STANFORD ARTIFICIAL INTELLIGENCE LABORATORY June 13, 1968
OPERATING NOTE NO. 28.1

LISP 1.6

by Lynn Quam and John Allen

ABSTRACT: This note describes the LISP 1.6 system that runs on the PDP-6 computer in the Stanford A.I. Laboratory. The description is intended for readers who are generally familiar with the LISP 1.5.

ACKNOWLEDGEMENT: The PDP-6 LISP system was developed at M.I.T. and adapted for the Stanford A.I. Project by John Allen. This is a revision of SAILON-28, which was an adaptation of M.I.T. A.I. Memo No. 116.
"→" in the left hand margin indicates a new feature.

INTRODUCTION

This is a mosaic description of PDP-6 LISP, intended for readers familiar with the LISP 1.5 Programmer's Manual or who have used LISP on some other computer. Many of the features are subject to change. Thus, you should consult a PDP-6 systems programmer for any differences which may exist between LISP of June 1968 and present LISP on the system tape.

SOME DISTINCTIVE CHARACTERISTICS

Top-level input is to EVAL; there is no EVALQUOTE. Thus LISP's listen loop may be described by

```
(PROG NIL
    A    (TERPRI)
         (PRINT (EVAL (READ)))
         (GO A))
```

One types ATOM followed by a space to get the value of ATOM, or (FN (QUOTE arg1) (QUOTE arg2)...) to apply a function to explicit arguments. For this reason, users frequently arrange their top-level functions to be FEXPRs or FSUBRs to avoid the necessity of quoting explicit arguments. Alternatively, they may SETQ atoms to frequently-used or unwieldy argument lists, and apply their functions to these atoms.

(EQUAL fixed-point-number floating-point-number) is NIL. (ZEROP 0.0) is NIL. (EVAL T) is T. (EVAL NIL) is NIL. *T* and F do not exist.

There is no a-list. Current bindings of interpreted variables and free compiled variables are stored in the CDR of the variable's VALUE property, which is also the variable's SPECIAL cell.
Their values may be changed with SET and SETQ; CSET and CSETQ do not exist. Higher-level bindings of these variables are stored on the special pushdown list.
All interpreted variables and free compiled variables are automatically SPECIAL and provide complete communication between compiled and interpreted functions. COMMON does not exist.

Flags are not allowed; elements on a property list of an atom are expected to be paired.

MAP, MAPCAR, etc. assume the first argument is the function, and the second is the list of arguments to which it is to be applied.

There is no DEFINE; defining of functions is usually done with DEFPROP.

The available input-output devices are the on-line teletype, the disk, the DECtapes, and MAGTAPE and the line printer. For their use, see "Input-output".

One may allocate as much as one wishes (within limits) of each type of LISP storage each time LISP is started. See "Allocator" for how to use this feature.

The characters space, tab and comma are treated identically and are used to separate items on a list, and to distinguish dot-notation periods from periods denoting floating-point or decimal numbers. A sequence of spaces, commas, or tabs is equivalent to a single one. Spaces commas, or tabs adjacent to parentheses have no effect.

Object	Indicator	Relevant Information
ABS	SUBR	Absolute value.
ADD1	SUBR	Result is fixed or floating, same as argument.

Allocator

There are five storage areas in LISP:

a/ Free Storage: holds s-expressions
b/ Full Word Space: holds character strings of print names, floating point and large fixed-point numbers (see "Numbers").
c/ Binary Program Space: holds compiled functions and arrays.
d/ Special Pushdown List: holds higher-level bindings of all special variables.
d/ Regular Pushdown List: holds return addresses for sub-routine calls, bindings of all local variables, and is also used by various internal routines.

When LISP is loaded, the Allocator types out ALLOC? and waits for the user to type Y (yes) or N (no). If Y is typed, the Allocator responds with FULL WORDS= and waits for the user to type an octal number (ending with a space). The Allocator then similarly requests typed-in parameters for the size of Binary Program Space, Special Pushdown List, and Regular Pushdown List. Free Storage is given all remaining space. For any typed-in number, SPACE alone may be typed in and a standard value will be taken:

FULL WDS	4ØØ
BIN.PROG.SP.	2ØØØ
SPEC.PDL	1ØØØ
REG.PDL	1ØØØ

If N (or SPACE) is typed as the response to ALLOC? all standard values are taken. In case of error, RUBOUT will type an X and cancel the current value. See the appendix for reallocation of restarted LISP files and a description of the LISP MACROX loader.

