Little Meta Translator Writing System can be interfaced to a program that knows about fuit and how to make inferences based on this information. The input and output are English sentences.

5. Your Program. Far West is actively soliciting contributions from users. These will either be published in the newsletter or offered as part of the growing UOLISP program library.

UO-LISP Bugs & Complaints

We hope to keep this section small but we won't be foolish enough to deny that there aren't any such things around. The following have come to our attention:

Version 1.5a,b only (TRS-80 Model I, III). The Little META Translator Writing system has a problem running under version 1.5a (not version 1.5 or before). The use of the - sign in the test-x- construct causes problems and the sample distributed programs don't work. To solve this problem, edit these files and place at least one blank after every minus sign.

Version 1.13-1.14b (CP/M system). Characters with a code less than 32 are ignored by the input reader. Consequently assigning entries for them in the read table will not work. This has been fixed in subsequent versions.

Other News

FOLLK (Friends of LISP/Logo & Kids) is a recently formed non-profit Educational & Scientific Corporation based in San Francisco. It publishes a quarterly newsletter with articles describing LISP, Logo, and PROLOG. It also sponsors computer camps, a hot-line for answering questions about LISP, Logo and other AI languages, and monthly FOLLK-Meets. A subscription to the newsletter is $7.50 and a FOLLK regular membership is $25.00 and a student membership is $15.00. FOLLK can be reached at 254 Laguna Honda Boulevard, San Francisco, California 94116, (415)-753-6555. FOLLK is not connected with Far West in any way.

Coming Events

The 1984 ACM Composium on LISP and Functional Programming will be held at the University of Texas in Austin on August 5-8, 1984.

EUROSAM '84 International Symposium on Symbolic and Algebraic computation will be held in Cambridge, England on July 9-11, 1984.

Bibliography

In this section we will list recently published articles and books of interest to the LISP programmer. We would also like to print book and article reviews.

Marti, J., 'The Little Meta Translator Writing System', Software Practice and Experience, October 1983, pp. 78-xx.
Describes the Little Meta Translator Writing System by presenting a syntax checker, an interpreter and a compiler for a small programming language. Recommended reading for Little Meta users (reprints are available on a first come first serve basis from Far West).

Questions and Complaints from the Users

In this section we will present questions we have received from users both over the phone and by letter as well as the best answers we can give.

Can I reconfigure the data spaces for my own applications?
Answer: No. Z80 code is not very relocatable and not every CP/M or TRS-80 system has a relocating linker. We have spent considerable time tuning the data spaces so that most applications will run without reassembling the system. The 8086 version of UO-LISP will be statically reconfigurable. The Z80 system can be easily be reconfigured by Far West upon request (please call us to discuss your needs). For example we could create a version which permitted you to have 20 files open at one time, or a version with 32k binary program space (at the expense of stack and dotted-pair space), or most any other special request you might have, as long as it will fit into 64k.

I'm getting really sick of seeing the CAR and CDR of NIL error

message.
There are two solutions to this problem. The easiest is not to take CAR or CDR of an atom, usually this error is a symptom of some other error. Some LISP dialects (Interlisp and MacLisp for instance) do permit you to take CAR and CDR of NIL, but not other atoms. If this style of programming appeals to you, simply create a special CAR and CDR which check for this special condition (don't forget the composites too):

```
(DE CAR!* (x) (AND (PAIRP x) (CAR x)))
(DE CDR!* (x) (AND (PAIRP x) (CDR x)))
```

Programs

In this first issue we include the following interesting algorithm programmed by Julian Padget of the University of Bath in England. It computes PI to as many decimal places as one wishes to wait for using a method called "continued fractions". The following program can be run as is. The last lines computed PI to 10 and 20 decimal places respectively. We have used this routine to compute PI to 100 decimal places, but it takes about 30 minutes.

```
% Load the packages required.
(FLOAD "USEFUL")
(LOADF "MACROS" "RTABLE" "FIXED")

% !*PREC is precision to compute to, default=10.
(GLOBAL '(!*PREC))
(SETQ !*PREC 10)

% Define a read macro for big numbers.
(DRM !#
  (WHILE (DIGIT (SETQ C (R!$)))
         (WITH C L)
         (INITIALLY (READCH))
         (RETURNS (MKQUOTE L))
         (DO (SETQ L (CONS (I2NO C) L))
             (READCH))))
(DE I2NO (C) (DIFFERENCE (I2N C) 48))


% Compute PI for N iterations.
(DE PI (N)
  (FOR (WITH AN2 AN1 AN BN2 BN1 BN TMP1 TMP2)
       (INITIALLY (SETQ AN2 #0)
                  (SETQ AN1 #1)
                  (SETQ BN2 #1)
                  (SETQ BN1 #1))
       (FROM I 2 N)
       (DO
         (SETQ TMP1 (BEXPT (BIGNUM (SUB1 I)) 2))
         (SETQ TMP2 (BSUB1 (BTIMES2 #2 (BIGNUM I))))
         (SETQ AN (BPLUS2 (BTIMES2 TMP1 AN2)
                          (BTIMES2 TMP2 AN1)))
         (SETQ BN (BPLUS2 (BTIMES2 TMP1 BN2)
```

```
                        (BTIMES2 TMP2 BN1)))
      (SETQ AN2 AN1) (SETQ AN1 AN)
      (SETQ BN2 BN1) (SETQ BN1 BN) )
    (RETURNS (!$QUOTIENT
               (BTIMES2 #4 (CONS 0 AN))
               (CONS 0 BN))))))

% Gets about .75 digits/iteration.
(!$PRINT (PI 16))
(SETQ !*PREC 20)
(!$PRINT (PI 28))
```

This program uses both the FIXED and BIGNUM packages as well as the MACROS, RTABLE, and USEFUL packages. Note that FIXED automatically causes the BIGNUM package to be loaded if it is not already so. The read macro for defined by the call to DRM is used to all big numbers to be used in the source. Any number prefixed by an # will be converted into big number format. The # read macro uses must of the features of the WHILE macro which is defined in the MACROS package. The argument of the PI function is the number of iterations to perform. The algorithm computes slightly less than .75 digits per iteration.

Next Issue

The next issue will appear in April 1984 and features articles on the structure editor and a Little Meta translator for a subset of LOGO.