ELISP - EXTENDED ADDRESSING LISP

ELISP is an implementation of Lisp for the DECSYSTEM-20. It uses the full 24-bit extended address space. It can be reconfigured easily to use any address space of up to 30 bits. It requires TOPS-20 version 4 or later and a KL-10 model B processor. (Patches are included to turn on extended addressing in a version 4 monitor.)

There are three design goals:

1. To provide a version of Lisp that supports a very large address space.

2. To provide most of the data types and other facilities of modern Lisps, but not facilities that imply great complexity or runtime penalties.

3. To be upward compatible with UCI Lisp.

Here are some major features of its design:

- Typed pointers, with 6-bit type fields and 30-bit pointers
- Immediate 32-bit integers and reals (i.e. no CONS's are needed to create 32-bit numbers)
- Arbitrarily large integers, integrated into the system transparently
- A two-space copying garbage collector
- The ability to load .REL files by calling LINK in a subfork
- A memory-mapped interface to EMACS
- A single-stepper for interpreted functions
- A debugger, break package, and form editor adapted from Interlisp
- New data types, including hash tables, vectors, and records

ELISP involves a complete recoding of the assembly language part of Lisp. The user facilities and many of the functions are Lisp code that was simply moved from Rutgers/UCI Lisp without change. The compiler is a somewhat modified version of the Standard Lisp compiler from the University of Utah.

A few features, which were felt to require undesirably complex or expensive implementation, were omitted:

- There is no spaghetti stack
- There are no optional or &rest arguments. However arguments may be omitted, and default to NIL
- There is no lexical binding or similar mechanism
- Declaration mechanisms for the compiler. Generic arithmetic has been optimized as much as possible. Our philosophy has been to provide the best possible performance to "normal" users, rather than providing tools for experts to tune systems.

Currently work is being done on an implementation of Common Lisp, to be done with the same technology. Although it will have many of the things that were left out of ELISP, This will result in an appropriate penalty in runtime.

There is a subset of Interlisp implemented on top of ELISP. It implements "straightforward" parts of Interlisp, but not such things as the spaghetti stack or DWIM. (However it does implement a number of the macros that are normally done using DWIM.) The subset is sufficient to support UNITS, a widely-used AI system.

For more information on ELISP contact:

Rutgers University
Center for Computers and Information Services
Hill Center
PO Box 879
Piscataway, NJ 08854
(201) 932-3088