AND — FSUBR — Returns NIL or last non-NIL argument: does not evaluate arguments past first NIL.

APPEND — LSUBR — Appends together any number of arguments, copying the top level of all but the last of them. (APPEND) is NIL and (APPEND X) is EQ to X.

APPLY	LSBR	(APPLY fn (args)) or (APPLY fn (args) alist). See EVAL for the formal of alist.
ARRAY	FSUBR	(ARRAY name par diml dim2 ...) sets up name as an array (actually as a SUBR). Par should be T for arrays whose elements are s-expressions, and will consequently be protected from garbage collection. Par should be NIL for arrays whose elements are floating point numbers (FLONUMS). A par from 1 to 36. specifies the byte size for a flexible byte size array, whose elements are fixed point integers (FIXNUMS). Both STORE and NSTORE evaluate their second argument before their first. An array may have no more than five dimensions; indices run from $\emptyset$ to dim-1. Arrays are stored in binary program space. ARRAY returns the base address of the array.
ARG	SUBR	See LSUBR.
ASCII	SUBR	The fixed point argument is converted into the corresponding single-character ASCII atom.
ASSOC	SUBR	Uses EQ.
BAKGAG	SUBR	(BAKGAG T) enables a backtrace of any LISP error except pushdown list overflow; (BAKGAG NIL) disables it. A backtrace is printed as a series of function calls as determined from the regular pushdown list, most recent first:

fnl-fn2	fnl calls fn2
fnl-EVALARGS	arguments being evaluated preparatory to calling fnl
fnl-ENTER	fnl being evaluated
?-fnl	? represents an internal routine.

BASE	VALUE	Current radix of fixed-point number output; may be modified with SETQ.
Binary Program Space		Storage area for compiled functions and arrays; usually is $2\emptyset\emptyset\emptyset$ registers long.

BOOLE LSUBR (BOOLE n a b c ...) causes a 36-bit bitwise Boolean operation to be performed on a and b, on the result and c, etc. The number n selects the operation as follows:

n	operation	n	operation
0	0	10	$\bar{a}.\bar{b}$
1	a.b	11	a≡b
2	$\bar{a}$.b	12	$\bar{a}$
3	b	13	$\bar{a}$ ∨b
4	a.$\bar{b}$	14	$\bar{b}$
5	a	15	a ∨$\bar{b}$
6	a ⊻b	16	$\bar{a}$ ∨$\bar{b}$
7	a ∨b	17	1

BPEND VALUE Highest location available for use as Binary Program Space.

BPORG VALUE Current lowest unused location in Binary Program Space.

CAR SUBR Left half. CAR of an atom is the "pointer" - 1.

CDR SUBR Right half. CDR of an atom is its property list.

CAAR, CADR etc. through CDDDDR

SUBRS Any combination of up to 4 D's and A's.

CHRCT SUBR The value of (CHRCT) is the number of characters remaining on the current output line.

COMPILER The compiler is a separate dump file called COMPLR requiring 24K. This core image also contains LAP. The compiler will compile LAP-able code onto to the selected output device. Error messages will appear on the teletype. At the conclusion of the compilation the list of functions compiled is printed on the teletype. One can continue to compile onto the selected file or do "(OUTC NIL T)" and close the file. In any case after compiling the last

functions, "(OUTC NIL T)" must be done to assure a proper end of file. Now "INC"-of-"INPUT" to the LAP-able file will begin input at the current value of BPORG. If one is finished with the compiler resetting BPORG to the origin of the compiler will reclaim the Binary Program Space. If a "clean system" is desired, read in LISP with appropriate core and allocate sufficient BPS with the allocator. Read in LAP and then input the compiled code. To compile a list of functions f1, f2,..., fn execute (COMPILE f1 f2 ... fn). To compile the currently selected input file onto the selected output file, execute (COMPL). The input file should be terminated by the atom T. The output file will contain both the LAP code and any sexpr on input file which is not an EXPR, FEXPR or MACRO.

COND	FSUBR	"COND pairs" of predicate and value may have other than two elements. If the predicate evaluates non-NIL, the remaining elements are evaluated in CAR-to-CDR order, and the value of the last is the value of the COND. If no predicate is true, the value of the COND is NIL.
CONS	SUBR	One of the five elementary function of LISP.
Current Problems		Compiled GOs and RETURNs happen when encountered; interpreted GOs and RETURNs happen when evaluation of the functions they are contained in (except PROG) is completed. Arguments to GO are repeatedly evaluated until atomic when interpreted, but are evaluated only once when compiled. Compiled GOs and RETURNs must appear explicitly in the PROG to which they apply, GOs and RETURNs occasionally confuse the compiler when appearing in compositions of functions other than PROG, COND, AND, OR, and NOT. Variables used free in a functional context must be declared SPECIAL or else the compiler will mistake them for undefined EXPRs or SUBRs.

CSYM	FSUBR	The initializing function for GENSYM. (CSYM SYM N) will set the five-character name, SYM, as the initially generated symbol for (GENSYM N). If CSYM is not used G0001 is assumed.
DEFPROP	FSUBR	(DEFPROP atom property indicator); value is atom.
DEPOSIT	SUBR	Causes the fixed-point number which is its second argument to be deposited in the absolute machine location which is its first argument.
→ DDTIN	FSUBR	(DDTIN T) selects DDT submode input on teletype. (DDTIN NIL) selects buffered line mode input on teletype. (DDTIN) returns T or NIL according to the currently selected input mode. In DDT submode, characters are input one at a time as opposed to buffered line mode in which an entire line (string of characters terminated by a carriage return, line-feed, or altmode) is unit of input from the teletype. This difference is most noticable when running in DDT submode under swapping conditions, which force the entire program to be swapped into core for each character. In line-mode, the program need be swapped into core only at the end of a line of input. For more information see READ.
→ DDTOUT	FSUBR	(DDTOUT T) selects single character teletype output (similar to DDT submode output). (DDTOUT NIL) selects buffered mode teletype output. (DDTOUT) returns T or NIL according to the currently selected output mode. Usually, one should use (DDTOUT NIL) for swapping efficiency. In buffered output mode, output is forced (see FORCE) whenever either the buffer is full, or a teletype input is requested.

DIFFERENCE	LSUBR	arg1 - arg2 - arg3 ... (DIFFERENCE) is $\emptyset$.
EDREAD	SUBR	The basic read function of the LISP editor. Calls on EDREAD are terminated by altmode or ellipsis (...). If ellipsis is used the next call an EDREAD will return NIL; otherwise EDREAD returns a list of the elements typed substituting %LP for (, %RP for) and %D for . .
EQ	SUBR	Will work for atoms and positive fixed-point numbers less than about $4\emptyset\emptyset\emptyset_{1\emptyset}$; otherwise use EQUAL.
EQUAL	SUBR	Floating point numbers are EQUAL only if they are exactly equal. To compare a fixed to a floating-point number, first float the former by performing (PLUS fixed-point number $\emptyset.\emptyset$).
ERRSET	FSUBR	(ERRSET expression arg) has the value (expression) if no error occurs while evaluating "expression". If an error occurs not caused by ERR, ERRSET returns NIL. if the error is caused by ERR, then the value is that returned by ERR. LISP error messages will be printed iff the second argument "arg" is T.
EVAL	LSUBR	If a second argument is given it is used instead of the present "p a-list". (EVAL number) is that number. In entering a function, arguments are evaluated from left to right. (EVAL atom) is (CDR (GET (QUOTE atom) (QUOTE VALUE))), unless there is no VALUE property or the above computation returns the pseudo-atom UNBOUND, in which case a LISP error occurs and the offending atom is printed, followed by the message "UNBOUND VARIABLE - EVAL." (EVAL x) where x is non-atomic and (CAR x) is atomic proceeds as follows: the interpreter determines (GETL (QUOTE CAR x) (QUOTE(SUBR EXPR FSUBR FEXPR LSUBR MACRO))), or, if this is NIL, (EVAL (CAR x)). If x has no VALUE property, a LISP error occurs and x

		is printed with the message "UNDEFINED FUNCTION." Having found a function it then evaluates its arguments if appropriate and applies the function to them. When (CAR x) is non-atomic it immediately evalutes the list of arguments i.e., (CDR x). It then compares (CAAR x) to LAMBDA, LABEL and FUNARG, taking appropriate action in each case. Failing this, it applies (EVAL (CAR x)) to the members of (CDR x).
EXAMINE	SUBR	Argument is number, which is taken as absolute machine address; value is contents of said address as a fixed point number.
EXPLODE	SUBR	Argument is s-expression; value is list of atoms whose print names are single characters, which concatenated would form the print name of the argument. For example, (EXPLODE (QUOTE FOO)) has the value (F O O). EXPLODE like PRIN1, inserts slashes, so (EXPLODE (QUOTE (QUOTE FOO/ BAR))) PRIN1's as (F O O // / BAR) or PRINC's as (F O O / B A R).
EXPLODEC	SUBR	EXPLODEC is to EXPLODE as PRINC is to PRIN1. Example: (EXPLODEC (QUOTE FOO/ BAR)) would PRIN1 as (F O O / B A R) or PRINC as (F O O B A R).
EXPR		Indicator for s-expression function.
FEXPR		Indicator for s-expression special form. Its lambda-list may have one or two members; the second is bound to the current "a-list".
FIX	SUBR	Argument is a fixed or floating-point number; value is truncated fixed-point value of argument.
FIXNUM		Indicator for a fixed-point number. See "Numbers".
FIX1A	SYM	(PUSHJ P FIX1A) Turns the actual fixed point number in 1 into a LISP number, q.v.

FLATSIZE	SUBR	(LENGTH (EXPLODE arg))
FLONUM		Indicator for floating-point number. See "Numbers".
→FORCE	SUBR	Causes the teletype output buffer to be printed on the teletype.
Free Storage		Storage area in LISP for s-expressions. Takes all memory which is free after the four other storage areas have been allocated. See "Allocator".
FSUBR		Indicator for special-form subroutine; the next pointer on the property list points to the first instruction in the routine. To use the current "a-list" a FSUBR must call *AMAKE, which leaves a pointer to same in 2.
Full Word Space		Storage area in LISP for character strings of print names, floating-point numbers, and negative fixed-point numbers, and large positive fixed-point numbers.
FUNARG		One of the three indicators the interpreter looks for when evaluating a non-atomic s-expression with a non-atomic CAR.
FUNCTION		To the interpreter, has the same effect as QUOTE. To the compiler, causes compilation of its argument.
GCGAG	SUBR	(GCGAG T) prints statistics after garbage collection; (GCGAG NIL) silences printout.
GC	SUBR	Takes no arguments, causes a garbage collection, and returns NIL.
GET	SUBR	(LAMBDA (A B) (COND((NULL (CDR A)) NIL) ((EQ (CADR A) B) (CADDR A)) ((GET (CDDR A) B)))) A typical use. (GET (QUOTE atom) (QUOTE indicator)), would return the property if it was found or NIL. If the property could be NIL, an amiguity could result in the second case.

GENSYM	SUBR	If the argument is NIL or an integer, 0 thru 6, then a generated symbol, G0001, ARG05, etc, will be produced. To set the initial value for any of these 8 symbol generators use CSYM. E.G. (CSYM FNC01 0) gives (GENSYM 0) value as FNC02, (CSYM B0001 NIL) gives (GENSYM) value as B0002. See also CSYM.
GETL	SUBR	Similar to GET but second argument is a list of indicators. The value returned is either NIL (indicating no property with any of the desired indicators was found) or such that (CAR (GETL ...)) is the indicator, (CADR (GETL ...)) is the property, and (CDDR (GETL ...)) is the rest of the property list. GETL like GET stops at the first satisfactory pair on the property list.
GO	FSUBR	If argument is non-atomic, it is repeatedly evaluated until it is atomic. However, see "Current Problems".
GREATERP	LSUBR	Takes any number of arguments; returns T only if arg1 > arg2 > arg3 ...; returns NIL otherwise.
IBASE	VALUE	Radix of fixed-point number input. May be modified with SETQ.
→ INITFN	SUBR	(INITFN (QUOTE FOO)) selects FOO as an initialization function which is evaluated whenever a LISP error to the outer level occurs, (including BELL). (INITFN) returns the currently selected initialization function.
→ INNO	SUBR	see TERPRI1.
Input-output		There are four functions which control I-O in the LISP system. They are: INC, OUTC, INPUT, and OUTPUT. All input in the LISP system comes from the teletype or from "the other input device". The function INPUT determines what this other device is and the function INC determines whether input comes from the teletype or from the secondary device. OUTPUT and OUTC are entirely analagous. The argument of INPUT

is the name of the input device to be used. If the input device is a DECtape the name of the file to be read is also required. Thus to read file FOOBAZ and dectape drive 3, the following could be used:

(INPUT DTA3: FOOBAZ)

Note that the name of the device must be followed by a colon and a space. The value of input will be T if the device is available, and in the case of DECtape, or disk, if the file can be found. If either of these conditions is not satisfied, an error message will be printed. The change in the source of input does not occur at the time the INPUT function is used, the change actually takes place when the function INC is executed with argument T. INC with argument NIL will cause input to be taken from the teletype. An automatic (INC NIL) is performed when LISP reads an end of file from a DECtape, and "*" is printed. OUTPUT works the same as INPUT except with the DECtape if the file does not already exist it will be created.

Since the main LISP loop consists of the cycle "READ", "EVAL", "PRINT", it is possible to read in function definitions by saying "(INC (INPUT DTA3: FOOBAZ))". If a function is to perform input from DECtape it is easiest to have the function perform (INC T). The input device can then be specified before calling the function and can easily be varied.

There are certain differences between INPUT and OUTPUT and between INC and OUTC. INPUT can be given a string of input files. (INC T) will then cause all of these files to be read in succession and the (INC NIL) will be performed only after seeing the end of file on the last file.

Thus to read FOO1 and FOO2 from DTA3 and FOO3 from logical device BAZ, use

(INPUT DTA3: FOO1 FOO2 BAZ: FOO3)

OUTC takes two arguments rather than only one as INC does. Namely OUTC is defined as follows:

1. (OUTC T T), output goes to the device and file described in the last (OUTPUT DEV: FILE) command.
2. (OUTC NIL T) end-of-file on last "device and file". Successive output goes to the teletype.
3. (OUTC NIL NIL) output changes to the teletype from "device and file". No end-of-file is given.
4. (OUTC T NIL) output returns to "device and file" from the teletype.

INTERN SUBR — Argument is pointer to atom structure; puts said atom on OBLIST and returns (probably new) atom pointer.

INUM SYM — The largest fixed-point number which may be represented by no more list structure than a "pointer" one larger than itself. See numbers.

LABEL — One of the indicators which EVAL looks for when evaluating an s-expression with a non-atomic CAR.

LAMBDA — One of the three indicators EVAL looks for when evaluating an s-expression whose CAR is non-atomic.

LAP — LAP is not part of the LISP on the system tape, but rather as a file of s-expressions on the system tape as file LAP. (LAP name indicator), where the indicator is SUBR, FSUBR, or LSUBR, causes LAP to call READ repeatedly, each time reading one tag or storage word. An atom is taken as a tag, except for NIL which indicates the end of the function being read; a non-atom s-expression is taken as a storage word in either the format (Inst Acc Adr) or the format (Inst Acc Adr Indx). Inst should be a PDP-6 instruction mnemonic, optionally suffixed @ (e.g., HLRZ@) for the indirect bit. Acc should be a number from Ø - 17, or P, the regular pushdown pointer. Adr may be a numeric machine address, a tag in that function, a negative number, one of certain symbols for entry points

to LISP internal routines, or a list in one of the following forms: (QUOTE atom) for a pointer to atom; (SPECIAL atom) for a pointer to atom's special cell (which is the CDR of the VALUE property); (E atom) for a pointer to the name of a function being called; or (C w x y z) for a pointer to a constant, i.e., a storage word containing (w x y z). Indx is an optional left-half quantity, such as an index register specification, of the same form as Adr. A SUBR may have no more than 5 arguments, the values of which are placed in accumulators 1 through 5 when the SUBR is called. A pointer to the argument list of an FSUBR is in 1. The arguments of an LSUBR are on the regular pushdown list, last on top, and - (number of arguments) is in 6. Accumulators 1 through 7 are the ONLY accumulators available for use within a subroutine, which is expected to return its value in accumulator 1. (PUSHJ P *AMAKE) will return a pointer to the "a-list" in accumulator 2.

Thus, a typical usage of LAP to get a hand-coded function into the system would be:

```
(LAP ABS SUBR)
    (PUSHJ P NUMVAL)
(MOVMS ∅ 1)
    (JCALL 2 (E MAKNUM))
NIL
```

Four UUO (trap) instructions are available for LAP: CALL, JCALL, CALLF and JCALLF. These are to be used for function calls in the form (CALL n (E fn)) where n is as follows: ∅ through 5, calling a SUBR or EXPR with ∅ to 5 arguments; 16 octal, calling LSUBR with arguments on pushdown list and - (number of arguments) in accumulator 6; 17 octal, calling an FSUBR or FEXPR with pointer to argument list in 1.
When one of these UUOs is first executed, the UUO handler will call the interpreter

if calling an EXPR or FEXPR, otherwise in the case of CALLF or JCALLF will execute a PUSHJ or JRST, respectively, to the specified function, and in the case of CALL or JCALL will execute the PUSHJ or JRST but in addition will change the UUO to a PUSHJ or JRST unless (NOUUO T) has been evaluated.
The F forms are necessary to call functions whose names are computed; the J forms save code in the case of (RETURN (fn ...)).

LAST SUBR
```
(LAMBDA (A)
        (COND ((ATOM (CDR A))A)
              ((LAST (CDR A)))))
```

LENGTH SUBR
```
(LAMBDA (A)
        (COND ((ATOM A) ∅)
              ((ADD1 (LENGTH (CDR A))))))
```

LESSP LSUBR
Takes any number of arguments, returns T only if arg1 <arg2 <arg3 ... etc., and NIL otherwise.

LINELENGTH SUBR
If the argument is NIL return the current line length. If the argument is a number, the line length is set to this value.

LIST FSUBR
(LAMBDA (A) (MAPCAR (FUNCTION EVAL) A))

LSH SUBR
Logical left shift of the value of the first argument by n places, where n is the value of the second argument, which may be negative.

LSUBR
Indicator for subroutine function which may take any number of arguments which are placed on the regular push down list, last on top. The negative number of arguments is placed in accumulator 6. In an s-expression function, the notation (LAMBDA var ...) where the "LAMBDA-list" is an atom, indicates that the variable var is to be bound to the number of arguments, and the values of the arguments may be gotten by calling (ARG 1), (ARG 2), etc.

MACRO

Indicator of a MACRO property, simulated by the interpreter and expanded at compile time. The property should be a function of one argument. When a call to this function is encountered, the list which is the function call (i.e., CAR of it is the function name) is fed to the macro definition as the one argument. Then the MACRO property function is expanded, for example:

```
(DEFPROP CONSCONS
         (LAMBDA (A)
                 (COND ((NULL(CDDR A)) (CADR A))
                       ((LIST (QUOTE CONS)
                              (CADR A)
                              (CONS (CAR A)
                                    ((CDDR A))))))
         MACRO)
```

would cause (CONSCONS A B C) to expand first, to (CONS A (CONSCONS B C)), then to (CONS A (CONS B C)), which is what would be interpreted or compiled.

MAKNAM SUBR

Argument is list of atoms whose print names are single characters (actually it takes the first character of each print name); value is pointer to s-expression which if printed out, would be the concatenation of the single characters which were its arguments.

MAKNUM SUBR

Turns the pointer which is its first argument into a LISP fixed - or floating-point number, according as whether its second argument evaluates to FIXNUM or FLONUM.

MAPC SUBR

First argument is function; second is argument list. Returns NIL, does no CONSes, otherwise identical to MAPCAR.

MAPCAR SUBR

First argument is function; argument is second.

MAPLIST SUBR

Function is first argument; argument list is second.

MEMBER	SUBR	Uses EQUAL.
MEMQ	SUBR	Like MEMBER, but uses EQ.
MINUS	SUBR	- (argument).
MINUSP	SUBR	Returns T if argument is less than ∅.
NCONC	LSUBR	Takes any number of arguments. (NCONC) is NIL. (NCONC x) is EQ to x.
NCONS	SUBR	(LAMBDA (A) (CONS A NIL))
NIL	VALUE	False value of predicates, explicitly tested for by COND. (EVAL NIL) is NIL (NIHIL ex nihilo). (MAKNUM NIL (QUOTE (FIXNUM) is ∅. NIL ends lists.
NOT	SUBR	Identical in value to NULL.
NOUUO	SUBR	(NOUUO T) prohibits the UUO handler from changing CALLs and JCALLs to PUSHJs and JRSTs, enabling tracing of the functions so called. (NOUUO NIL) restores that ability.
NSTORE	FSUBR	(NSTORE (NAME indx1 indx2 ...) number) stores number in the array element specified. This differs from STORE in that the number is not in LISP number format. NSTORE evaluates its second argument first and returns as value the second argument as with SETQ.
NULL	SUBR	(LAMBDA (A) (EQ A NIL))
NUMBERP	SUBR	Returns T if its argument is a number; NIL otherwise.

Numbers

Positive fixed-point numbers less than about 4∅∅∅ decimal are represented by "pointers" one greater than their value and no further list structure. Floating-point numbers and other fixed-point numbers are represented by atom structures whose property list contains a pointer to either one indicator FIXUM or the indicator FLONUM, and a pointer to a word in Full Word Space containing the actual number. For numeric input, see READ. Numeric output for

floating-point is decimal; otherwise, is in radix BASE, and if that radix is 1Ø decimal, LISP will end the number with a decimal point unless *NOPOINT has been set to T. The arithmetic functions (except MINUS) take any number of arguments, and use fixed-point arithmetic until the first floating-point operand is encountered, at which time the current result and all succeeding fixed-point operands are floated, and the result is floating-point.

NUMVAL	SYM	(PUSHJ P NUMVAL) with a LISP number in 1 returns a machine number in 1.
OBLIST	VALUE	The object list, a list of buckets of
OPS	FEXPR	or FSUBR Part of LAP. Takes pairs of arguments of the form sym value, gives each sym a SYM property of value.
OR	FSUBR	Takes any number of arguments; returns first non-NIL argument or NIL; does not evaluate arguments past the one it returns.
PLUS	LSUBR	arg1 + arg2 + arg3 ...
PNAME		Indicator for print name property. The property is a list of pointers to words in full word space containing the actual character string.
PRINC	SUBR	Prints any s-expression; does not insert slashes before characters which must be quoted with a slash on type-in. PRINC does not print a space before or after what it prints.
PRINT	SUBR	Identical to (PROG2 (TERPRI) (PRIN1 arg) (PRINC (QUOTE /))).
PRIN1	SUBR	Prints any s-expression; inserts slashes before characters which would otherwise be syntactically incorrect as part of an atom's print name such as +, (, etc.

be syntactically incorrect as part of an atom's print name, such as an initial digit or (or ., may be quoted with the character / . / itself may be quoted this way.

If an illegal atom occurs while typing, (e.g., 8A), READ will enter a "Bell" loop and will ignore all characters except rubout and Bell. Rubout will erase the illegal atom and return control to READ. "Bell" as usual will return control to the top level.

READCH SUBR Reads one character from the selected input device; see "Input-Output". The various characters which control LISP's input-output switches are seen and treated as any other characters, but still have their usual effects.

READLIST SUBR Similar to MAKNAM q.v. , but automatically INTERNs any atoms appearing in the resulting s-expression.

Regular Push Down List Storage area in LISP containing returns for function calls and bindings of all local variables. Normally contains 1ØØØ registers.

REMLAP FEXPR Part of LAP. REMOBs all functions associated with LAP. With an optional argument of T, removes the SYM property of all atoms which possess it.

REMOB FSUBR Takes any number of atomic arguments; removes them from the OBLIST and returns NIL.

REMPROP SUBR Removes the property with the indicator which is the value of its second argument from the atom which is the value of its first argument. Returns T if the property was there, and NIL otherwise.

RETURN SUBR See "Current Problems".

REVERSE SUBR Reverses the top level of a list, using CONS.

RPLACA	SUBR	Replaces CAR of the value of its first argument with the value of its second argument.
RPLACD	SUBR	Same as RPLACA, but uses CDR.
SASSOC	SUBR	Uses EQ.
SET	SUBR	Causes the value of its second argument to be placed, with RPLACD, in CDR of the VALUE property of the value of its first argument.
SETQ	FSUBR	(SET (QUOTE arg1) arg2).
SPEAK	SUBR	Takes no arguments, returns current value of CONS counter.
SPECBIND	SYM	(PUSHJ P SPECBIND) (Ø Ø var1) (Ø n var2) ... causes the current binding of the special variables var1 and var2 to be saved upon the Special Push Down List, and causes var1 to be bound to NIL and var2 to be bound to the contents of accumulator n.
SPECIAL	FEXPR	Part of the compiler. Declares all of the atoms which are in its list of arguments to be SPECIAL. However, all free variables in compiled functions are automatically special. There is no UNSPECIAL; use (REMPROP (QUOTE atom) (QUOTE SPECIAL).
Special Push Down List		storage in UTSP where higher-level bindings of special variables are stored. Normally 1ØØØ registers long.
SPECSTR	SYM	(PUSHJ P SPECSTR) restores most recent batch of special variable bindings.
STORE	FSUBR	(STORE (name indx1 indx2 ...) value) sets the value of the specified cell in the array name to the value value. See "ARRAY". Evaluates its second argument first.
SUBR		Indicator for subroutine function property. Property is pointer to first instruction in routine.

SUBST	SUBR	Uses CONSes. Use (SUBST ∅ ∅ arg) to copy the value of arg.
SUB1	SUBR	Value is fixed or floating, same as argument.
SYM		Indicator for symbol value property. Used by LAP.
T	VALUE	True value of predicates. (EVAL T) is T (Veritas numquam perit).
TERPRI	SUBR	Takes no arguments; prints a carriage-return-line-feed, and returns NIL.
→ TERPRI1	SUBR	Like TERPRI except it will preface each line with an EDIT2 line number. Use INNO to initialize numbering.
TIME	SUBR	Returns in fixed point the number of milliseconds your job has been computing.
TIMES	LSUBR	arg1 * arg2 * arg3 ...
→ TYI	SUBR	Logical inverse of TYO, i.e., inputs a single character in ASCII code.
TYO	SUBR	Outputs the character whose ASCII value is the value of its one argument.
VALUE		Indicator for value property. The value cell/special cell is CDR of the VALUE property.
XCONS	SUBR	(CONS arg2 arg1)
ZEROP	SUBR	(ZEROP ∅.∅ is NIL).
*AMAKE	SYM	(PUSHJ P *AMAKE) returns a pointer to the current "a-list" in 2.
*APPEND	SUBR	APPEND of two arguments.
*DIF	SUBR	DIFFERENCE of two arguments.
*EVAL	SUBR	EVAL of 1 argument.

*FUNCTION	SUBR	Causes FUNARG binding when encountered; otherwise identical to FUNCTION.
*GREAT	SUBR	GREATERP of two arguments.
*LCALL	SYM	(JSP 3 *LCALL) in an LSUBR which has been called by the interpreter puts the number of arguments in 1, and a pointer to the arguments on the top of push down list.
*LESS	SUBR	LESSP of two arguments.
*NOPOINT	VALUE	When set to T, inhibits the printing of decimal points after typeout of decimal numbers. Setting *NOPOINT to NIL restores this typeout.
*PLUS	SUBR	PLUS of two arguments.
*QUO	SUBR	QUOTIENT of two arguments.
*RSET	SUBR	(*RSET T) inhibits the rebinding of special variables to top-level values when a LISP error occurs; (*RSET NIL) permits this rebinding.
*TIMES	SUBR	TIMES of two arguments.